Structures Cellulaires Non-Standards en Bois

Création d'un outil paramétrique d'aide à la conception et à la fabrication

Stage au Centre de Recherche en Architecture et Ingénierie

Mémoire de Master Design Global
Spécialité Architecture Modélisation et Environnement

Thomas EHRHARDT - Guillaume GINEFRI
Sous la direction de : Gilles DUCHANOIS et Oskar GAMEZ
Soutenu le 7 Septembre 2016

Structures Cellulaires Non-Standards en Bois

Création d'un outil paramétrique d'aide à la conception et à la fabrication

Stage au Centre de Recherche en Architecture et Ingénierie

Mémoire de Master Design Global
Spécialité Architecture Modélisation et Environnement

Thomas EHRHARDT - Guillaume GINEFRI
Sous la direction de : Gilles DUCHANOIS et Oskar GAMEZ
Soutenu le 7 Septembre 2016

Remerciements

Nous tenions tout particulièrement à remercier Gilles Duchanois et Oskar Gámez pour nous avoir confié ce travail de recherche très riche et intéressant ainsi que pour la confiance et l'autonomie qu'ils nous ont laissé.

Merci également pour tout le temps que vous nous avez accordé à nous donner des pistes, aides et améliorations en tout genre.

Merci également à Julien Meyer pour ses précieux conseils et les échanges que nous avons pu avoir et qui ont enrichi ce travail.

Enfin, nous tenions également à remercier tout le personnel du CRAI, chercheurs, doctorants, et stagiaires pour le cadre, les échanges, les conseils et l'ambiance de travail.

SOMMAIRE

I. INTRODUCTION

1. CONTEXTE DU STAGE

Ce stage finalise notre formation en double cursus «Architecture, Modélisation et Environnement». Il a été effectué à mi-temps lors de notre deuxième semestre de master 2 au Centre de Recherche en Architecture et Ingénierie à l'École Nationale Supérieure d'Architecture de Nancy, et s'inscrit donc dans la même temporalité que notre projet de fin d'études.

Notre choix s'est porté sur ce stage en particulier car il nous a semblé constitué une opportunité de découvrir le travail de recherche à travers une thématique qui nous intéresse : l'utilisation d'outils de conception innovants et performants qui permettent de définir un projet jusque dans les détails précis de sa réalisation technique. Ayant déjà travaillé ensemble sur plusieurs projets en s'initiant à la modélisation paramétrique avec Rhinocéros et Grasshopper, il nous a semblé intéressant de pouvoir suivre ce stage en binôme. De plus, nous avons mené en parallèle à ce stage un travail de PFE commun dont le sujet est proche et a pu enrichir notre démarche car il concerne des structures cellulaires non standard en bois.

Dans ce document de synthèse de notre travail de recherche, nous commencerons par vous présenter un état de l'art sur l'évolution et l'importance des moyens de représentations de l'architecture, du dessin traditionnel à la modélisation paramétrique. Cet état de l'art a pour but d'introduire de manière générale comment fonctionne la modélisation paramétrique, quel est son intérêt et quelle influence peut avoir cet outil sur la conception architecturale.

Nous vous présenterons ensuite les objectifs de ce stage et la méthode que nous avons mis en place pour y parvenir. Une majeure partie de ce mémoire sera ensuite consacrée à la description du fonctionnement des algorithmes développés, afin de faciliter une éventuelle poursuite ultérieure de ce travail.

Enfin, nous conclurons par un bilan sur le travail effectué, en présentant les points forts et les pistes d'améliorations envisageables pour cet outil.

2. ÉTAT DE L'ART

a. Du dessin traditionnel...

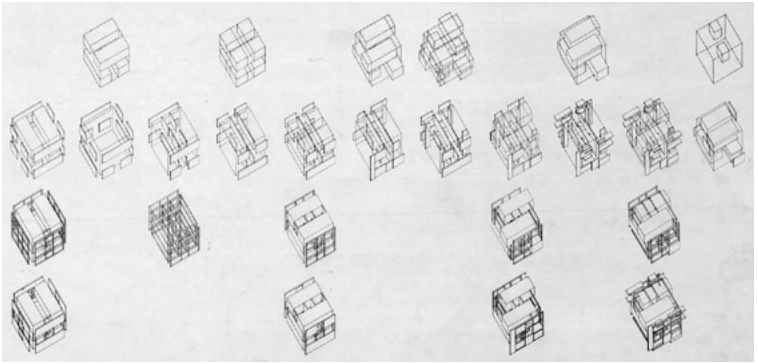
*« Architects do not make buildings, they make **drawings** of buildings ». Robin Evans*

*« Les architectes ne construisent pas de bâtiments, ils **dessinent** des bâtiments ».*

Les architectes ont toujours dessiné avant de construire, le dessin étant l'outil qui leur permet d'organiser leurs idées, les espaces et la mise en œuvre des matériaux. C'est le moyen de figurer et de faire exister un projet avant sa réalisation, dans le but de le communiquer mais également d'anticiper les problèmes techniques et de les résoudre avant la construction.

1. Pierro della Francesca, la cité idéale (Urbino), peinture à l'huile réalisée vers 1500. Un des premiers exemples de perspective à un point de fuite.

Au cours des siècles l'évolution des moyens de représentation a fait évoluer et émerger de nouveaux styles architecturaux comme la perspective à la renaissance (fig. 1) ou l'axonométrie durant le modernisme (fig. 2), à travers laquelle les architectes vont chercher à exprimer et communiquer la complexité de leurs projets. Ces outils ont marqué à leur époque une avancée majeure dans la conception qui montre à quel point la conception architecturale et la capacité à représenter l'architecture sont liés.



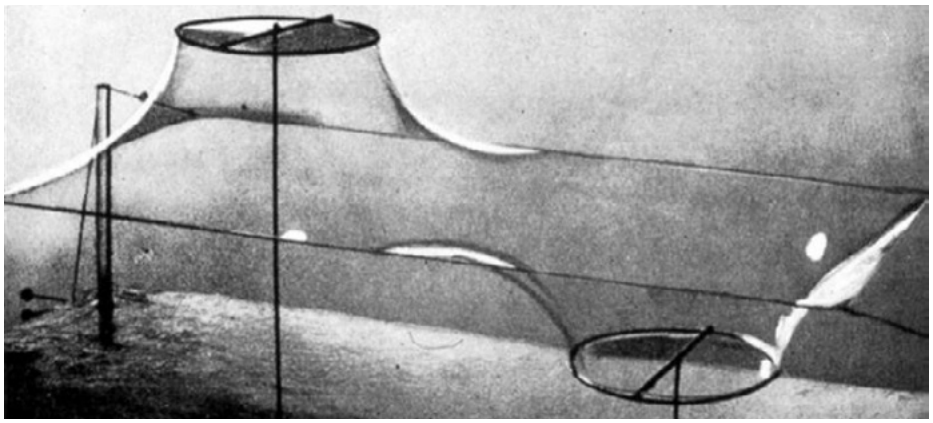
2. Peter Eisenman, House IV, 1971. Opérations géométriques qui mènent au projet final.

En effet, si tracer des figures régulières telles que des droites, polygones, cercles ou même des ellipses est relativement simple à condition que l'on dispose du bon outil, plus les formes deviennent complexes et irrégulières plus elles deviennent difficiles à représenter. De plus, le processus de dessin exclu les aspects physiques qui régissent la génération des formes dans le monde réel. Par exemple, un dessin ne permet pas de gérer les forces comme la gravité et les contraintes qui affectent et restreignent certaines déformations et déplacements. Ces limites ont restreint l'exploration architecturale et les concepteurs ont parfois été forcés de réutiliser des systèmes tectoniques plutôt qu'innover.

b. ... à la conception paramétrique

En dépit de ces limitations, le dessin a été le moyen de concevoir l'architecture à travers les siècles. Cela a été rendu possible par l'utilisation par les architectes de typologies, qui sont des solutions techniques éprouvées. La typologie ne fait plus du dessin un simple moyen de représentation mais un système qui permet au concepteur d'explorer et d'affiner des variations d'une liste spécifique de formes et de contraintes structurelles qui ont fait leurs preuves.

Une nouvelle approche émerge à la fin du XIX^{ème} siècle. Des pionniers comme Gaudi (1852-1926), Isler (1926-2009) et Otto (1925-2015) ont rejeté les typologies classiques et se sont intéressés à des processus d'auto-formation existants dans la nature. La conception des formes architecturales s'effectue alors par la contrainte et l'expérimentation. Comme la forme ne peut provenir de solutions éprouvées, le dessin traditionnel ne peut pas être utilisé comme un outil pour prédire l'architecture.

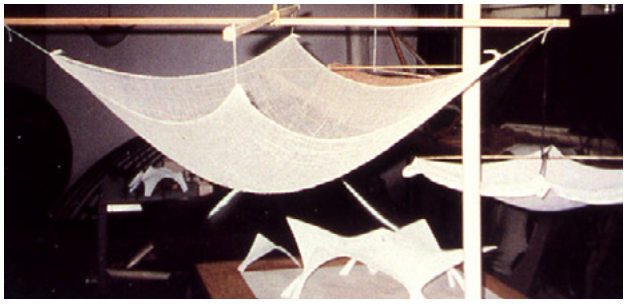


1. A gauche, maquette d'étude utilisant un film d'eau savonneuse par Frei Otto. 2. A droite, maquette d'étude utilisant la chaînette renversée par Gaudi.

Pour cette raison les pionniers se sont appuyés sur des maquettes qui leur ont permis d'expérimenter et de prédire le comportement de ces nouvelles architectures :

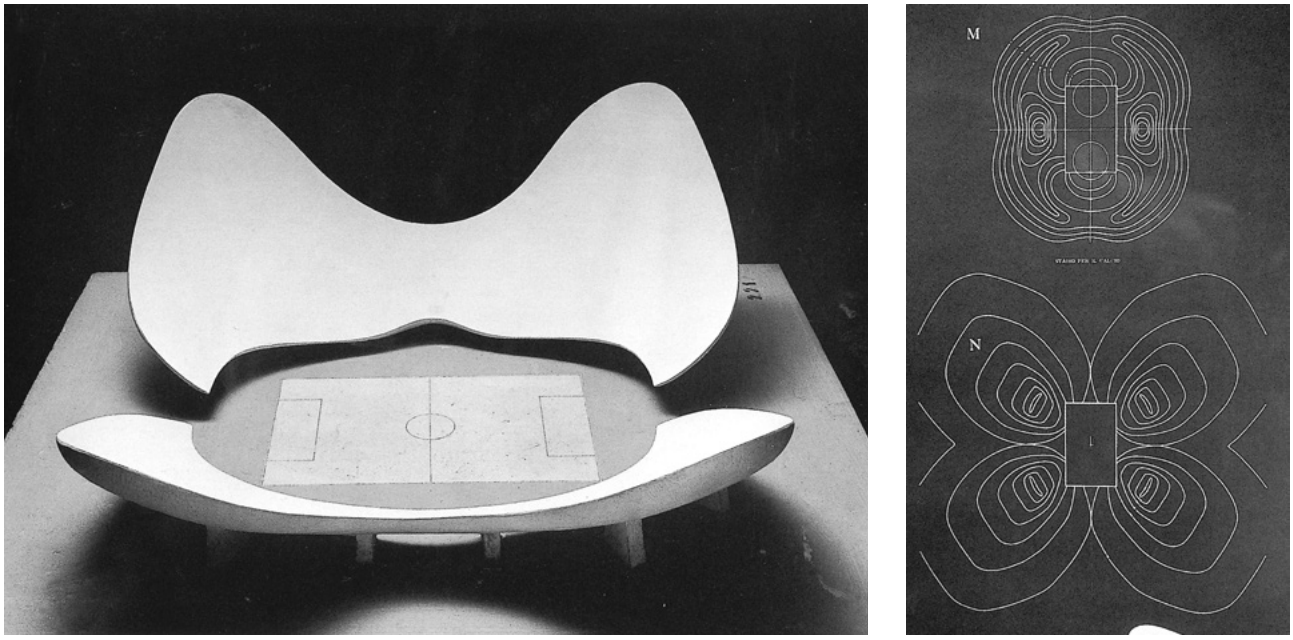
- Films d'eau savonneuse pour trouver les surfaces minimales qui relient plusieurs courbes (fig. 1)
- Structures suspendues qui permettent de trouver une voûte de compression optimale suivant le principe des arcs en chaînette renversée (fig. 2)

Le moyen de conception n'est alors plus le dessin mais bien la maquette, qui permet d'obtenir de nouvelles formes architecturales auto-optimisées en fonction d'une contrainte : la gravité terrestre (fig. 3). D'une certaine manière on peut parler de conception «mono-paramétrique».



3. Station service à Deitingen sur l'autroute A1 Zurich-Bern par Heinz Isler, 1968. La maquette d'étude a permis de trouver la forme la plus adaptée à donner à la voûte en béton en fonction de la gravité.

C’est en 1939 qu’est inventé le terme «architecture paramétrique» par l’architecte italien Luigi Moretti (1907-1973). Il s’agit de prendre en compte dans le processus de génération d’une forme architecturale des données de différentes natures : géométriques, physiques, environnementales ou d’usages. Ses projets de stades innovants présentés en 1960 sont conçus de manière «paramétrique». En collaboration avec le mathématicien Bruno De Finetti, Luigi Moretti cherche par des règles mathématiques et le tracé de courbes paramétrées à développer une forme architecturale permettant d’offrir le meilleur angle de vue depuis n’importe quel siège des tribunes tout en respectant des contraintes d’économie de matière et donc de budget (fig. 4).

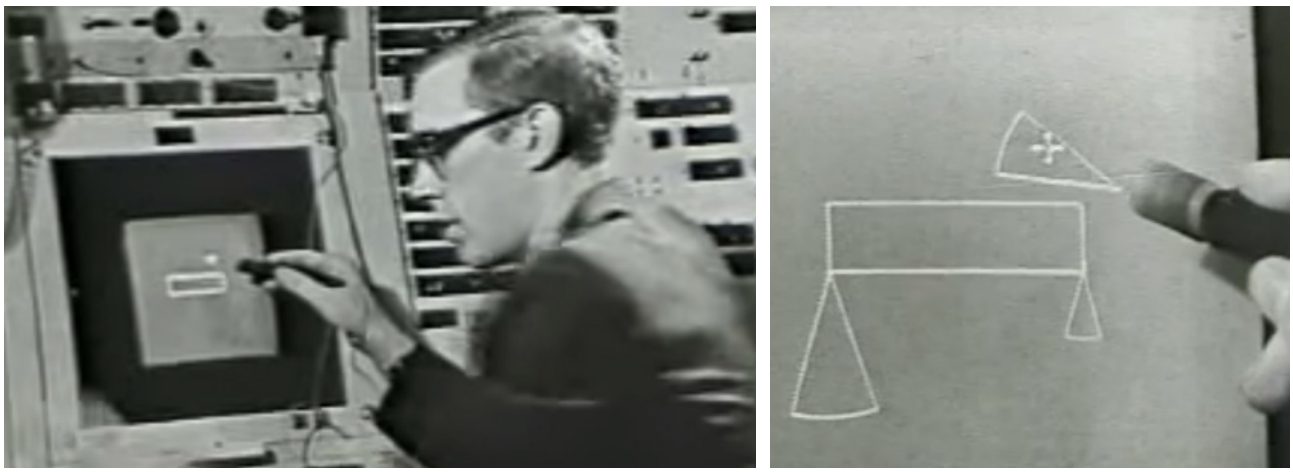


4. «Architecttura Parametrica» par Luigi Moretti à l’Exposition Triennale de Milan, 1960. Maquette d’un stade de football et les diagrammes dessinés pour générer la géométrie des tribunes.

Bien que l’informatique n’en soit encore qu’à ses débuts, Moretti est persuadé que la conception basée sur des paramètres va devenir le code du nouveau langage architectural, porté par la logique, les mathématiques et les ordinateurs. Pour lui, l’intérêt des ordinateurs est de permettre au concepteur d’exprimer des paramètres et leurs relations qui seront appliquées au projet à travers une liste de routines d’auto-correction jusqu’à atteindre la meilleure configuration possible.

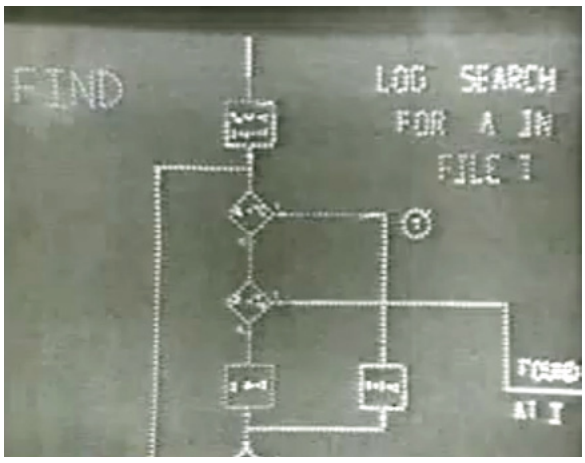
c. Du dessin assisté par ordinateur...

L'intuition qu'a Moretti concernant les possibilités qu'offre l'ordinateur se confirme rapidement avec l'apparition en 1963 du premier programme interactif de Dessin Assisté par Ordinateur (DAO) : «Sketchpad», développé par Ivan Sutherland, considéré comme un des programmes les plus influents jamais écrit (fig. 1).



1. Ivan Sutherland effectuant une démonstration de l'interface de son programme Sketchpad en 1963.

Sketchpad est la transposition du dessin à la main dans le monde numérique. Le logiciel est le premier à proposer une interface graphique permettant à l'utilisateur de dessiner directement sur l'écran à l'aide d'un stylo lumineux, mais le programme va plus loin et constitue une révolution. En effet, le dessin traditionnel sur papier suit une logique d'addition : chaque symbole est constitué de traits que l'on ajoute successivement jusqu'à atteindre le résultat souhaité. Le Dessin Assisté par Ordinateur (DAO) suit quant à lui une logique d'association. Par exemple, si deux lignes sont dessinées en commençant par le même point A, alors chaque mouvement de A impliquera une modification des deux lignes. Les contraintes peuvent être combinées et définir des relations entre les objets, dépassant la logique d'addition du dessin conventionnel. Dans Sketchpad, l'utilisateur peut visualiser et intervenir sur ces contraintes à travers un diagramme appelé «Flow chart», la modification des relations dans le diagramme entraînant directement une modification du dessin (fig. 2).



2. Le «Flow chart», diagramme qui régit les relations entre les objets dans Sketchpad.

Ainsi Sketchpad a prouvé que les ordinateurs n'étaient pas seulement destinés à être utilisés dans l'ingénierie et le dessin répétitif, mais pouvait potentiellement être utilisé de manière interactive par les concepteurs et artistes pour la recherche d'idées innovantes durant la phase de conception.

Il faudra néanmoins attendre le milieu des années 1980 pour voir arriver en France les ordinateurs personnels équipés des premiers programmes grand publics de DAO tels que Autocad (1982) et que les architectes commencent à l'utiliser. Le concepteur peut désormais copier et modifier son dessin plutôt que de le recommencer entièrement, «copier/coller» des éléments répétitifs, effectuer des zoom et changements d'échelles, demander l'affichage automatique de hachures ou de cotes, ce qui offre un gain de temps et de précision considérable. Pourtant, les capacités d'associations introduites par Sketchpad vingt ans plus tôt sont peu utilisées dans les logiciels de DAO.

En 1987, la commercialisation du logiciel Pro/ENGINEER développé par Samuel Geisberg pour la conception de systèmes mécaniques marque une autre étape importante : le début de la Conception Assistée par Ordinateur (CAO). Contrairement à la DAO où un trait est un trait, en CAO il ne s'agit plus de «dessiner» mais de construire virtuellement un projet. Les objets 3D sont associés à des informations supplémentaires qui permettent au logiciel de classer et générer la géométrie, on parle alors d'une modélisation «orientée objet». Chaque opération (ou fonction) utilisée durant la modélisation est définie par un comportement mécanique et des paramètres (longueurs, angles…) qui peuvent être modifiés facilement par la suite. Lors de la modification de ces paramètres, la géométrie des objets est alors recalculée et les documents associés sont automatiquement mis à jour : plans, coupes, assemblages… Il est alors possible de changer instantanément l'épaisseur d'un matériau, le profil d'une structure complexe, ajuster le diamètre de centaines de trous en même temps. Pro/ENGINEERS a permis de réduire le coût des modifications lors de la conception et de dépasser les contraintes de la simple modélisation 3D.

L'arrivée de la DAO puis de la CAO marque une modification en profondeur des outils de l'architecte… Mais pas réellement de la manière de concevoir l'architecture comme l'imaginait Moretti et comme les avancées de Sketchpad le suggéraient.

Les progrès les plus importants ont été réalisés entre la fin des années 1980 et aujourd'hui. La recherche académique s'est attachée à essayer d'échapper à la limite de la simple édition d'un modèle. Ces travaux ont permis d'explorer de nouvelles manières de manipuler les logiciels «de l'intérieur» afin de trouver des solutions et des formes inexplorées à travers la programmation. Beaucoup de concepteurs ont ainsi compris que des programmes plus sophistiqués permettaient de gérer la complexité au delà des capacités humaines en structurant des routines et des procédures. Ce type de modélisation relié aux langages de programmation donne des instructions qui peuvent être exécutées par un ordinateur par le biais d'une procédure pas à pas : l'algorithme.

d. ... à la modélisation paramétrique

La modélisation paramétrique ne doit pas être confondue avec la CAO décrite précédemment qui s'appuie également sur la notion de paramètres. On entend en fait par la désignation «modélisation paramétrique» une stratégie de modélisation basée sur l'utilisation d'algorithmes. En modélisation paramétrique, l'utilisateur ne dessine pas directement des éléments géométriques avec sa main ou sa souris mais conçoit des algorithmes qui permettent ensuite à l'ordinateur de générer cette géométrie. Mais qu'est-ce qu'un algorithme ?

«Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre un problème ou d'obtenir un résultat»

Wikipédia - https://fr.wikipedia.org/wiki/Algorithme (page consultée le 20/07/2016)

«Ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations. Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur.»

Larousse - http://www.larousse.fr/dictionnaires/francais/algorithme/2238 (page consultée le 20/07/2016)

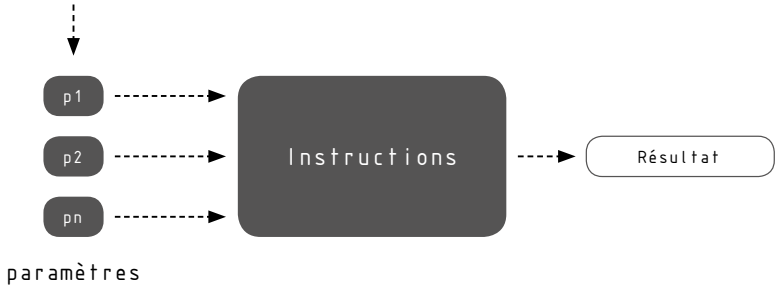
L'algorithme suit la capacité de l'être humain à diviser un problème en plusieurs opérations qui peuvent être accomplies simplement. Bien qu'ils soient fortement associés à l'informatique, les algorithmes peuvent être définis indépendamment d'un langage de programmation. Par exemple, une recette de cuisine peut être considérée comme une sorte d'algorithme. On peut effectivement décrire une procédure pour préparer une pizza :

- étaler la pâte
- disposer les ingrédients sur la pâte
- cuire la pizza au four
- sortir la pizza du four

Cependant une procédure ne peut être considérée comme un algorithme si les instructions restent ambiguës: «disposer les ingrédients», mais quels ingrédients ? Combien de temps cuire la pizza ? A quelle température ?

Cet exemple simple montre les principales caractéristiques d'un algorithme :

- Les instructions d'un algorithme sont précises et sans ambiguïtés
- Un algorithme nécessite des paramètres en entrée : ici type et quantité de chaque ingrédients, température et temps de cuisson…
- Un algorithme produit un résultat spécifique (fig. 1)



1. Représentation schématique d'un algorithme.

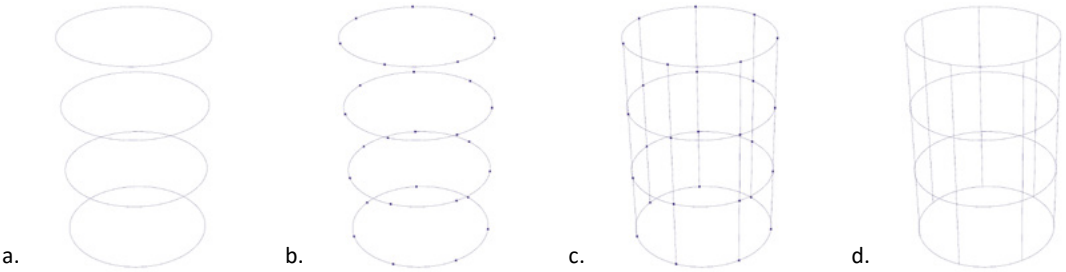
Les algorithmes peuvent être de différentes natures : numériques, logiques… Ils peuvent également produire et manipuler des données géométriques. Par exemple (fig. 2):

a. Tracer 4 cercles

b. Diviser les 4 cercles en N parties : on obtient N points pour chaque cercle

c. Connecter les points correspondants

d. Résultat de l'algorithme



a. b. c. d.

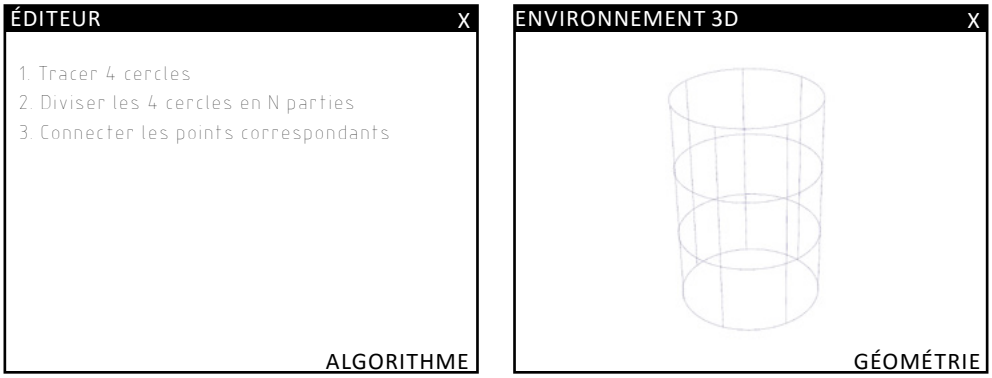
2. Illustration de la géométrie produite par l'algorithme à chaque instruction

Les algorithmes peuvent être exécutés par un ordinateur par le biais d'un langage de programmation que l'on saisi dans un éditeur. Il existe de nombreux langages de programmation standards (C#, VB.NET, Python…) ou plus spécifiques aux arts visuels (Processing), voir directement intégrés à des logiciels de modélisation (Autolisp pour Autocad, Ruby pour Sketchup, Rhinoscript pour Rhinocéros, MEL dans MAYA). La modélisation paramétrique que l'on peut également désigner par le terme de modélisation algorithmique, est constituée de deux environnements de travail (fig. 3):

- L'éditeur
- L'environnement de modélisation 3D

Ces deux environnements produisent deux résultats :

- L'algorithme
- Le résultat de l'algorithme : la géométrie qui lui est associée.

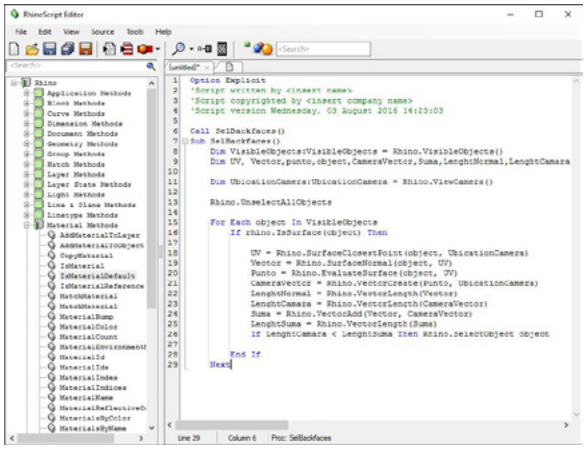


3. Représentation schématique des deux environnements de travail de la modélisation algorithmique

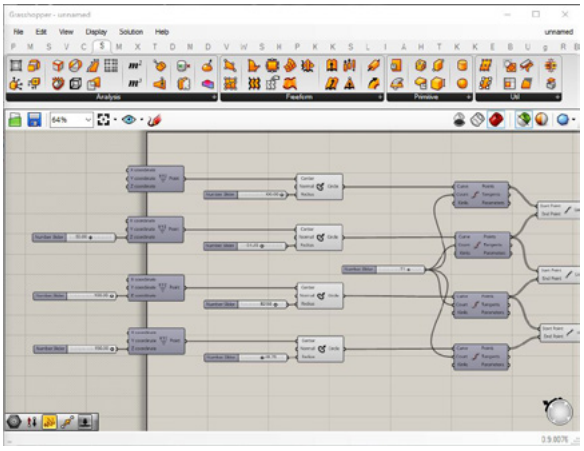
Ces éditeurs intégrés à des logiciels de modélisation permettent à l'utilisateur de manipuler la géométrie non plus avec la souris mais par une liste d'instructions et de procédures. Le résultat obtenu n'est pas un simple modèle 3D, mais peut-être considéré comme un modèle numérique interactif qui répond aux variations des paramètres en temps réel. En plus de pouvoir ajouter de nouvelles fonctions ou d'automatiser des tâches spécifiques, la modélisation paramétrique transforme le lien entre une idée et le résultat final, en permettant au concepteur de créer un processus plutôt qu'un simple objet figé.

4. Marc Fornes avec Vincent Novak et Claudia Corcilius (Theverymany). Explorations sur les processus de croissances programmées, 2006.

L'intérêt de cette méthode de conception est qu'elle peut mener à des résultats non imaginés au préalable par le concepteur, des résultats imprévus mais qui correspondent aux règles et aux paramètres définis. La modélisation paramétrique a le potentiel de générer et contrôler une complexité au delà des capacités humaines, notamment de grandes quantités de données et de produire de nombreuses variations et variantes (fig. 4). Elle stimule la créativité du concepteur en produisant des résultats nouveaux et inattendus. Ainsi, la modélisation paramétrique est un outil interactif, une force de proposition qui va au delà des possibilités de conception offerte par les logiciels conventionnels de DAO, CAO et de modélisation 3D.



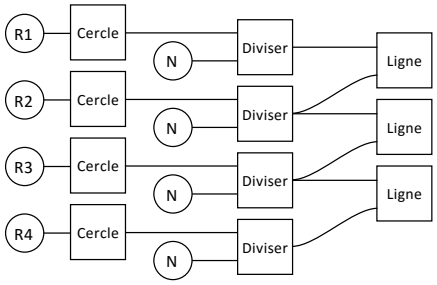
5. Programmation sur RhinoScript.

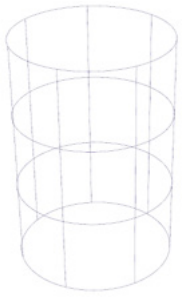


6. Programmation visuelle sur Grasshopper.

Néanmoins, dans le contexte de la conception architecturale ces possibilités sont bien souvent restreintes par la «barrière de la langue», les architectes étant rarement formés à la programmation informatique (fig. 5). Par le biais de la programmation visuelle, la modélisation algorithmique est rendue accessible aux non-programmeurs (fig. 6). Les lignes de codes sont remplacées par un diagramme constitué de nœuds et de connexions qui représentent les fonctions et leurs paramètres. De plus en plus de logiciels proposent des interfaces de programmation visuelle ce qui permet de rendre accessibles les outils au plus grand nombre, à l'image de Dynamo intégré au logiciel de CAO Revit, Generative Components de Bentley anciennement associé à l'utilisation de Microstation ou encore de Grasshopper, plugin gratuit de Rhinocéros le plus reconnu dans son domaine à l'heure actuelle. Les éditeurs de logiciels ainsi que les architectes commencent à prendre conscience des possibilités offertes par ces outils qui évoluent rapidement et dont l'accès tend à se généraliser. Preuve de cet engouement, Archicad a lancé en 2016 une version bêta qui propose d'intégrer une connexion directe avec Grasshopper et Rhinocéros.

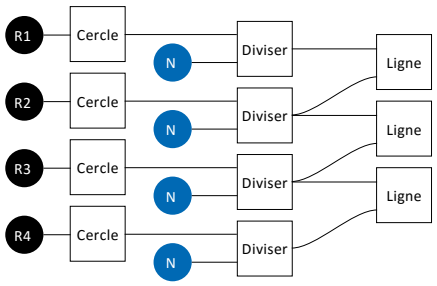
Le diagramme est constitué de nœuds et de connexions. L'avantage de la programmation visuelle est qu'elle permet de manipuler des algorithmes de manière intuitive. Reprenons l'exemple précédent (fig. 7). Les carrés représentent ici les fonctions (dessiner un cercle, diviser un cercle, créer une ligne à partir de deux points) et les cercles les paramètres (rayon, nombre de divisions).

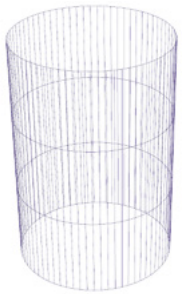




7. Principe de la programmation visuelle

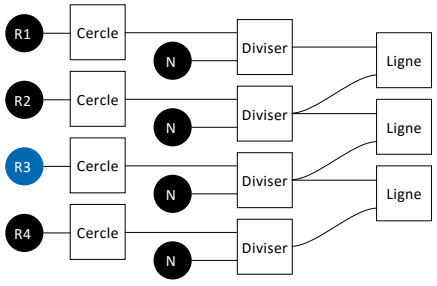
Lorsqu'un paramètre est modifié, par exemple N le nombre de divisions, le modèle génère les lignes supplémentaires correspondantes (fig. 8)

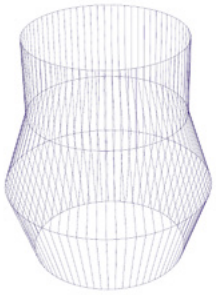




8. Modification du nombre de lignes

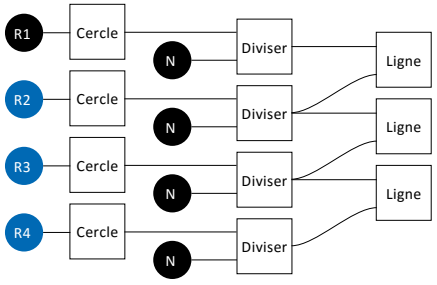
Les paramètres peuvent être modifiés indépendamment les uns des autres, ici le rayon d'un cercle est augmenté (fig. 9)

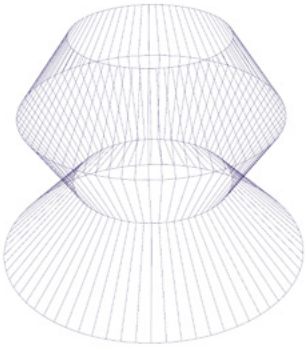




9. Modification d'un rayon

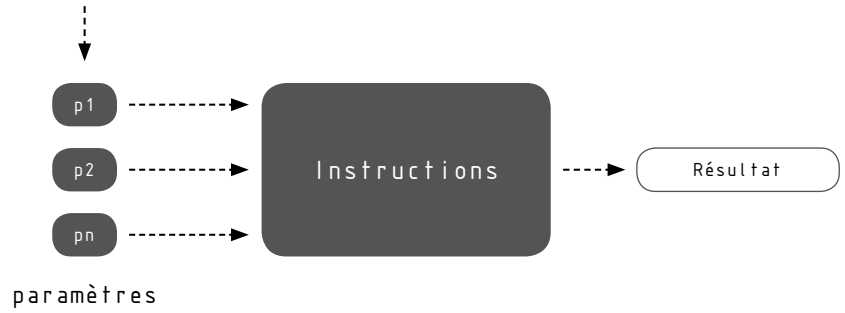
Cette méthode de conception permet d'explorer de nombreuses configurations et variantes en contrôlant les paramètres d'entrée (fig. 10)



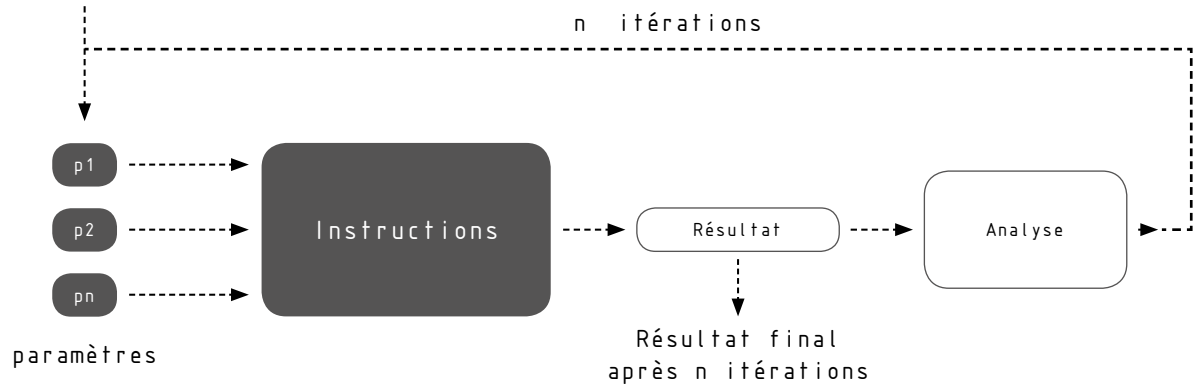


10. Modification de plusieurs rayons

La modélisation paramétrique permet autant une démarche de conception traditionnelle dans laquelle on cherche à obtenir une forme par une approche plastique (fig. 11), qu'une démarche de conception paramétrique où la forme est le résultat d'un processus et de contraintes extérieures (fig. 12). Propice à l'innovation et la recherche formelle, cet outil permet également de gérer de grandes quantités de données, d'élaborer des projets très complexes et de les maîtriser très précisément jusque dans leur réalité physique et constructive.



11. Représentation conceptuelle d'une approche traditionnelle de la conception



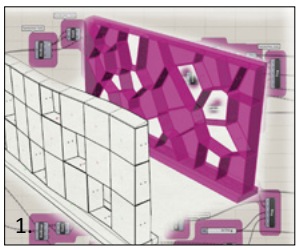
12. Représentation conceptuelle d'une approche paramétrique et algorithmique de la conception

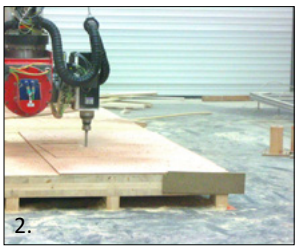
3. OBJECTIFS ET MÉTHODE

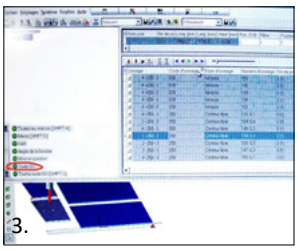
a. Objectifs

Ce stage a pour but de venir en aide à Oskar Gámez, doctorant au MAP-CRAI, dans son travail de thèse sur la création d'un outil de modélisation paramétrique d'aide à la conception et à la fabrication de parois cellulaires non-standards en bois.

Cet outil a pour objectif d'aider les architectes à concevoir des murs, parois ou enveloppes à ossature bois dont l'architecture est non-standard, c'est à dire dont la forme, le découpage et les différents éléments qui la constitue peuvent être courbes, irréguliers, et complexes. La liberté dans la géométrie de ces parois permet d'imaginer des usages multiples, allant de la construction neuve à des projets de réhabilitations qui doivent s'adapter à des bâtiments existants dont les façades peuvent être complexes ou avoir été déformées avec le temps. Cet outil doit aider le concepteur à modéliser en trois dimensions ces projets avec la plus grande précision possible et permettre des modifications rapides, afin de faciliter et de stimuler la conception architecturale en la représentant jusque dans sa dimension constructive le plus fidèlement possible (fig. 1). Cette modélisation précise permet ensuite la fabrication du projet par des outils à commande numérique de type CNC (fig. 2). Le logiciel de modélisation algorithmique choisi pour concevoir un outil qui répond à ces contraintes est Grasshopper®, le plugin gratuit de Rhinocéros®. Facile d'accès grâce à la programmation visuelle, il est le plus répandu dans le domaine de l'architecture dite paramétrique et bénéficie d'une communauté importante et de nombreux plugins.


1.


2.


3.

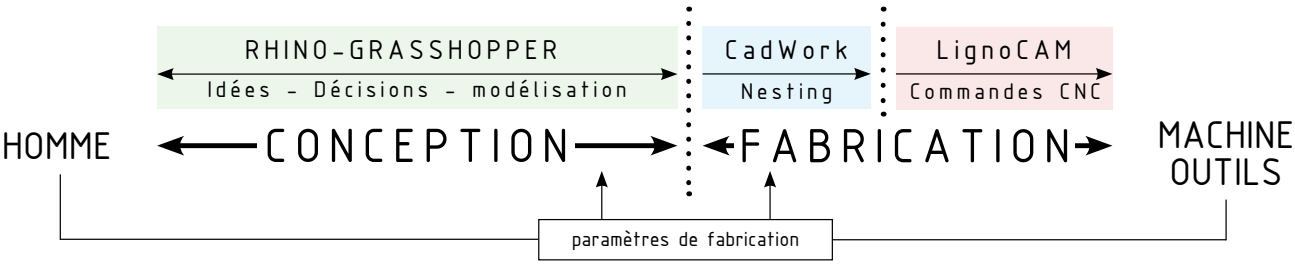

4.

1. Modélisation paramétrique du projet. 2. Machine à commande numérique (CNC). 3. LignoCAM, logiciel de pilotage CNC. 4. Assemblage des cellules.

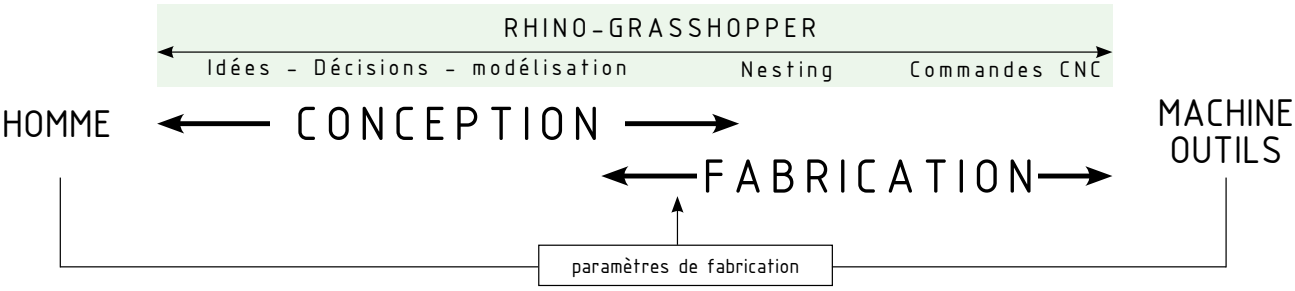
Le travail développé par Oskar Gámez lui a permis de démontrer le principe et la faisabilité de son outil. Il a notamment pu concevoir un projet et aller jusqu'à sa fabrication dans le cadre des Défis du bois 2014 (fig.4 et 5). Cette expérience a permis de faire émerger plusieurs problèmes lors de la conception et de la fabrication. Ainsi, le programme d'aide à la conception est à améliorer afin qu'il génère moins d'erreurs, qu'il soit plus rapide et qu'il propose plus de paramètres et de fonctions, notamment une fonction qui permettrait de générer une série de cellules les unes à la suite des autres qui jusqu'à présent devaient être construites individuellement par l'utilisateur. Lors de la fabrication, Oskar a rencontré de nombreux problèmes d'interopérabilité entre les fichiers enregistrés par son outil d'aide à la conception développé sur Rhinocéros® + Grasshopper® et le logiciel de préparation à la fabrication CadWork®, puis par la suite entre CadWork® et le logiciel LignoCAM® qui permet de piloter la machine à commande numérique utilisée. Ces étapes ont été plus longues et complexes que prévu car il était parfois nécessaire de redessiner une partie de la géométrie dans CadWork® et de saisir manuellement des lignes commandes à destination de la CNC (fig. 3).

5. Réalisation d'un prototype dans le cadre des défis du bois 2014 par Oskar Gámez.

Une des premières idées d'amélioration de cet outil est de supprimer tout autre logiciel du Workflow afin d'éviter les problèmes d'interopérabilités identifiés (fig. 6). Ainsi, le couple Rhinocéros® + Grasshopper® devient un outil complet d'aide à la conception jusqu'à la fabrication qui permet un continuum numérique (fig. 7). Cette double compétence offre une liberté supplémentaire et stimulante au concepteur en simplifiant ses allers-retours entre idées conceptuelles et aspects constructifs précis.



6. Workflow mis en place et expérimenté par Oskar Gámez avec en pointillés les problèmes d'interopérabilités rencontrés.



7. Objectif de Workflow : le continuum numérique.

b. Méthode

Pour aller le plus loin possible malgré un temps du stage très court, nous avons choisi de capitaliser au maximum notre temps de travail en nous focalisant sur les points qui nous ont paru les plus significatifs, et en nous appuyant sur plusieurs travaux : les algorithmes et le processus réalisés et imaginé par Oskar, des plugins qui pourraient répondre au mieux à nos besoins, ainsi que notre travail de PFE mené en parallèle dont la thématique et la stratégie de modélisation sont très proches. Nous avons également fait le choix de commencer directement le travail de modélisation sans établir au préalable un état de l'art sur le sujet des parois cellulaires non-standards en bois.

Les principaux objectifs sont :

- l'amélioration du travail existant dans son efficacité et son usage
- la résolution des problèmes d'interopérabilité par la conception de nouveaux outils dans Grasshopper qui permettent de se passer de logiciels extérieurs, et ainsi d'établir un continuum numérique
- de faire tendre le programme vers une forme de plugin pour Grasshopper

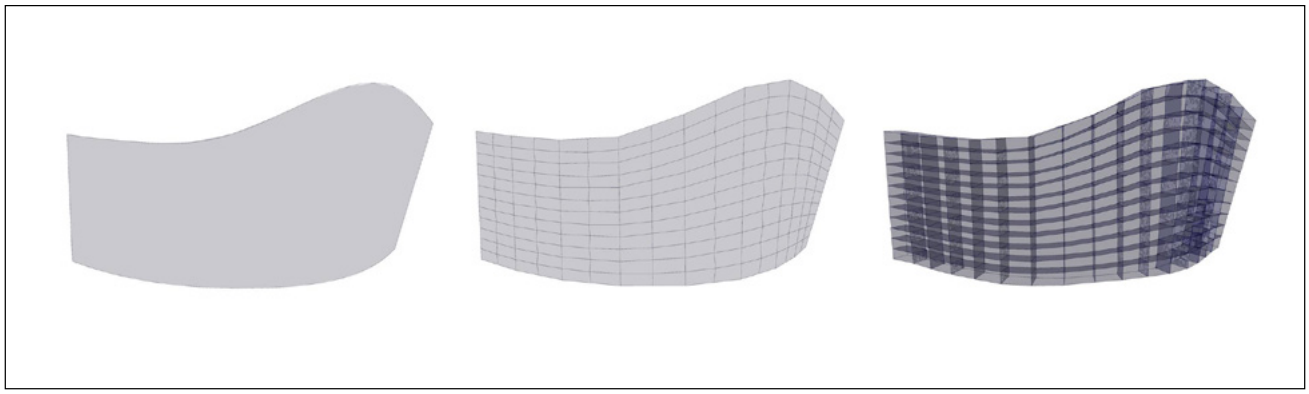
Pour y parvenir, nous avons décidé de suivre la stratégie de modélisation établie par Oskar en modifiant certaines étapes lorsque cela était nécessaire et en ajoutant un algorithme permettant la préparation et la simulation de l'usinage CNC, ce qui avait déjà été envisagé par Oskar. On retrouve ainsi trois parties, la première à l'échelle de la paroi, une seconde à l'échelle de la cellule et la dernière à l'échelle des éléments qui composent les cellules : les faces et leur fabrication. Ces trois étapes s'intitulent :

1. - Tessellation de surfaces
2. - Modélisation paramétrique d'assemblages en bois
3. - Préparation à la fabrication numérique

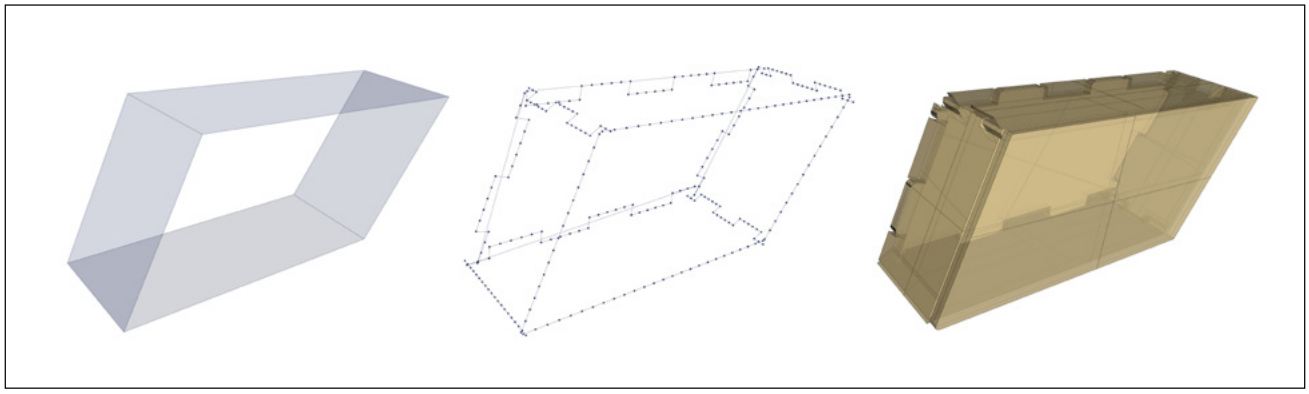
Nous avons choisi de traiter le problème de manière transversale afin de tirer parti de ce stage double pour pouvoir aborder un travail sur les différentes étapes et pouvoir démontrer une faisabilité du continuum numérique du début à la fin, plutôt que de nous focaliser uniquement sur une étape ou un outil spécifique. Les deux premières étapes ont été les plus approfondies car le travail avait été amorcé par Oskar, la dernière étape constitue quant à elle une piste de réflexion.

Pour des questions pratiques nous nous sommes chacun chargé d'une étape, en prêtant néanmoins attention à ce qu'elles soient parfaitement compatibles entre elles le but étant la création de différents composants d'un outil homogène. Thomas s'est chargé de la Tessellation de surfaces et Guillaume s'est occupé de la Modélisation paramétrique d'assemblages en bois, ainsi que de la Préparation à la fabrication numérique.

1 Tessellation de surfaces



2 Modélisation paramétrique d'assemblages en bois



3 Préparation à la fabrication numérique

II. TESSELLATION DE SURFACE

1. QU'EST CE QU'UN PATTERN ?

a. Approche générale

Avant toute chose, il est préférable de définir le terme de «pattern», terme clé de cet outil de conception. Une définition approfondie a été effectuée par Oskar dans son travail de thèse, il convient ici de simplement définir les idées clés de ce terme afin de mieux comprendre la manière de concevoir ce mur.

«Le mot anglais « pattern » est souvent utilisé pour désigner un modèle, une structure, un motif, un type, etc. Il s'agit souvent d'un phénomène ou d'une organisation que l'on peut observer de façon répétée lors de l'étude de certains sujets, auquel il peut conférer des propriétés caractéristiques. […] Un pattern constitue donc une solution générique à un type de problème fréquemment rencontré, en décrivant et formalisant les concepts sous-jacents à cette solution.»

Wikipedia - https://fr.wikipedia.org/wiki/Pattern (Page consultée le 28/07/2016)

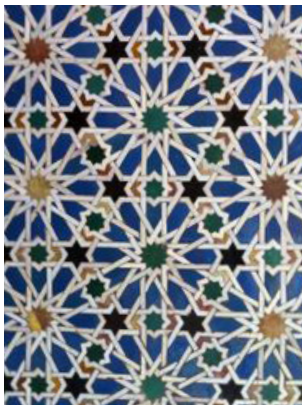
Un pattern est un terme difficilement qualifiable car il peut être utilisé dans de nombreux domaines. Cependant, sa principale propriété est qu'il ne peut être un élément isolé, il est toujours en lien avec un contexte et pris dans un ensemble qui peut être limité ou non. En graphisme, un pattern désigne un motif graphique susceptible d'être répété, comme un pavage, un carrelage, etc. On parle alors le plus souvent de pattern géométrique.

Une dimension symbolique est également présente dans un pattern, notamment dans le «pattern géométrique». En effet, cette propriété est très visible dans la culture grecque, musulmane ou romaine par exemple (fig. 1, 2 et 3). Chaque culture possède ses propres codes graphiques, construction centrée ou répétée, rectiligne ou arrondie, etc. Le pattern devient alors également social, culturel, politique, et humain.

En architecture, il existe deux types de pattern : un fonctionnel, en lien avec les activités qu'un bâtiment peut recevoir; Et un géométrique, appliqué à la forme d'un bâtiment, sa peau ou son squelette. Ces patterns géométriques appliqués au champ de l'architecture sont le plus souvent inspirés par la biologie et les études de

1. mosaïque grecque (musée de Delphes)

2. mosaïque romaine à Carmona (Séville)



3. mosaïque de l'Alcazar (Séville)

morphogénétique. En architecture, l'intérêt est surtout porté sur les processus de création par mimétisme des formes de la nature dans un contexte précis, et en particulier celui des organismes végétaux dans leurs propres milieux. Aujourd'hui, ces processus peuvent être simulés grâce à la modélisation paramétrique.

Pour cet outil, nous allons restreindre le terme de pattern à sa dimension géométrique appliquée à une architecture en tant que dispositif par lequel la forme et la structure peuvent être déclinés.

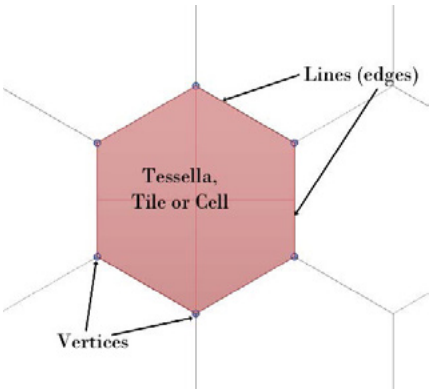
b. Construction d'un pattern

Afin de bien comprendre l'approche d'Oskar Gámez concernant la tessellation d'une surface par rapport à un pattern, quelques notions doivent être expliquées.

Comme nous l'avons vu précédemment, un pattern est un motif décoratif créé à partir d'une forme (shape) susceptible d'être répétée à un intervalle régulier. Ce terme se rapproche ainsi grandement d'une tessellation d'un point de vue géométrique. En effet, une tessellation est un ensemble de tesselles (également appelé tessella, tile ou monad en anglais), unité qui vient diviser une surface ou un plan. En d'autres termes, une tesselle est une région encadrée par des segments adjacents, eux même dérivés d'un système de points interconnectés (fig. 1). Ainsi une tessellation recouvre la totalité d'une surface et il n'est permis ni écarts, ni recouvrements entre tesselles.

Ce réseau de points découle d'une grille (Grid en anglais). Cette dernière est créée à partir d'intervalles donnés suivant les coordonnées U et V d'une surface NURBS. La grille est donc orientée en fonction du système de coordonnées d'une surface NURBS. Un réseau de lignes interconnectées (Lattice en anglais) peut ensuite être formé à partir de cette grille de points. Ce réseau n'est pas forcément orthogonal, il peut être triangulaire, hexagonal, etc.

Ainsi un pattern peut être défini comme une région créée à partir d'une grille de points, basée sur les coordonnées d'une surface. Cette région forme un motif, qui, une fois répété sur l'ensemble de la surface forme une tessellation.



1. Composition d'une tesselle (Thèse d'Oskar Gámez)

c. Types de Tessellations

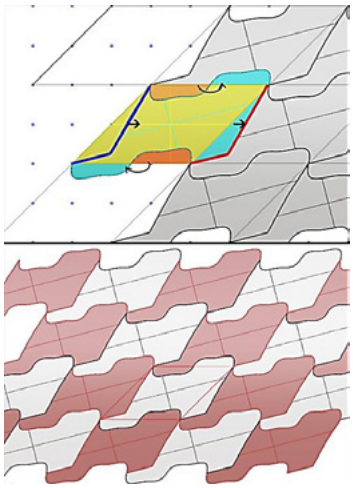
Les différentes catégories de Tessellations expliquées ci-dessus proviennent des travaux de M. C. Escher, R. Penrose et R. Serrentino, et sont expliqués dans le travail de recherche d'Oskar Gámez. Le but ici est simplement de comprendre certaines variantes de tessellations et leurs complexités afin de mieux comprendre l'outil proposé.

- Une tessellation régulière découle d'une seule forme répétée, une seule tesselle qui va devenir un proto-tessella ou prototile. La tessellation est alors dite «monohédral». Cependant, un proto-tessella peut être simple, c'est à dire composée d'une seule forme simple, ou composite c'est à dire composée d'un ensemble de formes créé à partir de rotations, symétries ou translations d'une même forme.

- Une tessellation semi-régulière quant à elle découle de deux ou trois tesselles formant un ensemble qui peut être répété. Un ensemble de deux tesselles est appelé tessellation «bihedral», de trois tesselles est appelé «trihedral», etc. Cependant, chaque ensemble doit pouvoir être répété sans créer de vides ou de chevauchements lors de la répétition du motif. Une tessellation semi-régulière est ainsi plus compliquée à concevoir.

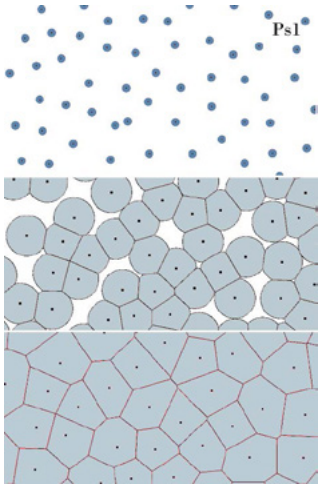
- Ces tessellations peuvent également être composées de formes plus complexes créées à partir de déformations d'une forme simple par des opérations s'appliquant en miroir (fig. 2), ceci ayant pour but de créer un pattern répétable sans vides ni chevauchements. Un ensemble de formes complexes peut créer des pattern plus recherchés comme par exemple celui de la façade de la philharmonie de Paris de Jean Nouvel qui est composé de 7 tesselles (fig. 3).

- Une dernière catégorie est la tessellation non-homogène. C'est le cas par exemple du Voronoi qui est construit à partir d'un nuage de points, et dont la construction des cellules est basée sur un diagramme mathématique qui défini les relations entre les points (fig. 4). D'autres patterns s'inspirant de la manière dont les formes sont créées dans la nature, liés aux études de biomimétisme, peuvent également être générées à l'aide de la modélisation paramétrique.



2. Exemple de déformation d'une forme simple par opérations en miroir (Thèse d'Oskar Gámez)

3. Zoom sur la façade de la philharmonie de Paris construit par l'architecte Jean Nouvel en Janvier 2015



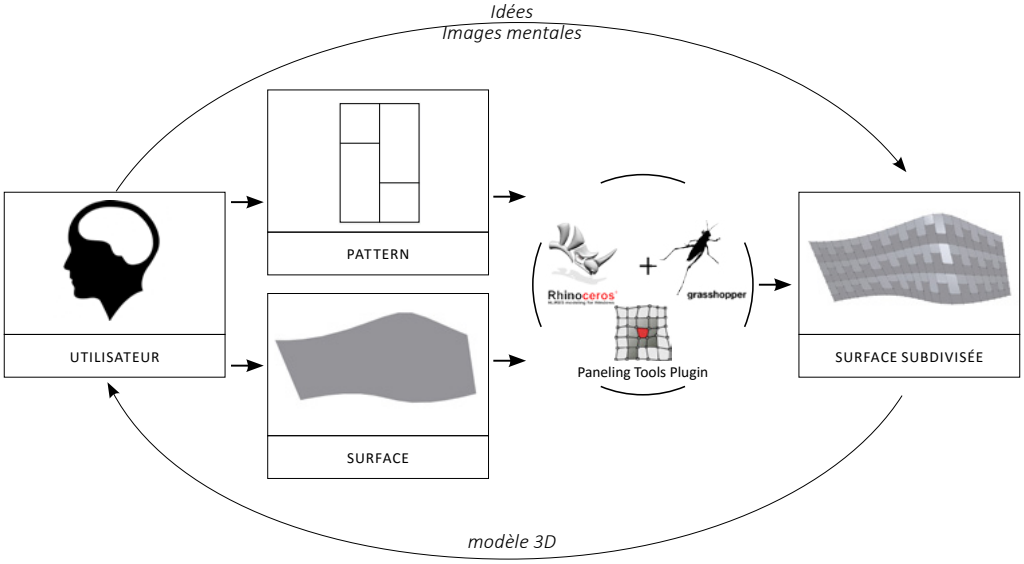
4. Formation du pattern Voronoi
(Thèse d'Oskar Gámez)

2. MÉTHODE D'OSKAR

a. Approche générale

Dans cet outil d'aide à la conception et à la fabrication de parois en ossature bois non-standard, la partie tessellation est une phase clé de la conception. En effet, c'est à ce moment que l'utilisateur dessine, imagine et décide des choix esthétiques et structurels de la paroi. Le but de cet outil est de permettre à l'utilisateur de dessiner ou modifier un pattern et de voir le résultat de sur la surface en temps réel.

La modélisation paramétrique joue ici un rôle très important puisqu'elle permet à l'utilisateur d'obtenir très rapidement une représentation de son idée, puis de la modifier en changeant certains paramètres, jusqu'à trouver le résultat optimal. L'idée est alors d'utiliser au maximum les avantages de la modélisation paramétrique et ainsi faciliter les allers-retours entre idées du concepteur et modélisation 3D (fig. 1).



1. Schéma de fonctionnement de l'outil de tessellation d'une surface

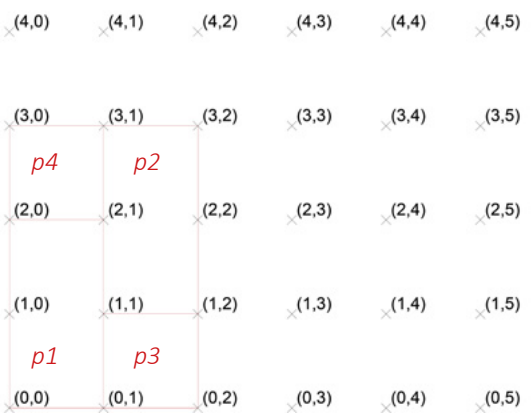
Dans la méthode d'Oskar, l'utilisateur dessine dans Rhinocéros un pattern en reliant plusieurs points d'une grille construite au préalable (1). Ce pattern peut être régulier (monohédral), mais également plus complexe (bi-, tri-, voir n-hédral) car l'utilisateur peut dessiner un ensemble de formes à l'aide d'un grille. Il doit ensuite regarder si, en se répétant, les différents éléments du pattern ne se chevauchent pas ou ne créent aucun vide entre eux.

Une fois dessiné, il peut exporter ce pattern à l'aide du plugin Paneling Tools de Rhinocéros. On obtient ainsi un fichier texte (.txt) composé du nom du pattern (2), du déplacement nécessaire afin d'être répété, puis d'une liste de coordonnées de points qui représente les différents points reliés par l'utilisateur ainsi que l'ordre dans lequel ils sont reliés.

Une fois le pattern imaginé, l'utilisateur peut ensuite importer sa surface NURBS (3), dans Rhinocéros-Grasshopper et y appliquer une grille (4) à l'aide du même plugin Paneling Tools pour Grasshopper. Cette grille peut être créée en fonction d'une division dépendant des coordonnées U et V de cette surface. La grille est le repère du futur pattern, elle permet de gérer sa densité et ainsi les dimensions des futures cellules.

Il ne reste plus qu'à appliquer le pattern sur la surface à l'aide de la grille. Cette opération est possible grâce au plugin Paneling Tools de Grasshopper (5) et au fichier texte créé auparavant. En effet, l'utilisateur doit indiquer à Grasshopper les différentes informations contenues dans le fichier texte. Une fois ces données saisies et rentrées dans le cluster, le résultat est bien la surface subdivisée par le pattern (6).

1 DESSIN D'UN PATTERN À L'AIDE D'UNE GRILLE



2 FICHIER.TXT X

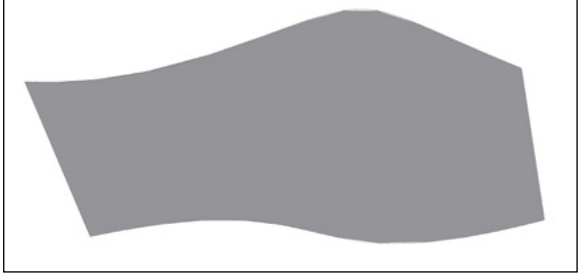
```
!-----------------------------------------------------------------
!-- Custom 2D patterns of PanelingTools plugin for Rhinoceros
!-- Recorded on Wednesday, July 22, 2016 at 14:41:22
!-- Pattern format:
!--    Name: String starting with a letter
!--    Shift: (x_shift,y_shift)
!--    Connections: (x0,y0)(x1,y1);(x2,y2)(x3,y3)(x4,y4);...
!-----------------------------------------------------------------

Oskar Pattern EFStagered_Vert
(2,3)
(0,0)(0,2)(1,2)(1,0)(0,0);          p1
(1,1)(2,1)(2,3)(1,3)(1,1);          p2
(1,0)(1,1)(2,1)(2,0)(1,0);          p3
(0,2)(1,2)(1,3)(0,3)(0,2)           p4
```

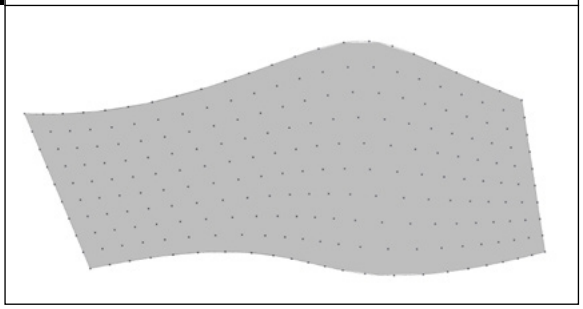
A

B

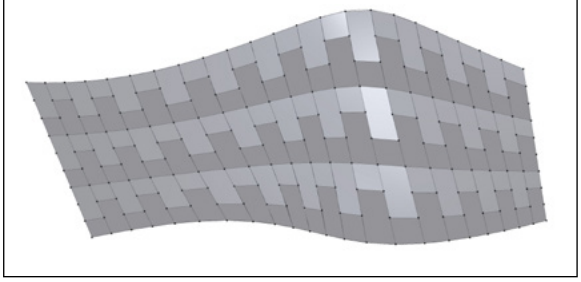
3 SURFACE NURBS IMPORTÉE



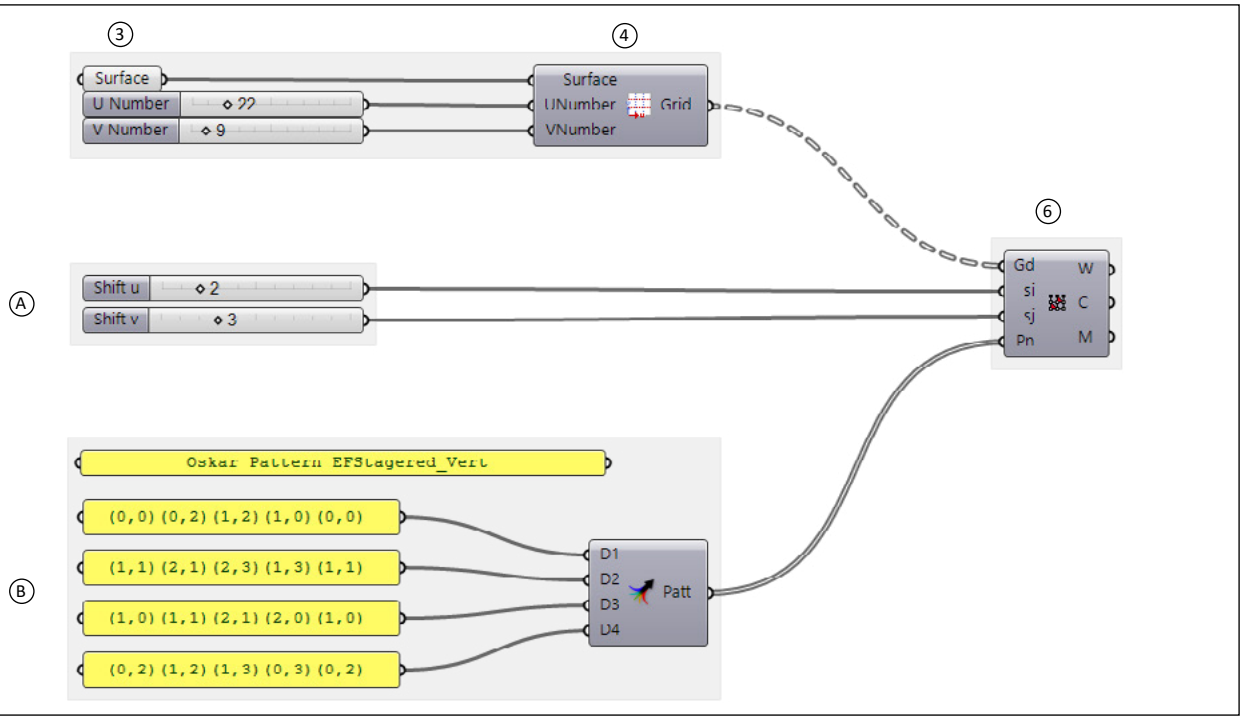
4 CRÉATION DE LA GRILLE DE POINTS



6 APPLICATION DU PATTERN SUR LA SURFACE



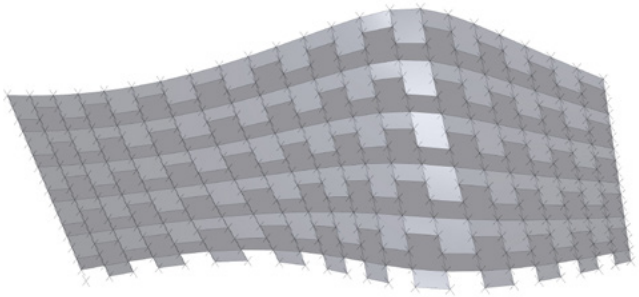
5



b. Limites de cette méthode et travail effectué

Cette méthode est plutôt efficace et rapide à mettre en œuvre. De plus, elle offre au concepteur une grande liberté d'expression architecturale. Cependant, on remarque vite un certain nombre de problèmes lorsque l'on commence à changer le nombre de valeurs en U et V de la grille de points.

En effet, lorsqu'il n'y a pas assez de points pour finir une forme du pattern sur la grille, cette dernière ne se referme pas avec les points restants et laisse ainsi un vide (fig. 1).

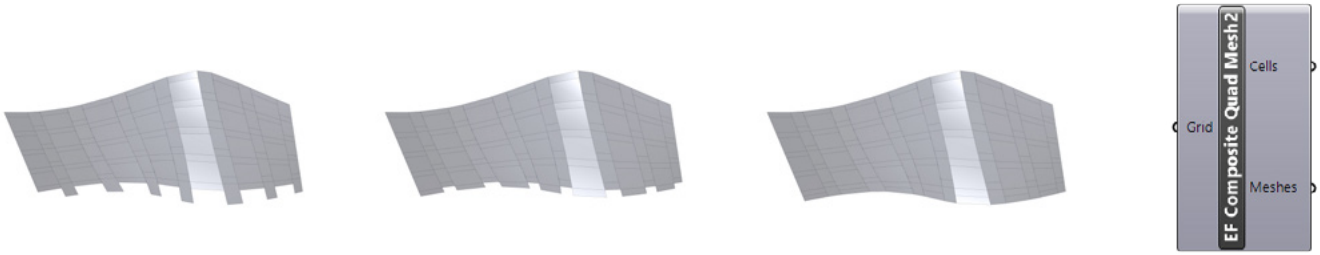


1. même pattern que dans l'exemple avec un paramètre U à 22 et un paramètre V à 16.

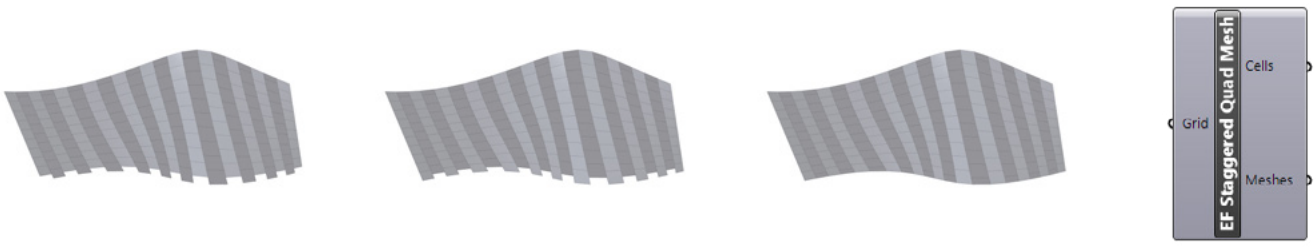
La seule solution que j'ai trouvé est de détecter les paramètres U et V qui sont à l'origine des erreurs et de recréer chaque surface manquante à l'aide d'une nouvelle liste de coordonnées. En somme, la solution est d'ajouter un ou plusieurs patterns spéciaux qui vont venir combler les «trous» qui apparaissent seulement dans certaines configurations des paramètres U et V (fig. 2 et 3).

Cette méthode marche plutôt bien malgré qu'elle soit très fastidieuse. Elle nécessite une bonne connaissance du logiciel et peut se révéler parfois très compliquée pour un utilisateur lambda, ce qui n'est pas le but recherché de cet outil. C'est pourquoi Oskar avait déjà commencé à proposer certains types de pattern sous la forme de clusters prêts à l'emploi.

Mon travail à donc tout d'abord consisté à résoudre les problèmes de chaque pattern proposé initialement par Oskar. Moins les points utilisés sur la grille sont nombreux, plus les erreurs dans les patterns sont difficiles à résoudre et demandent un plus grand nombre de nouveaux patterns pour combler les «trous».



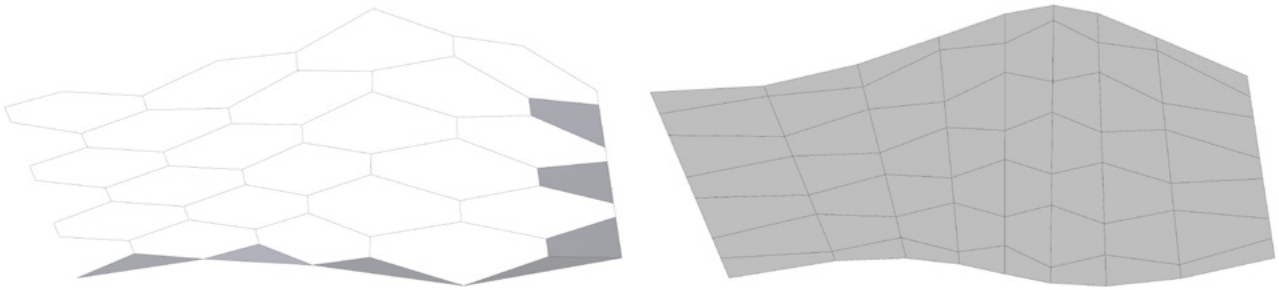
2. à gauche : même pattern dans 2 combinaisons de paramètres U et V différents, le troisième est l'algorithme modifié. A droite : Cluster modifié



3. à gauche : même pattern dans 2 combinaisons de paramètres U et V différents, le troisième est l'algorithme modifié. Adroite : Cluster modifié

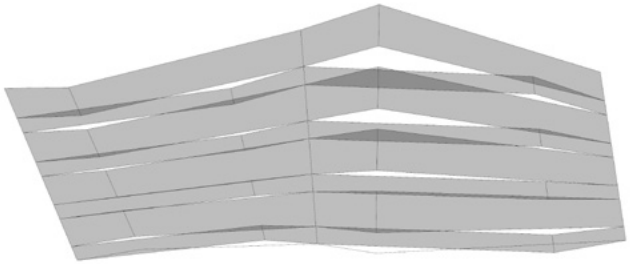
En regardant plus précisément le résultat obtenu par cette méthode, j'ai pu constater plusieurs problèmes :

En sortie de ce composant, trois résultats sont disponibles : une polyligne fermée (le contour du pattern), les segments qui la composent et un mesh (maillage en francais) composé de triangles ou de quads. Cette dernière contrainte limite énormément la créativité de l'utilisateur puisque les polygones de son pattern ne peuvent être définis que par un maximum de quatre points. Le pattern hexagonal proposé par Oskar a ainsi du être divisé en deux trapèzes (fig. 4).



4. à Gauche : Pattern Hexagonal initial, seul les triangles et les Quads sont crées . A droite : Pattern modifié : redécoupage en quads

De plus, ce plugin nous permet de créer des patterns qui, bien qu'ils soient complètement fermés à plat, créent des vides ou des chevauchements une fois appliqué à une surface (fig. 5). Ceci est du au fait que deux formes adjacentes ne partagent par forcément tous les points utilisés sur la grille. Ainsi deux segments adjacents colinéaires à plats ne le sont pas forcément en 3 dimensions. Ces petits problèmes semblent infimes à première vue, cependant si l'on agrandit le pattern en diminuant sa densité on se rend vite compte de l'importance du problème.



5. Chevauchement et vides liés au pattern initial et à une grille peu dense

En résumé, cette technique semble être à première vue optimale car elle permet une grande liberté de création pour l'utilisateur et une rapidité d'exécution. Cependant, on peut rapidement constater que les patterns sont limités à des quads et des triangles d'une part, et qu'ils ne fonctionnent pas pour toutes les valeurs U et V de la grille de points d'autre part. De plus, des erreurs géométriques liées à une méconnaissance de l'utilisateur sur les patterns peuvent vite entraîner des vides entre les cellules de la future paroi bien qu'ils ne soient pas visibles au premier abord. La solution d'Oskar amène donc à créer un catalogue de pattern prêt-à-l'emploi pour l'utilisateur. Le créateur de l'algorithme doit donc créer cette bibliothèque de patterns en ayant corrigé tous les problèmes envisageables.

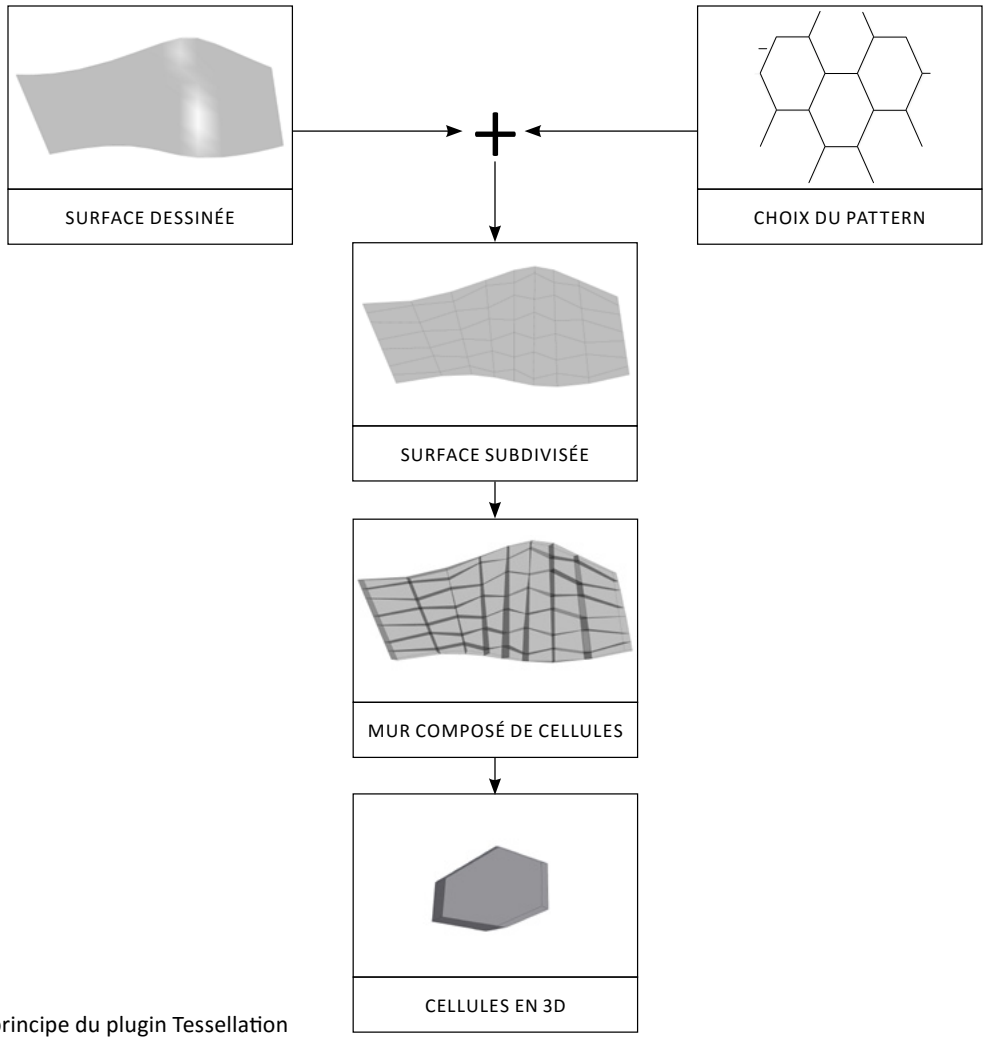
3. NOUVELLE MÉTHODE

a. Approche générale

Ce premier travail d'amélioration du travail effectué m'a demandé plus de temps que prévu, j'ai donc décidé de laisser cette partie en état pour essayer de compléter la partie tessellation afin d'obtenir une modélisation d'un mur utilisable pour Guillaume dans sa partie concernant les assemblages et l'usinage; Le but initial étant de créer un outil complet allant de la conception jusqu'à la fabrication.

J'ai donc utilisé les différentes étapes qu'Oskar avait imaginées en créant un plugin Grasshopper composé de différents clusters utilisables par le concepteur en fonction de ses besoins. Le but ici n'est pas de créer un plugin fini mais plutôt une ébauche d'un futur plugin plus complet fonctionnant dans un contexte plus large. Mon travail a été fait dans un but bien précis afin d'obtenir des éléments compatibles avec la suite du plugin créé par Guillaume. Les principales contraintes étant que les arêtes et les cadres des cellules soient des surfaces NURBS plates.

Très schématiquement, l'utilisateur part d'une surface, choisit un pattern et l'applique sur cette surface. Il peut ensuite extruder ce pattern pour créer le volume des cellules de la future paroi. Après quelques réajustements, chaque face de la paroi est bien plate et les cellules correctement listées peuvent être envoyées au plugin de Guillaume (fig. 1).



1. Schéma de principe du plugin Tessellation

b. Fonctionnement général

Ce plugin est composé de clusters divisés en six parties représentant les six étapes de la conception (fig. 1) de la paroi. Certaines parties sont elles-mêmes divisées en sous-parties. Ces différentes parties seront expliquées plus en détails par la suite.

1. vue d'ensemble du plugin et ses différents clusters

① Les clusters de la partie «Grid» vont permettre de générer une grille de points sur la surface NURBS utile pour certains types de patterns. Cette dernière est essentiellement composée de clusters créés par Oskar lors de son travail préalable.

② La partie «Pattern» est une bibliothèque de patterns. Elle comprend les clusters d'Oskar modifiés ainsi que des nouveaux créés à partir d'autres plugins. Une dernière sous-partie est composée d'outils permettant d'appliquer ou de modifier des patterns dessinés par l'utilisateur ou des patterns plus complexes.

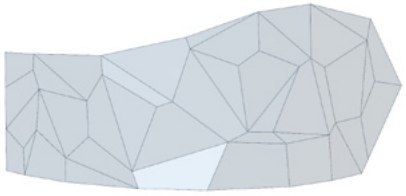
③ La partie «Volume» offre plusieurs choix d'extrusion du pattern.

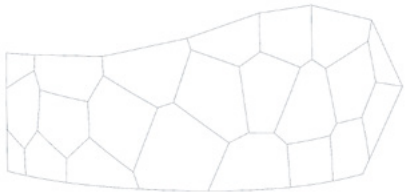
④ La partie «Planarization» est composée d'un unique cluster permettant de régler tous les problèmes liés à la complexité du logiciel et permet d'obtenir des surfaces NURBS adéquates pour la partie assemblage.

⑤ «HoleInWall» offre à l'utilisateur plusieurs choix de percements de la paroi afin de créer d'éventuelles baies ou différences de matériaux entre les fonds de caissons et ses cotés.

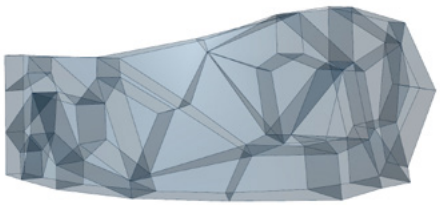
⑥ Enfin, «PatternAdjust» est une partie supplémentaire créée pour réajuster les cellules des patterns plus complexes possédant des motifs générés à partir de polygones plus complexes que des quads ou triangles (Un motif voronoï ou hexagonal par exemple).

②

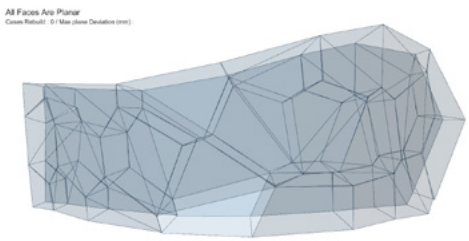




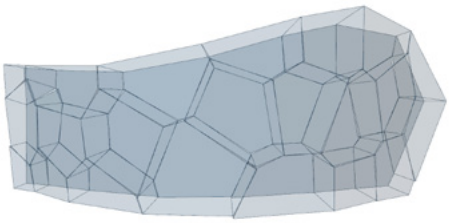
③



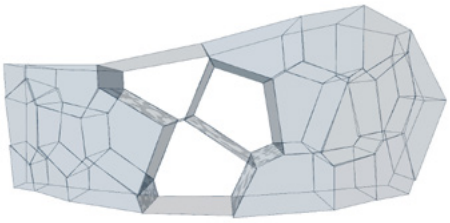
④



⑥

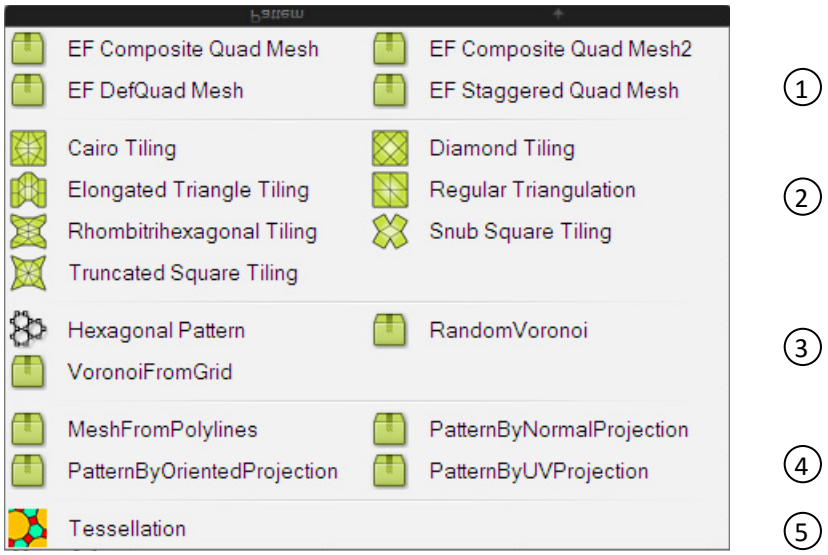


⑤



c. Patterns

La première grande étape de la partie «Tessellation» est la partie concernant le choix des patterns (fig. 1). Bien qu'il en existe une infinité, ce plugin a pour but de fournir un catalogue de patterns prêt à l'emploi. Cette partie est divisée en six sous-parties de patterns, chacun fonctionnant d'une manière différente.



1. Volet déroulant proposant tous les clusters de la partie «Pattern»

Comme nous l'avons vu précedemment il est très important d'unifier les paramètres d'entrée et de sortie de chaque cluster. L'objectif de cette partie est d'obtenir la tessellation d'une surface par un pattern. Le type de pattern est défini par le cluster choisi. Il nous faut pour ensuite des paramètres d'entrées et de sorties précises et similaires pour chaque cluster.

En entrée :

- une surface NURBS en tant que BREP, représentant l'envergure de la future paroi
- un paramètre gérant la densité du pattern. Cette dernière est définie soit par une grille de points générée au préalable, soit par des valeurs numériques définissant une division sur les coordonnées U et V.

En sortie :

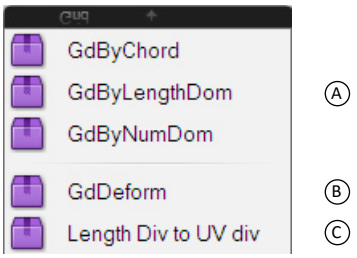
- Un Mesh ou un ensemble de Meshs[2] représentant la surface subdivisée par le pattern.

2 *Un mesh ou maillage est un objet tridimensionnel constitué de sommets, d'arêtes et de faces organisées en polygones qui sont uniquement des quads et des triangles. C'est une donnée obligatoire pour la partie «planarization» que nous verrons plus tard.*

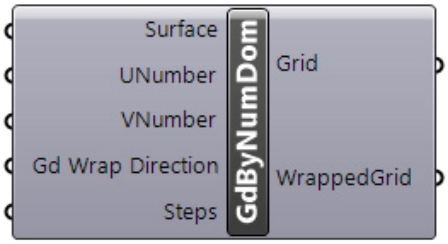
① *Patterns d'Oskar Gámez*

La première sous-partie est une bibliothèque de quelques patterns qu'Oskar avait commencé à produire. Cette liste n'est pas exhaustive et est amenée à être complétée lors d'un prochain travail. Ces clusters n'ont qu'une entrée qui est une grille de points préalablement générée. Une série de Meshs et de polylignes fermées sont disponibles en sortie classées et listées représentant chaque future cellule.

Pour générer la grille de points, plusieurs clusters sont disponibles dans la partie «Grid» (fig. 2).



2. Volet déroulant proposant tous les clusters de la partie «Grid»



3. Cluster «GdByNumDom» créé par Oskar Gámez

Les trois premiers clusters permettent de générer une grille :

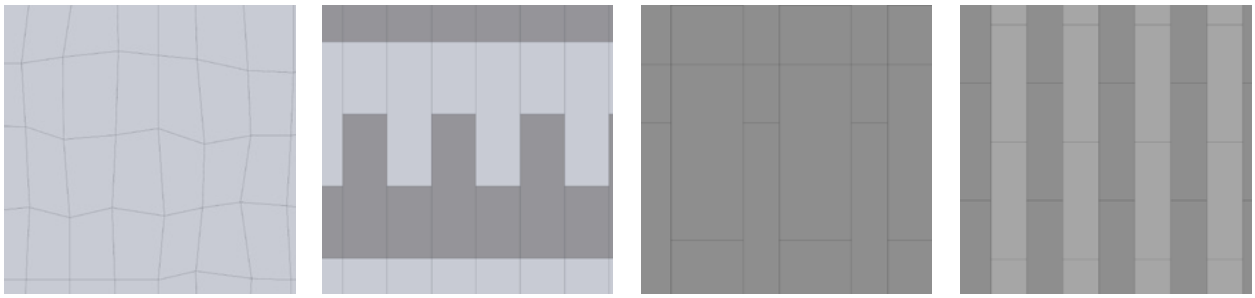
Ⓐ - GdByChord et GdByLengthDom génèrent cette grille par un découpage selon une distance, tandis que GdByNumDom (fig. 3) permet simplement de diviser la surface par un nombre sur les coordonnées UV.

Ⓑ - GdDeform est un cluster créé par Oskar afin de déformer la grille.

Ⓒ - LengthDivtoUVdiv permet de récupérer une division UV approximative en fonction de distances souhaitées sur et U et sur V.

La génération d'un pattern par une grille de points offre une grande variété (fig. 4). De plus, certaines options permettent de créer des variations par le décalage des points selon les coordonnées U et V.

Cependant, la stratégie de création d'une bibliothèque restreint énormément les patterns disponibles et nécessiterai un travail ultérieur d'approfondissement afin d'élargir les possibilités de l'outil.



4. EF DefQuad Mesh / EF CompositeQuad Mesh / EF CompositeQuad Mesh2 / EF StaggeredQuad Mesh

② *Patterns du plugin Mesh +*

En créant la bibliothèque de pattern à partir de Paneling Tools, j'ai décidé de me tourner vers la communauté Grasshopper3D pour chercher un plugin proposant des bibliothèques similaires déjà prêt à l'emploi. En comparant plusieurs plugins, j'ai découvert le plugin Mesh + qui propose plusieurs outils de modification, d'analyse et de création de meshs à partir de patterns.

Mesh + se détache des autres plugins par sa simplicité et son efficacité. De plus, c'est un outil spécialisé dans les meshs, ce qui est une des conditions principales de notre plugin. Chaque cluster contient quatre entrées minimum qui sont les paramètres de bases, puis des entrées optionnelles afin de créer des variations dans le pattern.

Paramètres d'entrées de base :

- Une surface NURBS en tant que BREP. Elle représente l'envergure de la future paroi.
- Deux valeurs numériques indépendantes pour gérer la densité du pattern sur les coordonnées U et V.
- Un Booléen «C» (Closed) permet de fermer le pattern en remplissant les vides sur les bords de la surface lorsque l'un des motifs n'est pas complet. Ce paramètre fonctionne de la même manière que que ce que nous avons essayé de mettre en place sur les patterns conçu par Oskar à l'aide de Paneling Tools.

Paramètres d'entrée spécifiques à certains clusters :

- Le booléen «O» (Options) permet de choisir les éléments du pattern que l'on souhaite conserver.
- Le booléen «T» (Evaluate), est surement l'une des options les plus intéressantes de ce plugin. Elle permet, par un paramètre variant de 0.0 à 1.0, de déformer le pattern pour obtenir une multitude de variantes (fig. 1).
- Les booléens SU (Shift U), SV (Shift V) et R (Shift) permettent de créer des décalages du pattern dans une ou toutes les directions, ce qui permet de créer d'autres variations dans le pattern.
- Le booléen F (Flip), permet d'inverser la composition du pattern, créant là aussi une autre variante de pattern.

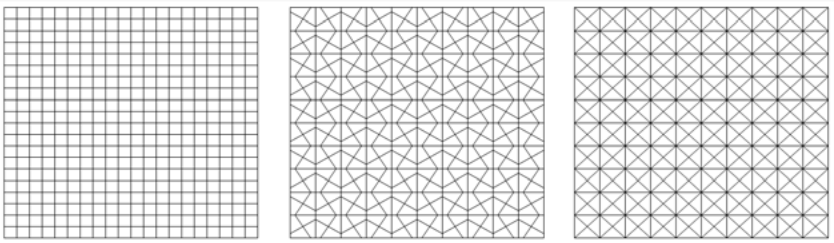
Paramètres de sorties de tous les clusters :

- «M» : un seul mesh composé de toutes les faces du pattern.
- «out» : ces clusters étant créés à partir d'un script VisualBasic, une sortie «out» est obligatoire sous Grasshopper et permet d'indiquer les éventuelles erreurs.

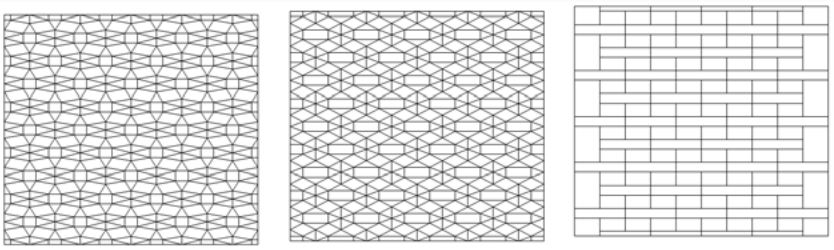
Ces clusters sont «ouverts» et nous permettent de comprendre la manière dont ils sont programmés et sont même modifiables. Ils permettent également de réduire considérablement le temps de calcul ainsi que le nombre de composants Grasshopper. Ces clusters permettent également de simplifier l'interface de l'utilisateur en supprimant les composants de la partie «Grid», tout en permettant des variations par décalage de la grille de points.

Cependant, l'unique manière de gérer la densité du pattern se fait par une division sur les paramètres U et V. C'est pourquoi j'ai créé par la suite le composant «LengthDiv to UVDiv» qui permet à l'utilisateur de convertir une distance en un nombre entier de divisions U et V.

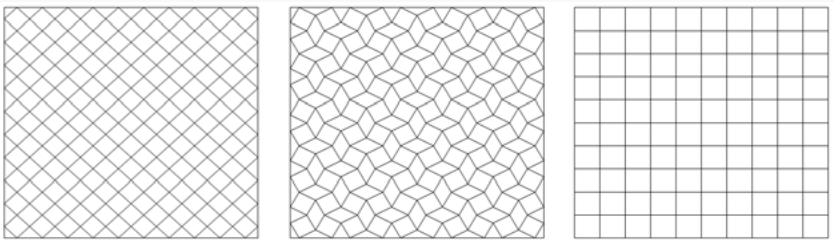
Cairo



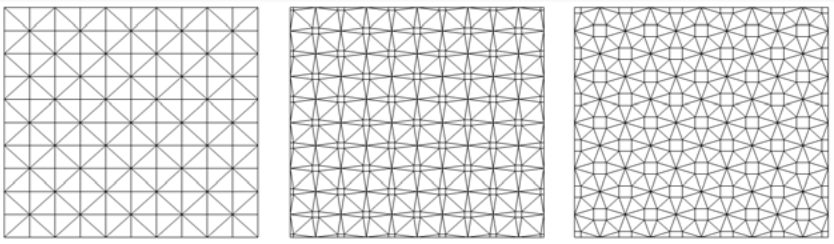
Rombitrihexagonal



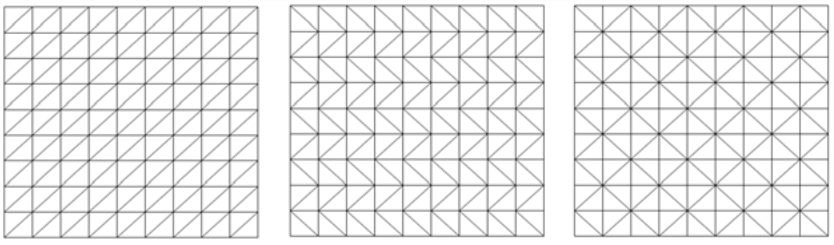
Snub Square



Truncated Square



Regular Triangulation

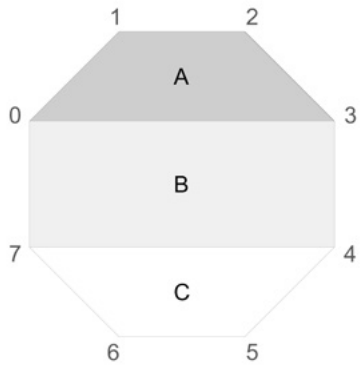


1. Liste de quelques clusters de la bibliothèque de Mesh + avec des paramètres «T» ou «O» différents.

③ *Patterns plus complexes*

Nous avons ainsi deux sous parties composées d'une bibliothèque de patterns prêts à l'emploi. Cependant, tous ces patterns se ressemblent plus ou moins car ils sont composés uniquement de triangles et de quads, une caractéristique obligatoire pour créer un mesh valide.

Pour pallier à ce problème et créer des patterns possédant des polygones plus complexes, un cluster a été créé afin de subdiviser des patterns possédant plus de quatre cotés en triangles et quads. Ce cluster, «MeshFromPolyline[2]», a besoin du contour des cellules sous forme de «Closed Polylines» pour subdiviser un pattern possédant plus que quatre cotés. Le cluster découpe ces polylines en fonction de leurs vertices suivant la figure ci-dessous (fig. 1). Les quads sont privilégiés afin d'obtenir un minimum de faces supplémentaires. «MeshFromPolyline» ne supporte que des polylines qui possèdent au maximum 12 cotés mais pourrait facilement être amélioré à l'aide d'une boucle ou d'un script en VB. Les polylignes initiales sont également reconstruites et disponibles en sortie pour une partie suivante qui permet de reconstruire le pattern d'origine à l'aide de ces polylignes.



1. Schéma de fonctionnement du cluster MeshfromPolyline

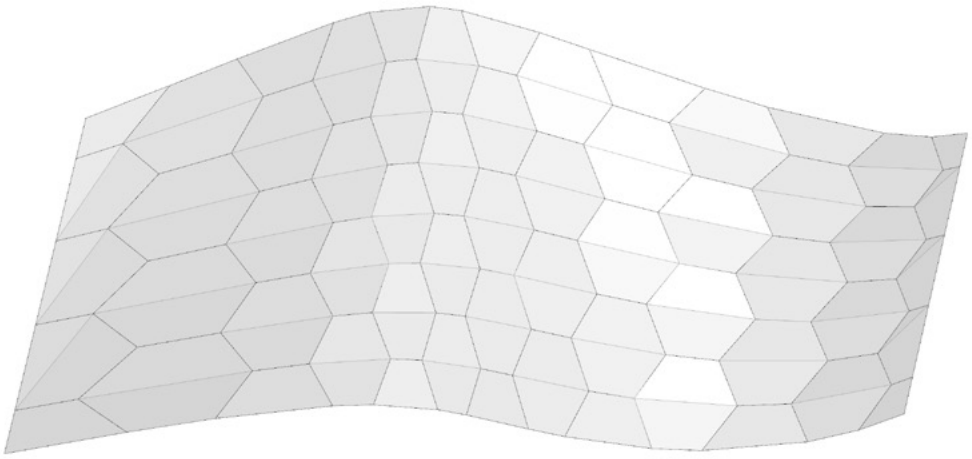
Grâce à ce cluster nous pouvons ainsi générer des patterns plus complexes, par exemple voronoi ou hexagonal. Dans cette sous-partie, trois types de patterns sont proposés à l'utilisateur.

- Un pattern hexagonal (fig. 2) qui utilise le plugin Vipers permet une tessellation à partir de paramètres U et V et d'une surface NURBS. Cependant, seules des courbes ou des vertices sont disponibles en sortie de ce composant. Un cluster regroupant ce composant ainsi que «MeshFromPolyline» permet à l'utilisateur de créer un pattern hexagonal à partir d'une seule fonction.

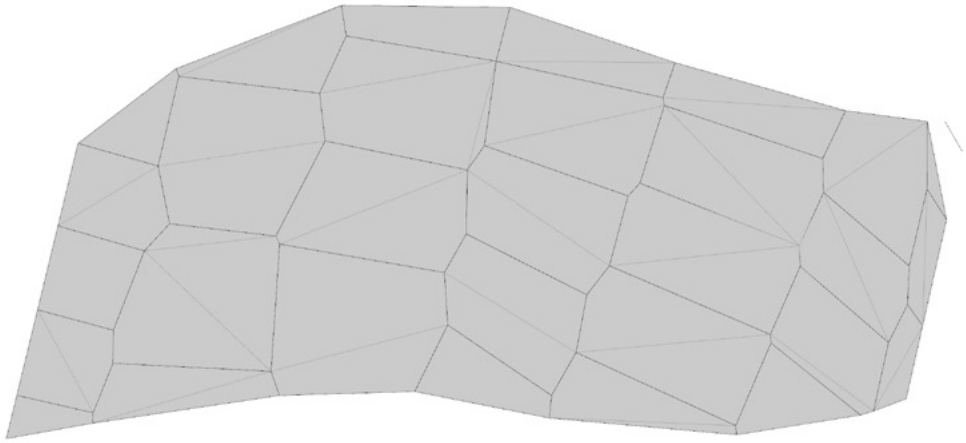
- Un pattern voronoi (fig. 3) avait déjà été créé par Oskar Gámez à partir d'une grille de point et d'une surface. Ce pattern est généré par la soustraction des points d'une grille. La soustraction peut se faire par une distance maximale entre deux points et un pourcentage permet de réduire le nombre de points de manière «aléatoire». De même, ce cluster est couplé au MeshFromPolyline pour n'avoir qu'un seul composant.

- Cette dernière méthode étant un peu «rigide» pour un Voronoi (fig. 4), un deuxième cluster permet de générer un motif plus aléatoire créé à partir de points positionnés au hasard sur la surface. Toutefois, une taille minimale de cellule est paramétrable en entrée.

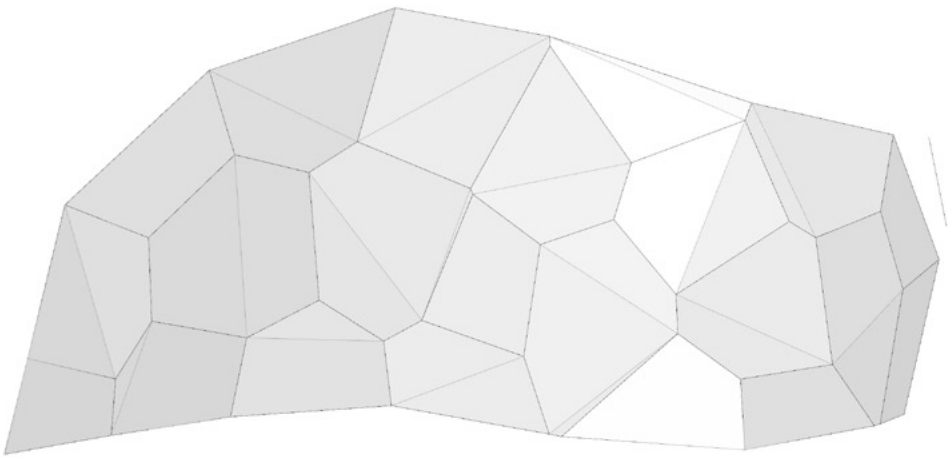
2 Ce cluster est présent dans la quatrième sous-partie de la catégorie «Pattern»



2. Application du cluster Hexgonal Pattern



3. Application du cluster VoronoiFromGrid



4. Application du cluster RandomVoronoi

4 *Tessellations par patterns dessinés*

Cette sous-partie est davantage une bibliothèque «d'outils» liés aux patterns. On peut par exemple y retrouver le cluster «MeshFromPolyline» vu précédemment. On peut également y trouver trois clusters permettant d'appliquer ou de projeter un pattern dessiné dans Rhinocéros[2] sur une surface NURBS.

Cette idée m'est venu lors de mon travail de projet de fin d'étude. Ce dernier, en lien direct avec celui de Guillaume, découle d'une problématique similaire. En effet, Guillaume et moi avons imaginé un système de couvertures non-standards utilisant la modélisation paramétrique. Cette couverture est composée de caissons structurels. Bien qu'une couverture et une paroi en structure cellulaire ne se comportent pas de la même manière, structurellement parlant, il existe beaucoup de liens dans la façon de concevoir ces deux éléments. C'est pourquoi de nombreux allers-retours entre ce travail de recherche et le travail de projet ont pu améliorer et/ou remettre en question certains aspects de ces travaux.

Dans le travail de PFE, le pattern devait suivre un dessin spécifique et des dimensions précises, c'est pourquoi il m'était plus judicieux de dessiner manuellement un plan de caissonnage et de le projeter sur la surface.

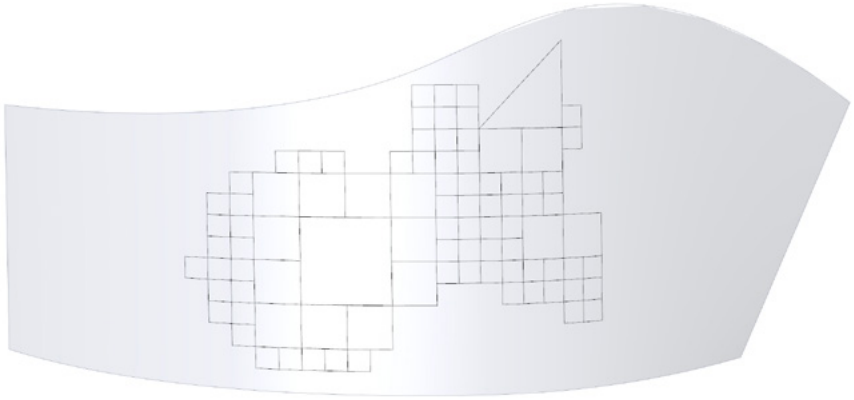
Trois manières de projeter un pattern dessiné sont proposées, dans chaque cas le pattern doit être créé à l'aide de polylignes fermés et coplanaires :

- «PatternNormalProjection» (fig. 1) projette simplement le pattern sur la surface en fonction de la normal du plan du pattern. C'est surement la manière la plus basique de projeter un dessin sur une surface mais elle permet de conserver les dimensions dans le plan du pattern. Cependant, il est très compliqué de remplir la totalité d'une surface courbe avec cette méthode. Cette fonction peut être utile pour créer une paroi à partir d'une élévation.

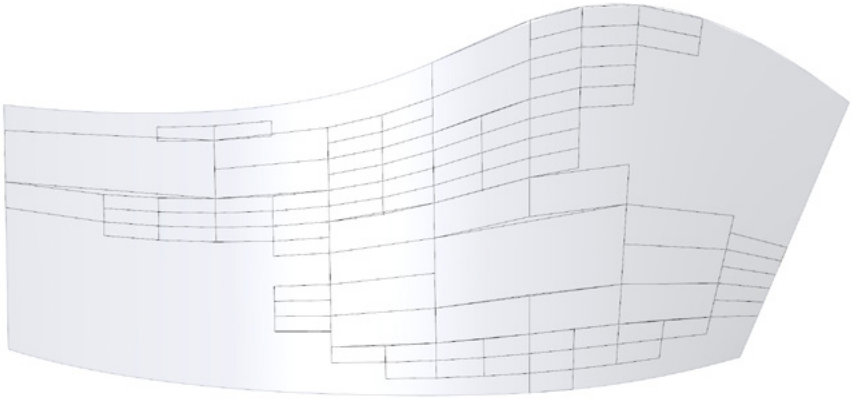
- «PatternByUVProjection» (fig. 2) applique le pattern sur la surface en fonction de coordonnées U et V. L'ensemble des vertices du pattern sont projetées sur une surface créée à partir de la BoundingBox du pattern. À partir de cette projection, des coordonnées UV sont créées pour chaque point du pattern. Il reste plus qu'à l'appliquer sur la surface initiale. Pour compléter entièrement la surface, il faut que le pattern couvre la totalité d'un rectangle ou d'un carré. Cette méthode déforme cependant le pattern initial.

- «PatternByOrientedProjection» (fig. 3) applique le pattern sur la surface en fonction d'une projection orientée[3] d'un plan initial du pattern sur le plan médian de cette surface. Une rotation autour du centre du pattern ainsi qu'une mise à l'échelle du pattern sont possibles par cette opération qui limite les déformations.

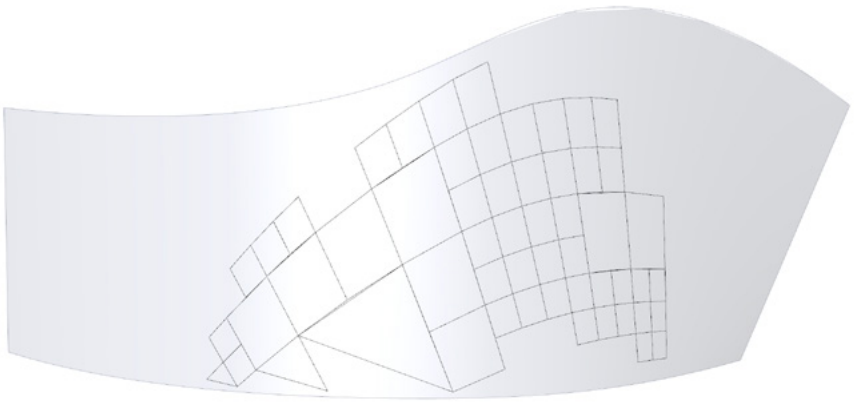
2 On peut également dessiner ce pattern dans un autre logiciel et l'importer dans Rhinocéros.
3 Composant «orient» de Grasshopper



1. Pattern By Normal Projection



2. Pattern By UV Projection



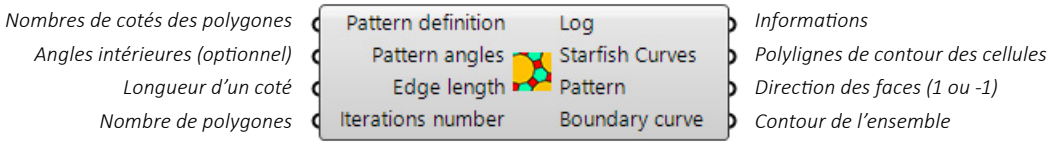
3. Pattern By Oriented Projection

5 Plugin StarFish

Cette dernière sous-partie n'a été développée qu'à la fin du stage lors de recherches plus théorique sur les patterns. En effet, c'est en faisant un état de l'art sur les patterns que je me suis rendu compte que le plugin proposé n'offrait qu'un choix très restreint de «familles» de patterns.

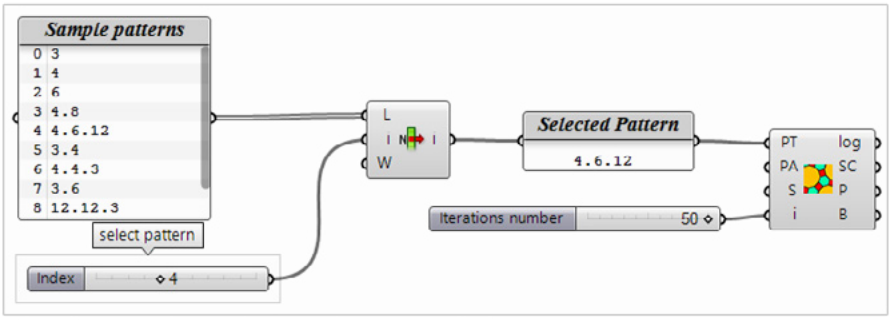
Les outils proposés dans la sous-partie précédente permettent d'appliquer des patterns créé initialement en plan. Le plugin StarFish[2] permet de générer des patterns en 2D et peut ainsi devenir un outil d'aide au dessin d'un pattern. Il permet de créer des patterns «monohedral» mais aussi des bi-, tri- voire «n-hedral».

Un pattern est construit à partir de diverses informations inscrites dans des «panels» :



1. Composant Tessellation du plugin StarFish

à partir de ce composants nous pouvons obtenir divers patterns[3] :

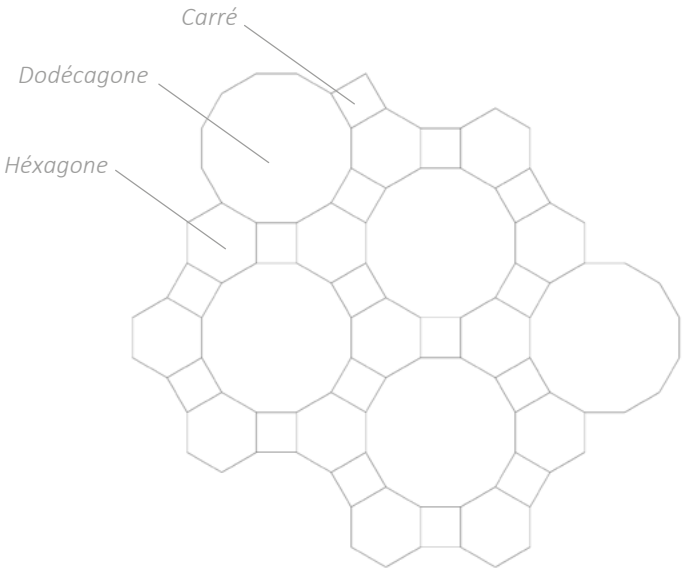


2. Extrait du fichier exemple créé par l'auteur expliquant la création d'un pattern regulier

Le panel de gauche sur la fig. 2 représente une liste de différents patterns proposées par l'auteur. Ici le pattern choisi est le n°4 donc «4.6.12». Ce pattern est donc constitué de polygones à 4 côtés, 6 côtés et 12 côtés. Pour écrire un pattern, chaque type de polygone est séparé par un point. Le nombre de polygones total est défini par l'utilisateur grâce au curseur i (itérations number) et le plugin gère ensuite automatiquement les différents types de polygones qui seront utilisés dans le pattern.

Si l'on ne met rien dans PA, chaque polygone est seulement défini par son nombre de côtés, il devient ainsi un polygone régulier. L'exemple de la fig. 2 est donc un pattern composé de carrés, d'hexagones et de dodécagones (12 côtés) (fig. 3). Pour des patterns composés de polygones non réguliers, il faut indiquer les angles souhaités.

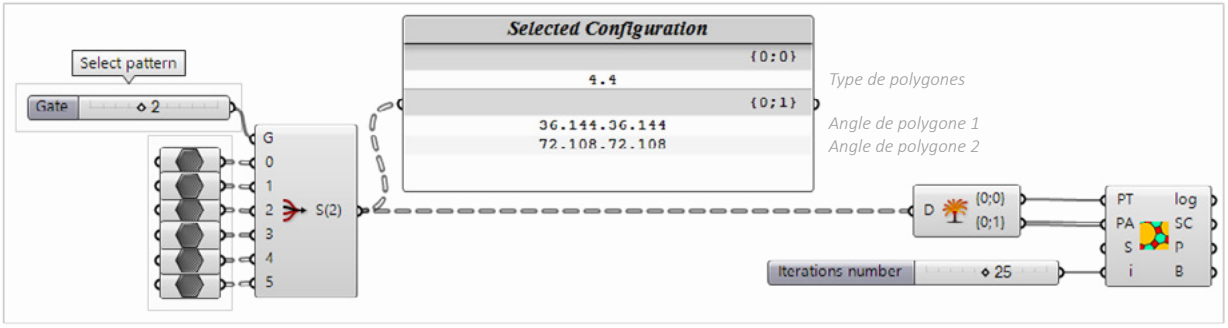
2 Plugin créé par Michael Weizmann disponible sur Food4Rhino
3 Ces deux images sont tirées du fichier Grasshopper exemple créé par Michael Weizmann



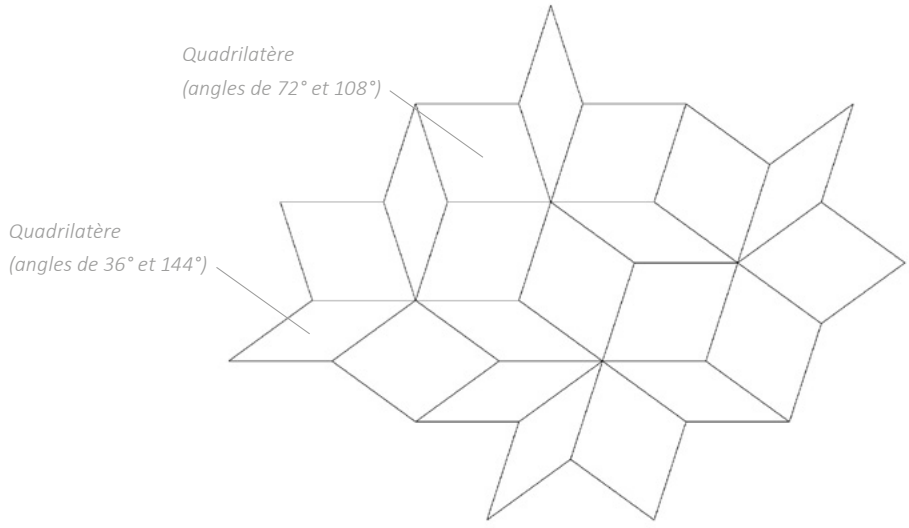
3. Extrait du résultat sur Rhincéros de l'exemple de la fig. 2

Le deuxième exemple (fig. 4 et 5) est composé de deux types de quadrilatères, le premier possédant des angles de 36° et 144°, le deuxième des angles de 72° et 108°.

Les différents angles d'un polygone sont séparés par un point. Il doit y avoir le même nombre de valeurs que de côtés et la somme des angles doit faire (n-2)x180°, n étant le nombre de cotés du polygone.



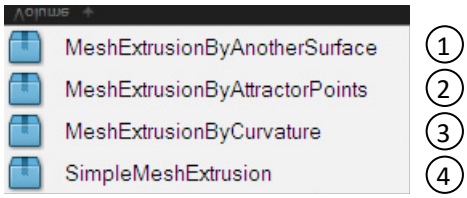
4. Extrait du fichier exemple créé par l'auteur expliquant la création d'un pattern non régulier



5. Extrait du résultat sur Rhincéros de l'exemple de la fig. 4

d. Volume

Une fois la surface subdivisée par le pattern choisi, l'utilisateur peut créer le volume du mur en extrudant le pattern. Pour ce faire, il dispose de quatre méthodes d'extrusion (fig. 1).



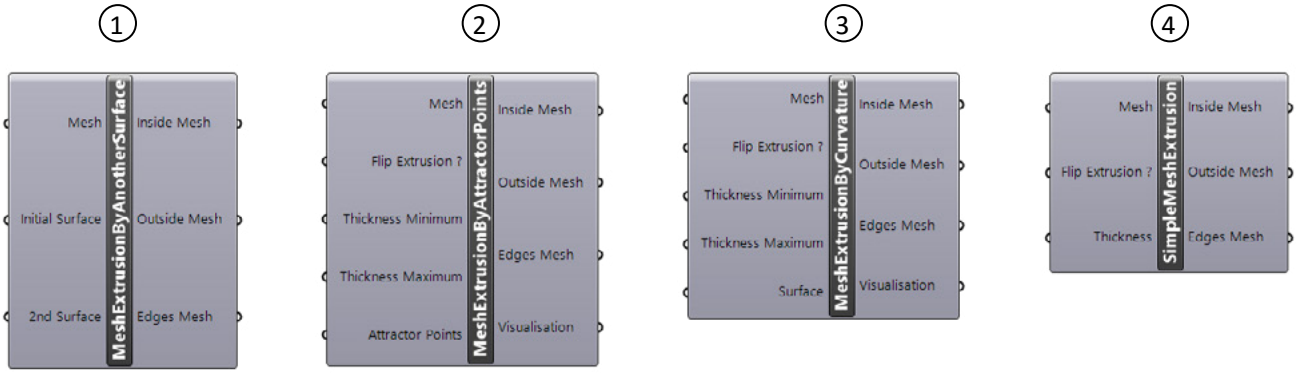
1. Volet déroulant proposant tous les clusters de la partie «Volume»

J'ai fait le choix d'utiliser des outils d'extrusion du plugin Mesh+ afin de ne pas changer de type de géométrie, l'étape suivante demandant également un mesh. J'ai pu comparer cette méthode d'extrusion de pattern avec celle établie en PFE basée sur une extrusion point par point en fonction de leur normale à la surface. Les composants du plugin Mesh+ sont bien plus rapides et procèdent de la même méthode puisqu'ils extrudent chaque point du mesh par rapport à leur normale, j'ai donc retenu le composant du plugin.

Ces quatre composants (fig. 2) ont besoin au minimum des paramètres d'entrées suivants :

- Un mesh ou ensemble de meshs correctement listé[2] représentant les futures cellules du mur.
- Une ou plusieurs épaisseurs (Min et Max) sous la forme de nombres correspondant à l'épaisseur souhaitée du futur mur.
- Un booléen pour inverser le sens d'extrusion, vers l'intérieur ou vers l'extérieur.

Chaque méthode permet d'obtenir trois différentes sorties : la face intérieure, la face extérieure et le cadre de chaque cellule, le tout en mesh. Chaque élément d'une cellule doit avoir le même identifiant ainsi que le même niveau de liste pour pouvoir reconstituer la cellule par la suite.

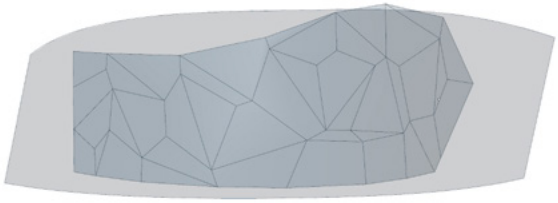


2. Les différents cluster proposés pour cette partie. On peut y voir leurs entrée et sorties

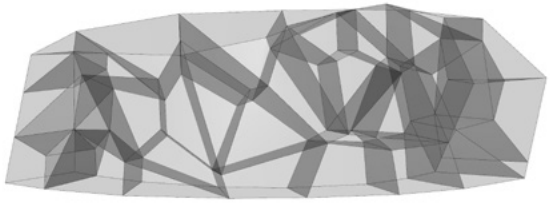
2 Le plugin Mesh+ créant un unique mesh composé de tous les polygones du pattern. Les composants d'Oskar ainsi que les patterns créés manuellement sont eux composés d'un ensemble de meshs représentant chaque cellule, une face pour des patterns simples, plusieurs correctement listés pour des patterns complexes.

Chaque méthode d'extrusion possède ensuite sa propre entrée spécifique :

① - «MeshExtrusionByAnotherSurface» nécessite la surface initiale ainsi qu'une seconde surface (fig. 3). Pour ce composant, il n'est pas question d'épaisseur minimum. Le pattern appliqué à la surface initiale est recopié sur la seconde en fonction des coordonnées UV. Ces deux patterns sont ensuite reliés pour créer des cellules (fig. 4). Cette méthode nécessite le bon sens de l'utilisateur afin d'indiquer une seconde surface compatible.

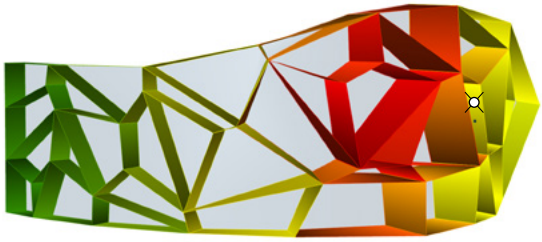


3. Pattern sur la surface initiale et seconde surface



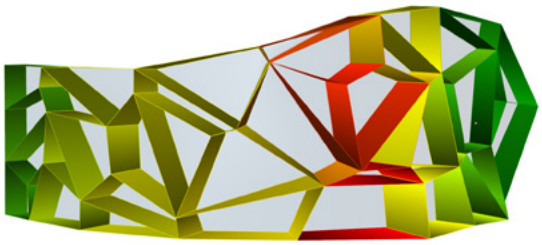
4. Volumétrie des cellules créées

② - «MeshExtrusionByAttractorPoints» permet de créer une épaisseur variable en fonction de la proximité avec un ou plusieurs points positionnés par l'utilisateur. Une sortie supplémentaire permet d'obtenir un panel de couleurs allant du vert au rouge permettant de visualiser la proximité des cellules avec ce ou ces points (et donc la variation d'épaisseur du mur)(fig. 5). En effet, plus une cellule est proche d'un point et plus l'épaisseur du mur à cet endroit sera grande.



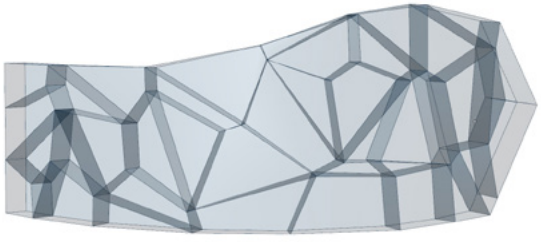
5. Visualisation de la variation d'épaisseur en fonction de la proximité avec un point

③ - «MeshExtrusionByCurvature» permet de générer une épaisseur variable en fonction de la courbure de la surface initiale. Il existe la même sortie que le composant précédent afin de visualiser graphiquement l'épaisseur du mur (fig. 6).



6. Visualisation de la variation d'épaisseur en fonction de la courbure de la surface NURBS

④ - «SimpleMeshExtrusion» est surement la manière la plus simple d'extrusion (fig. 7). Il suffit de rentrer l'épaisseur constante souhaitée.



7. Simple extrusion d'un pattern

e. Planarization

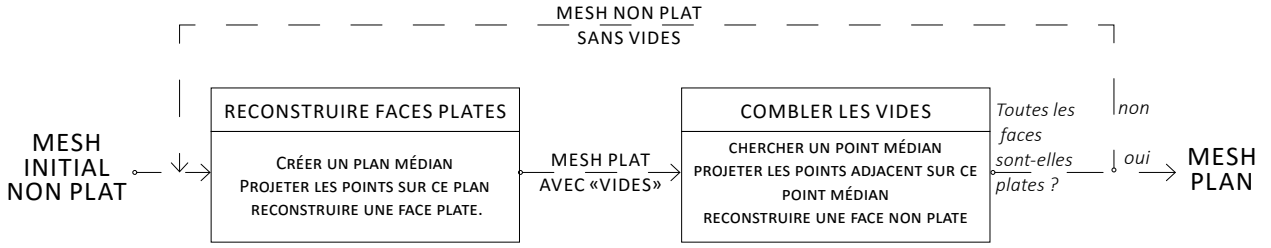
Bien que cette partie est composée d'un seul cluster, c'est la partie qui a demandé le plus de temps et de recherches. Le modèle 3D du mur obtenu à ce stade là est une géométrie dont les faces ne sont pas parfaitement plates car Rhinocéros est un modeleur pouvant créer des surfaces à partir de vertices non coplanaires. Cette technique de modélisation peut permettre de modéliser des doubles courbures par exemple. Cependant, dans notre cas la contrainte du matériau bois induit une géométrie plane[2]. C'est pourquoi le modèle 3D doit passer par une étape de «planarisation».

1 *Loop*

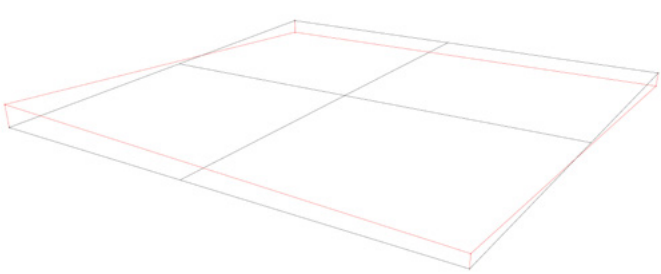
Une première méthode consistait a créer une boucle à l'aide du plugin Anemone[3] afin d'aplatir progressivement les faces (fig. 1). Cette dernière était composée de deux étapes :

- Un plan médian est défini pour chaque face du mesh à partir de leurs vertices. Deux contraintes sont définies sur les bords de la surface. Ainsi, Le plan médian des faces extérieures passe donc obligatoirement par une vertice présente sur ces bords. Tous les points des faces sont ensuite projetés sur leur plan médian, puis les faces sont reconstruites sur ces plans (fig. 2). On obtient ensuite en ensemble de faces totalement plates. Cependant, des vides entre les différentes faces sont maintenant présents.
- La deuxième étape consiste ensuite à recoller les faces entre elles en projetant tous les points adjacents en un point médian (fig. 3).

La boucle se répète ensuite afin d'arriver à un résultat où toutes les faces sont plates et qu'ils n'existe plus de vides entre les différentes faces du maillage.



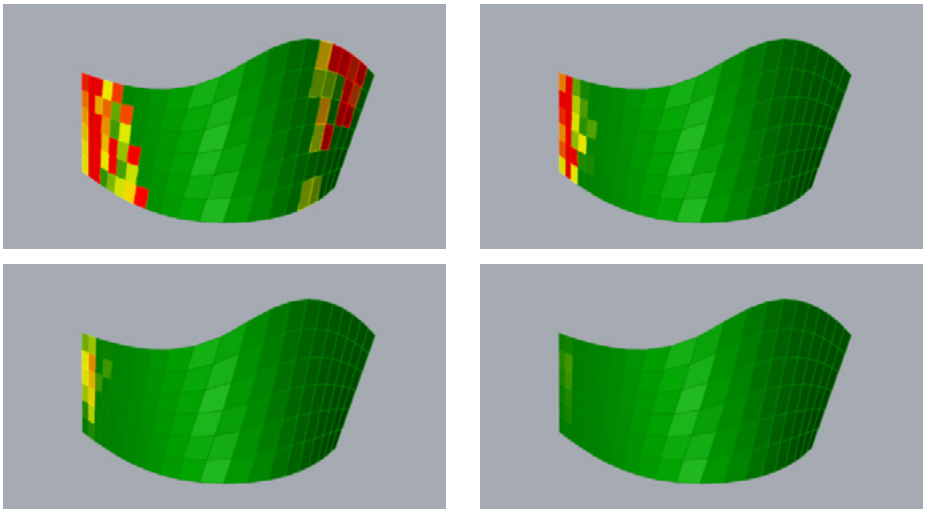
1. Principe de fonctionnement de la boucle



2. Projection des sommets sur le plan médian en rouge

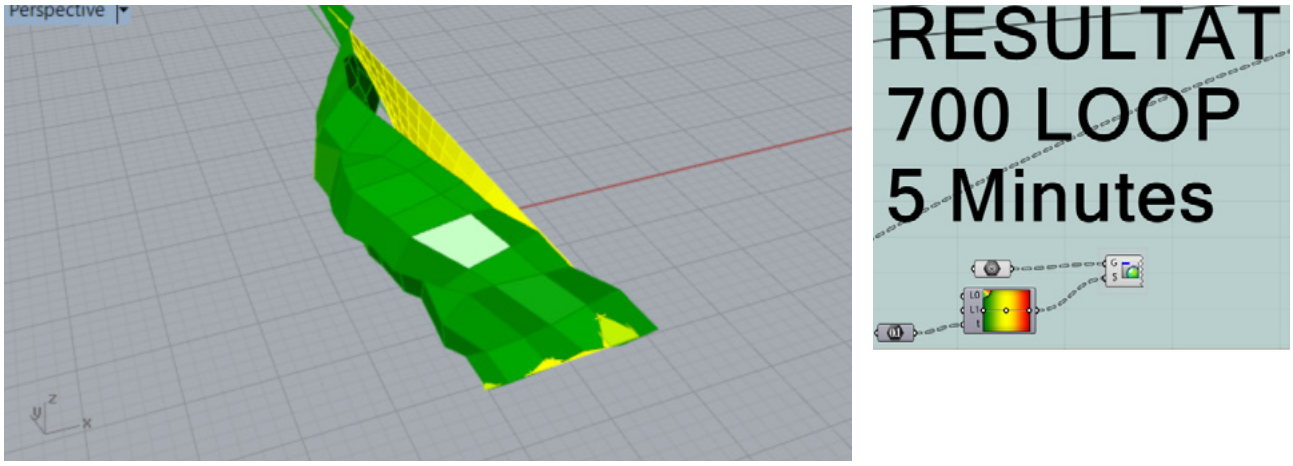
3. Projection des sommets adjacents en un point médian

2 Plane ou avec une petite tolérance, dépendant du matériaux lui même.
3 Plugin créé par Mateusz Zwierzycki et disponible sur Food4Rhino



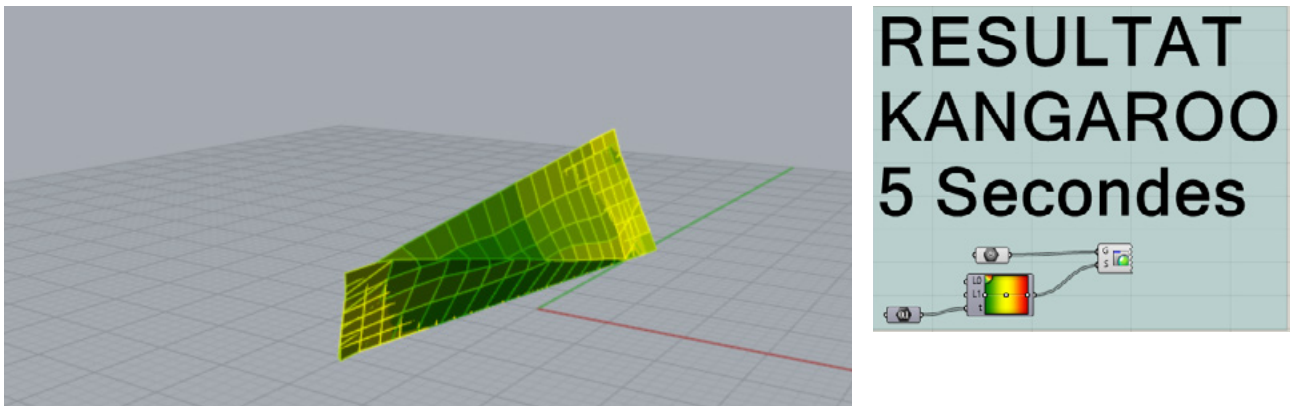
4. Test de planarization avec la boucle Anemone. Pattern et surface relativement simples. 250 boucles en 2 minutes de calcul

Après plusieurs tests (fig. 4), cette technique s'avère plutôt longue et les résultats obtenus peuvent vite devenir contestables dès que la surface initiale est un peu trop complexe. Comme on peut le voir sur les images ci-dessous (fig. 5), le résultat est très déformé par rapport à la surface originale et le calcul à duré plus de 5 minutes. Cependant, nous avons bien un mesh composé de faces parfaitement plates.



5. Test de planarization avec la loop. La surface est trop complexe et le résultat est très loin de la surface originale.

Un nouveau test a réalisé avec le plugin Kangaroo[2] avec les mêmes contraintes. Bien que le résultat est quelque peu étrange, la rapidité d'exécution laisse paraître un réel potentiel de la part de ce plugin.

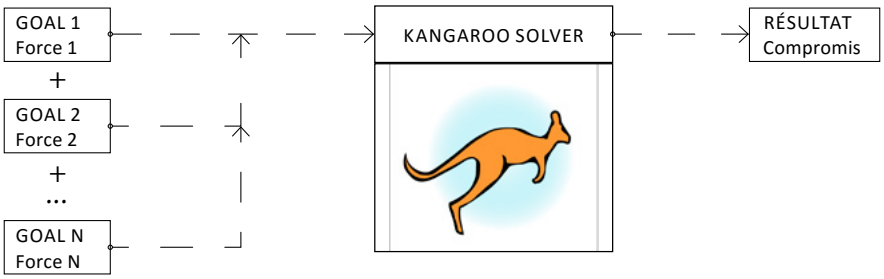


6. Test avec les mêmes paramètres que la figure 5 mais cette fois avec Kangaroo. Le résultat est beaucoup plus rapide et plus proche de la surface.

2 *Plugin créé par Daniel Piker et disponible sur Food4Rhino. On utilisera ici la version 2.0.2 sorti en avril 2015*

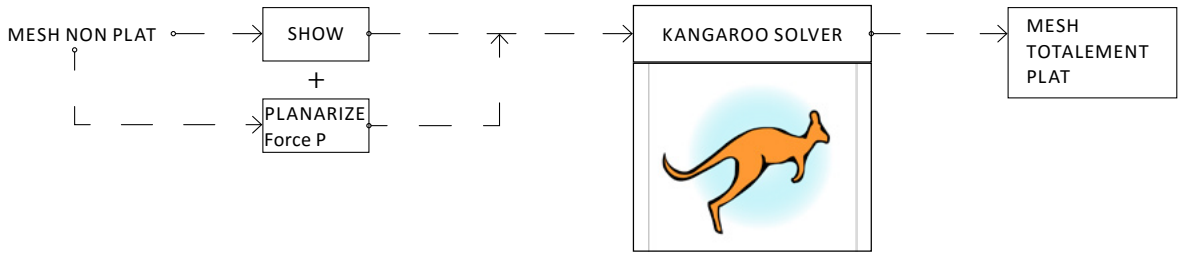
2 *Kangaroo*

Le plugin Kangaroo est un outil interactif permettant diverses simulations, optimisations, modélisations par la contrainte ou même de la recherche formelle. Kangaroo fonctionne à l'aide d'une série de composants «Goal» couplée à un composant «Solver» (fig. 1). Le fonctionnement d'un composant «Goal» semble assez clair puisqu'il s'agit d'un objectif[2] couplé à une force exprimée en valeur numérique. Ainsi, si l'on donne plusieurs «Goal» au solver, celui ci va tendre vers une solution qui résultera d'un compromis qui privilégie les objectifs dont la force est la plus importante.



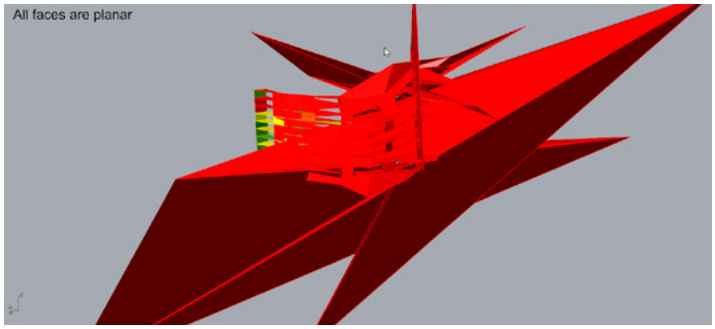
1. Principe général du plugin

Un «Goal» est cependant indispensable, c'est le composant «Show». Ce composant sert simplement à indiquer la géométrie initiale. La géométrie du résultat du solver sera obligatoirement du même type que celle entrée dans le composant «Show». De plus, tous les autres objectifs («Goal») doivent découler de la géométrie entrée dans le composant «Show». L'objectif «planarize» nécessite obligatoirement un mesh. Dans notre cas, nous aurons donc quelque chose comme le schéma ci-dessous (fig. 2).



2. Principe schématique du plugin dans le cas d'une planarisation d'un mesh.

Avec un plugin ainsi créé, le résultat de la planarisation peut prendre des formes très surprenantes et souvent très loin de la forme initiale. On s'aperçoit alors rapidement qu'il manque plusieurs objectifs supplémentaires pour obtenir un résultat proche de la surface dessinée par l'utilisateur. Néanmoins, l'ajout d'objectifs supplémentaires provoque rapidement des résultats très étonnants (fig. 3) et le compromis des forces appliquées à ces différents objectifs devient rapidement un casse-tête.



3. Exemple de bugs lors de la planarization avec Kangaroo

2 *Par exemple : pour le goal «coplanar», l'objectif est de rendre une série de points coplanaires. Le «Solver» va ainsi tendre vers la solution «tous les points sont coplanaires».*

En cherchant des cas similaires sur le forum www.grasshopper3d.com, j'ai trouver une réponse du créateur de Kangaroo, Daniel Piker, fournissant un ensemble de «Goal» afin de planariser les faces d'un mesh. Dans son fichier Grasshopper (fig. 4), D. Piker fourni un ensemble de 7 «Goals», chacun possédant sa propre force prédéfinie.

① «Planarize», c'est l'objectif principal, il permet de rendre les faces du mesh parfaitement plates. La force est à augmenter par l'utilisateur en modifiant un curseur (de 1.5 à 50) jusqu'à ce que toutes les faces soient totalement planes. Un code couleur est disponible en sortie et permet de visualiser l'état de chaque face. Un contrôle visuel de l'utilisateur est alors nécessaire.

② «Smooth», est un objectif de lissage

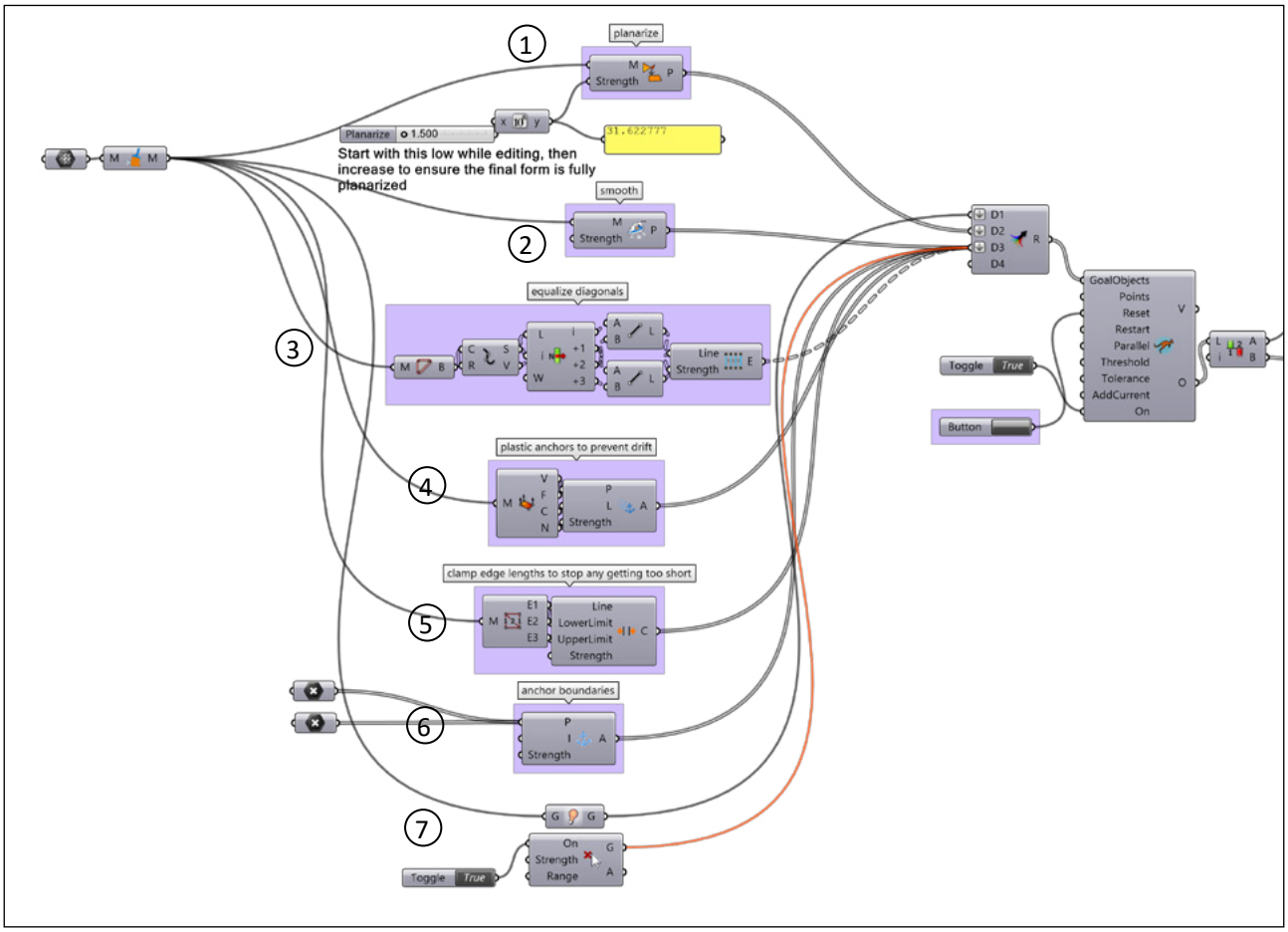
③ «Equalize Diagonals» permet de limiter la déformations des différentes faces du mesh en fonction de la longueur de leurs diagonales.

④ «Plastic Anchors», permet d'attribuer à toutes les vertices du mesh un contrainte de localisation pour empêcher le mesh de trop se déformer.

⑤ «Clamp Edges Length» permet de limiter la déformation des côtés de chaque face en créant un limite minimale et maximale de déformation.

⑥ «Anchor boundaries» permet d'ancrer le mesh sur plusieurs points en entrée.

⑦ «Show» permet d'indiquer la géométrie à planariser, ici le mesh.

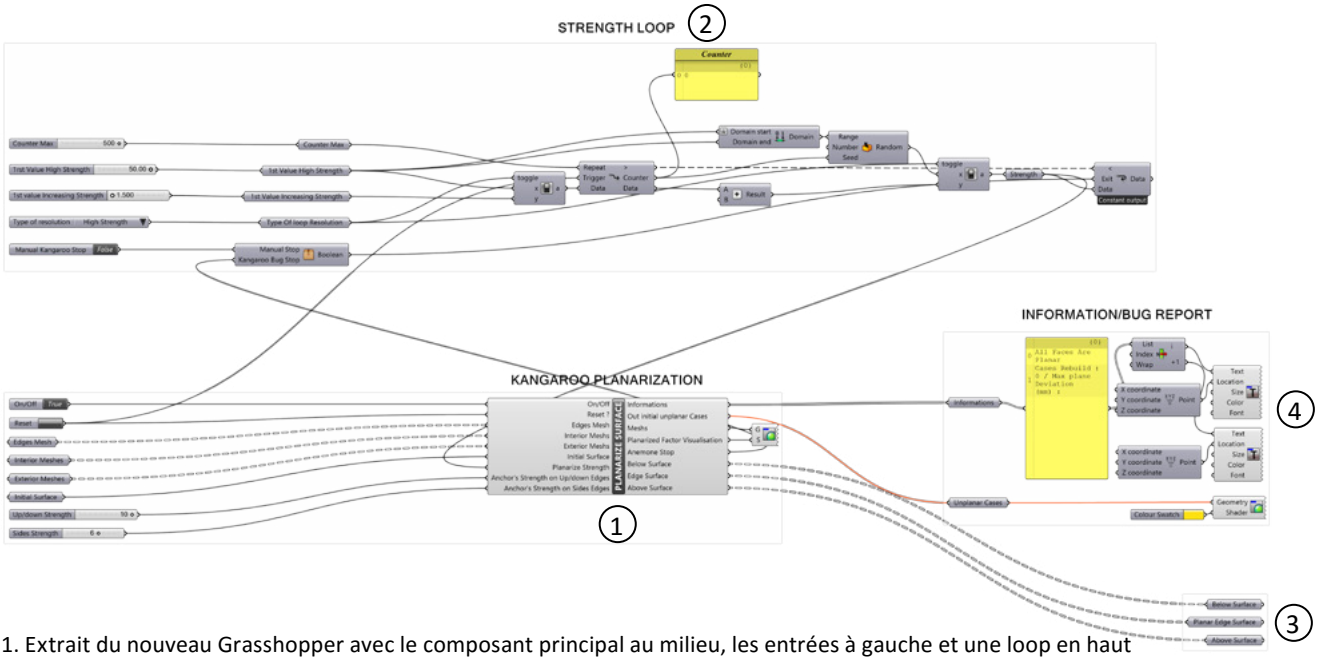


4. Fichier grasshopper fourni par Daniel Piker sur www.Grasshopper3d.com

Cette méthode fonctionne plutôt bien et le nombre de bugs est considérablement réduit. Cependant, le contrôle manuel n'est pas pratique pour un utilisateur lambda.

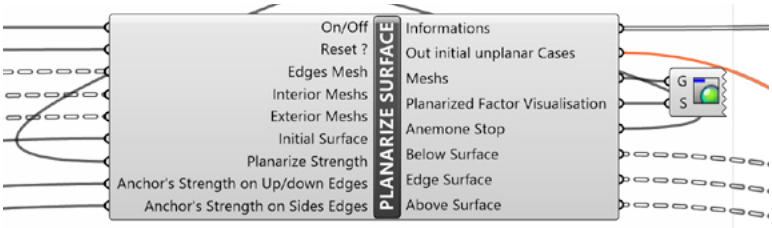
③ *Améliorations*

Plusieurs modifications, ajustements et corrections de bugs ont été rapporté à cette méthode pour obtenir un unique cluster utilisable par le concepteur.



1. Extrait du nouveau Grasshopper avec le composant principal au milieu, les entrées à gauche et une loop en haut

① - La première étape a été de créer ce composant unique. Par rapport, au cluster de Daniel Piker, la contrainte de localisation créée par des points a été modifiée. Dans la nouvelle méthode, cette contrainte est divisée en deux contraintes, une pour les côtés droits et gauches de la surface, une autre pour les côtés haut et bas. Chaque contrainte possède sa propre force d'application, allant de 0 à 10 et permet d'offrir un degré de liberté à la planarisation du mur sur un couple de côtés. Comme nous pouvons le voir sur les deux vues (fig. 1 et 2), un certains nombre de paramètres d'entrées et de sorties ont été rajouté à ce cluster.



2. Zoom sur le cluster principal regroupant tout le Grasshopper de D. Piker plus des améliorations

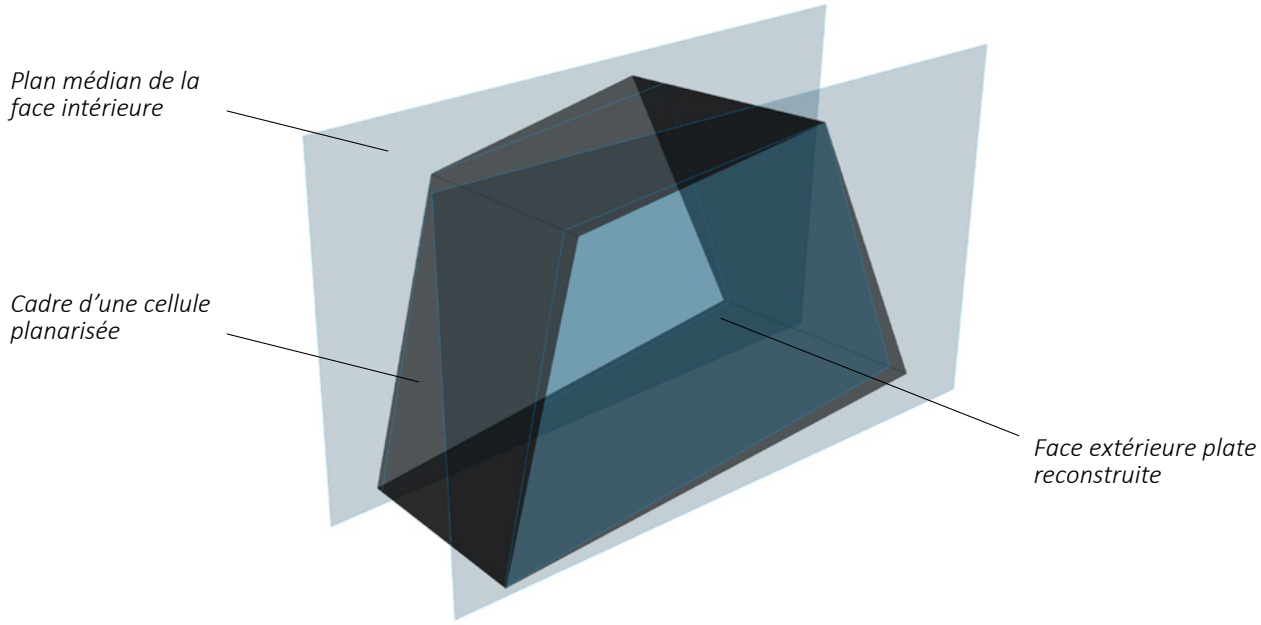
② - En faisant de multiples tests, j'ai pu remarqué que pour réussir une planarisation sans bugs, il y avait deux méthodes dépendantes du pattern initial. Il faut soit augmenter graduellement la valeur de la force de planarisation de 1,5 à 50, soit jouer avec une multitude de valeurs comprises entre 49,0 et 50,0.

Dans l'idée de réduire au maximum les tâches de l'utilisateur, une boucle avec le plugin Anemone a été créé. L'utilisateur doit d'abord choisir son type de boucle[2], entrer la fin de la boucle dans l'entrée du cluster de la planarisation et connecter la sortie «Anemone Stop» dans la partie «Exit» de la fin de boucle. Cette boucle permet simplement de générer des valeurs différentes en continu et s'arrête automatiquement dès que toutes les faces sont plates[3]. Cette méthode interrompt automatiquement le Solver de Kangaroo dès que la boucle s'arrête.

2 Valeurs de 1,5 à 50 ou valeurs entre 49,0 et 50,0.
3 Booléen d'arrêt présent dans la sortie «Anemone Stop»

Bien que cette méthode fonctionne parfaitement, elle reste un peu complexe pour un utilisateur lambda. Le plugin Anemone ne peut être contenu dans un cluster, il n'est donc pas possible d'intégrer cette fonction mais un script VB permettrait de créer une boucle similaire à l'intérieur du composant.

(3) - Après avoir essayé plusieurs méthodes[4], la planarisation des faces du cadre des cellules s'avère être la plus efficace. Une reconstruction des faces intérieures et extérieures est cependant nécessaire dans un second temps. Pour planariser ces deux faces, deux plans médians sont définis à partir les sommets du cadre d'une cellule. Une fois défini, l'intersection entre chaque plan et le BREP du cadre de la cellule est calculée. Il ne reste plus qu'à couper le cadre en fonction des intersections pour enfin reconstruire le cadre et les surfaces intérieures et extérieures (fig. 3). Cette opération peut être longue à calculer, c'est pourquoi cette opération n'est active que lorsque toutes les faces du cadres des cellules sont totalement plates.

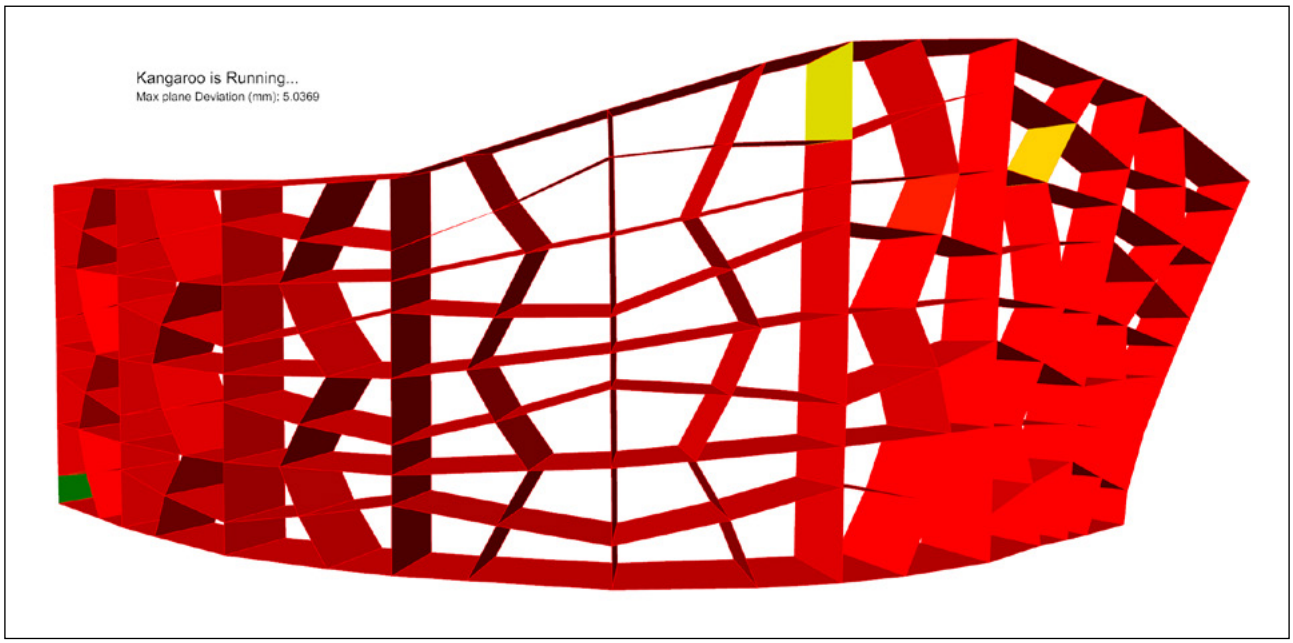


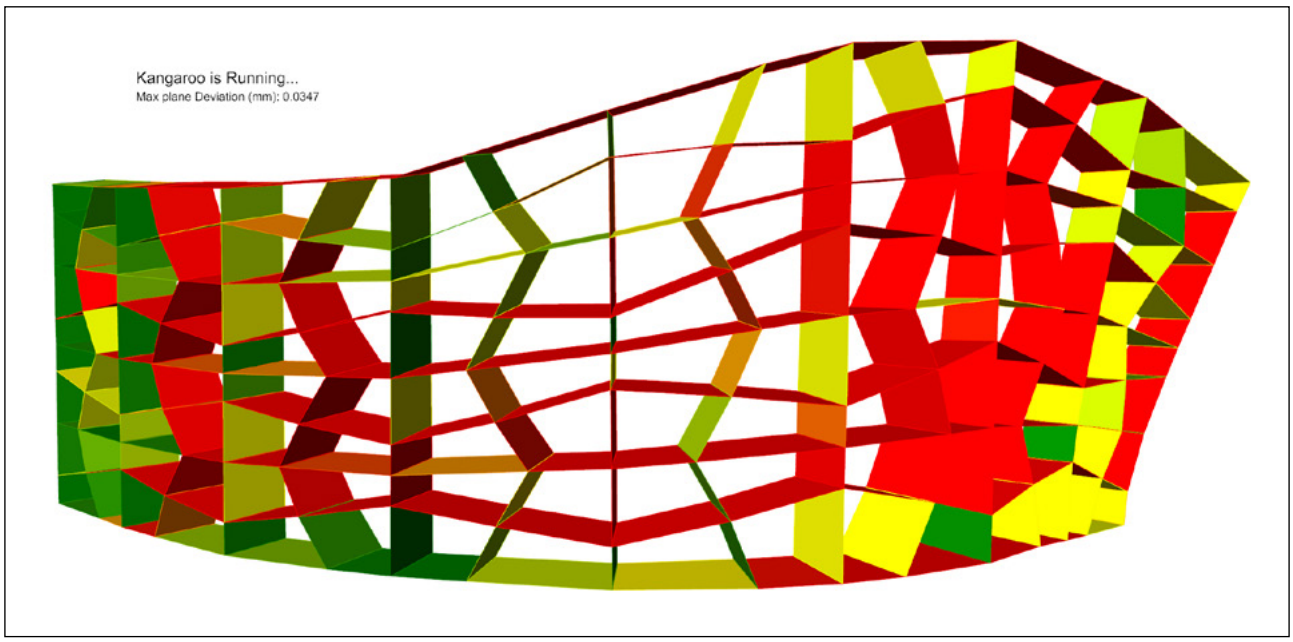
3. Principe de création des surfaces intérieures et extérieures. En bleu les plan médian des sommets des côtés.

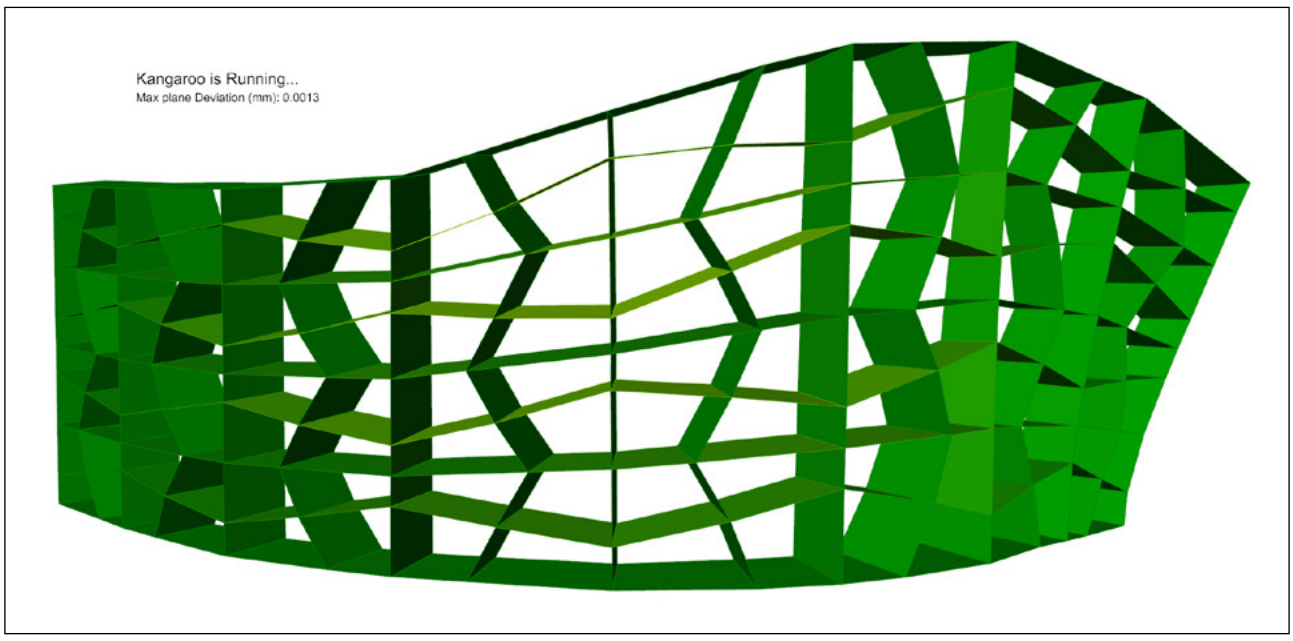
(4) - À la suite de cette étape, toutes les surfaces de chaque cellule doivent être parfaitement plates et présentes sous la forme de surface NURBS. Cependant, certains bugs peuvent arriver, lors de la planarisation ou lors de la reconstruction de certaines faces. C'est pourquoi des sorties «information» et «Out Initial Unplanar Cases» sont présentes dans ce cluster permettant à l'utilisateur de vérifier l'état de la planarisation. Un correcteur de bug à été créé afin de reconstruire les cellules qui n'était pas totalement plates. Cependant, cette reconstruction peut entraîner des chevauchements entre les cellules et demande peut être à être retravaillée. Les caissons initiaux avant la reconstruction sont disponibles dans la sortie «Out Initial Unplanar Cases».

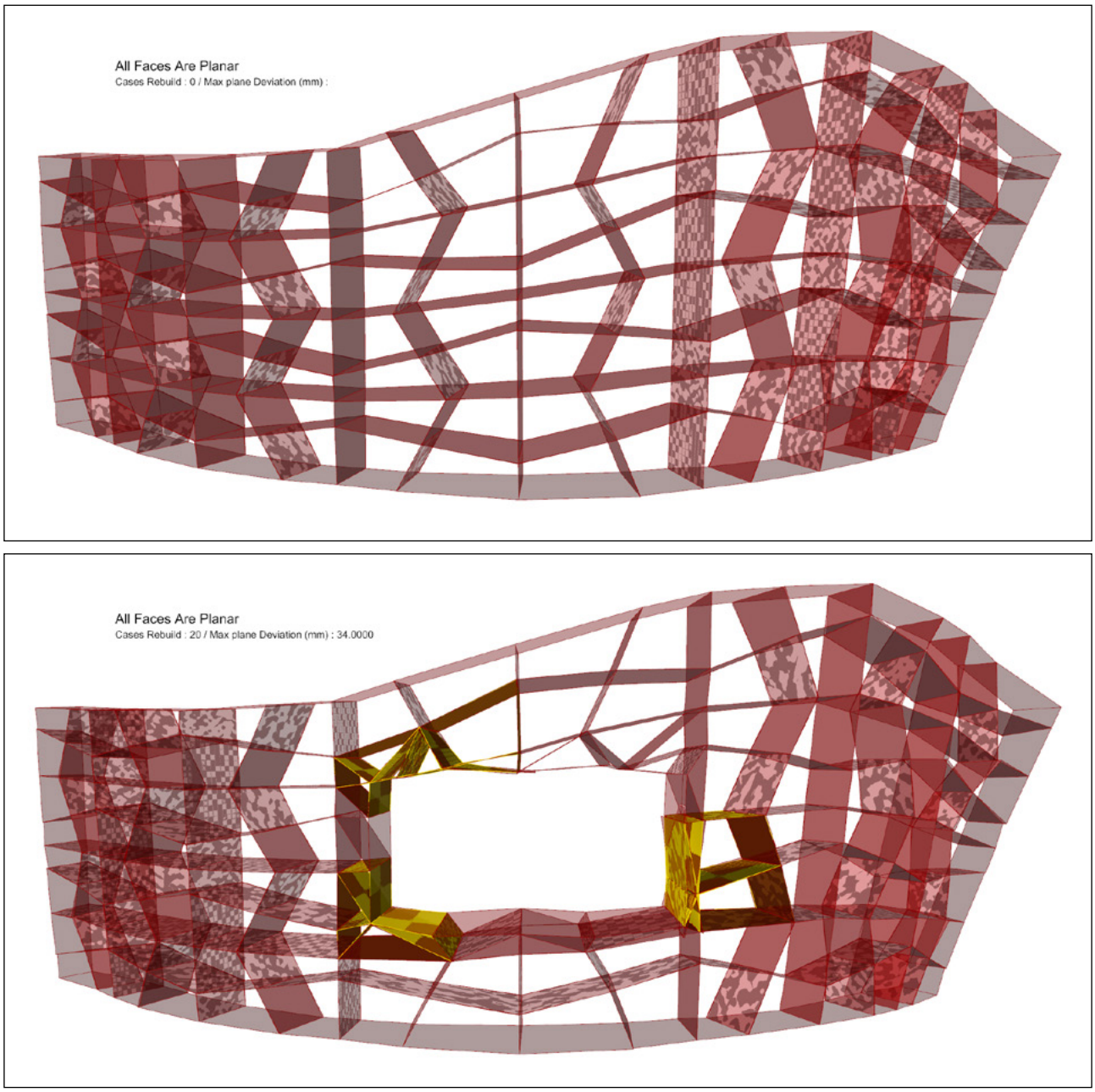
De plus, la visualisation de l'état de chaque face, en fonction d'un code couleur allant du rouge au vert créé par Daniel Piker, à été rajouté au cluster. Un certains nombre de textes ont également été créés. Cela permet à l'utilisateur d'observer l'état de la planarisation pendant et après le calcul. Ces sorties permettent également d'observer les cellules qui ont du être reconstruites ainsi que la courbure maximale des surfaces initiales avant la reconstruction.

4 *Faces intérieures, faces extérieures, cadre des cellules ou toutes les cellules dans un même mesh*


Kangaroo is Running...
Max plane Deviation (mm): 5.0369


Kangaroo is Running...
Max plane Deviation (mm): 0.0347


Kangaroo is Running...
Max plane Deviation (mm): 0.0013



4. Série des captures d'écrans prises lors de la planarization d'un élément. On peut ainsi observer l'évolution du calcul du plugin.

On peut observer l'état général du cluster en haut à gauche de l'écran : «Kangaroo is running» ou «All faces are planar». On peut également y voir «Kangaroo was Stopped», un message d'erreur signalant que l'opération à été arrêté car une ou plusieurs cellules étaient trop déformées par rapport à leur forme initiale.

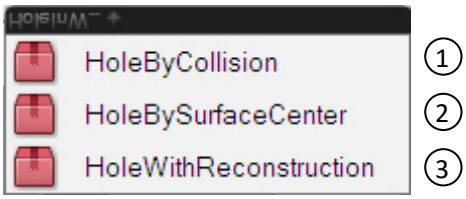
En dessous de l'état, On peut observer la courbure maximale des faces. Grasshopper considère qu'une surface est plate pour une déviation inférieure à 10^{-3} mm par rapport au plan médian.

Une fois que l'opération est finie, le texte indique le nombre de caissons reconstruits ainsi que la déviation maximale des surfaces initiales. Dans l'exemple ci-dessus, les cellules en jaune sont celles qui ont été soumises à une reconstruction.

À ce jour, cette étape est quasiment fonctionnelle mais comporte toujours quelques erreurs surtout avec des surfaces très courbées. Une optimisation des composants pourrait également accélérer le processus.

f. Hole In Wall

Cette partie permet de créer des perforations dans la paroi pour d'éventuelles baies. Trois manières de créer ces ouvertures sont proposées (fig. 1). Pour ces trois composants, la baie est représentée par un solide en tant qu'un BREP fermé (ClosedBREP).

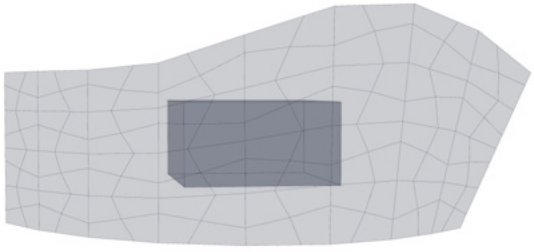


1. Volet déroulant proposant tous les clusters de la partie «HoleInWall»

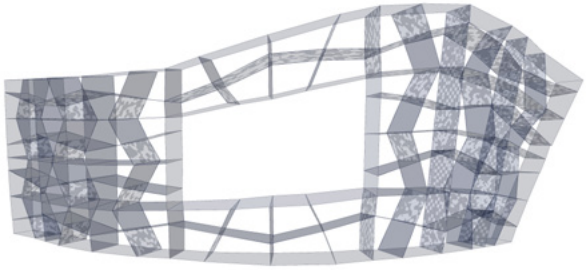
① ② «HoleByCollision» et «HoleBySurfaceCenter» permettent simplement de séparer des cellules déjà créées en deux parties, une liste de cellules présentes à l'intérieure du volume, une autre pour ceux qui sont en dehors de ce BREP (fig. 1). Cette étape se déroule après l'étape «planarization», une fois que le mur est modélisé et totalement plat. L'utilisateur peut seulement séparer les surfaces qu'il souhaite. Il peut ainsi, par exemple, créer une baie en conservant la structure des cellules, en supprimant simplement les fonds de la cellule. De même, il peut créer des différences de matériaux à l'aide de ces clusters (fig. 4).

«HoleByCollision» (fig. 2) sépare toutes les faces entrant en collision avec ce BREP. Ainsi, toutes les surfaces ayant, au minimum, une vertice à l'intérieur du volume seront identifiées «à l'intérieur du BREP».

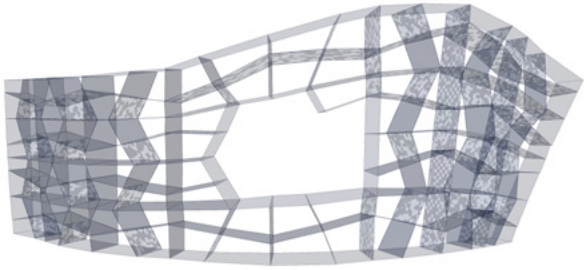
«HoleBySurfaceCenter» (fig. 3) quant à lui sépare les faces en fonction de la localisation du centre de ces dernières. Pour qu'elles soient qualifiées «à l'intérieur du BREP», il faut que leur centre soit situé à l'intérieur du volume.



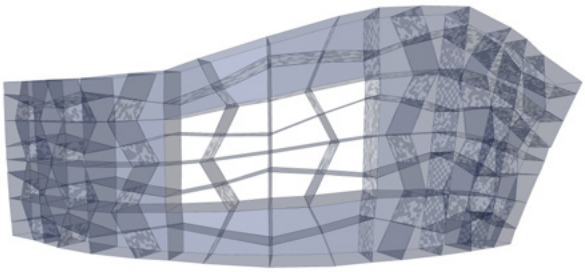
1. Pattern initial «Cairo» sur une surface avec le volume (brep) utilisé pour créer la Baie



2. Surface à l'extérieur du volume avec «HoleByCollision»

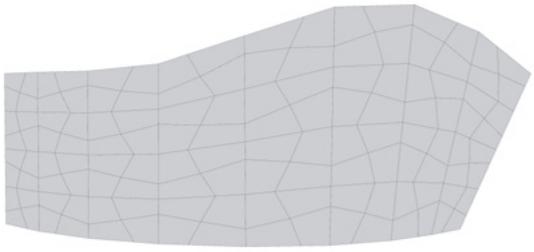


3. Surface à l'extérieur du volume avec «HoleBySurfaceCenter»

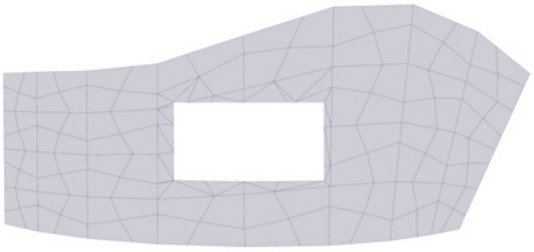


4. Exemple de baie en enlevant simplement les fonds présent à l'intérieur du volume de la baie.

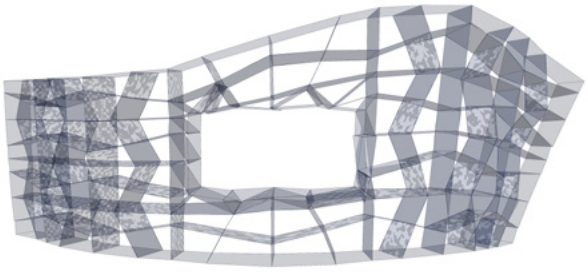
③ «HoleWithReconstruction» (fig. 5, 6 et 7) est un peu plus complexe puisqu'il reconstruit les surfaces du pattern coupées par le volume afin de totalement cadrer ce dernier. Pour cette étape le volume du BREP est transformé en mesh. Ensuite, une fonction de base de Grasshopper permet de créer un percement dans un mesh en fonction d'un autre et de reconstruire ce dernier. Afin d'éviter le maximum d'erreurs, cette étape doit être située entre la partie «Pattern» et la partie «Volume». (fig. 8)



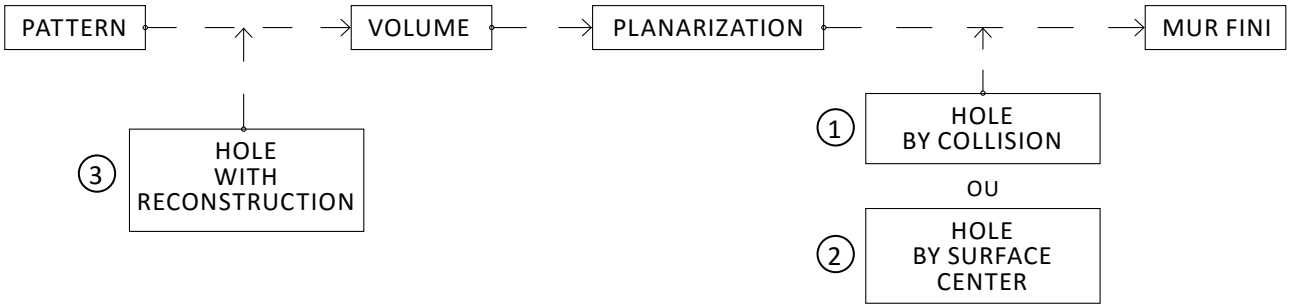
5. Pattern initial «Cairo» similaire



6. Pattern reconstruit avec «HoleWithReconstruction»

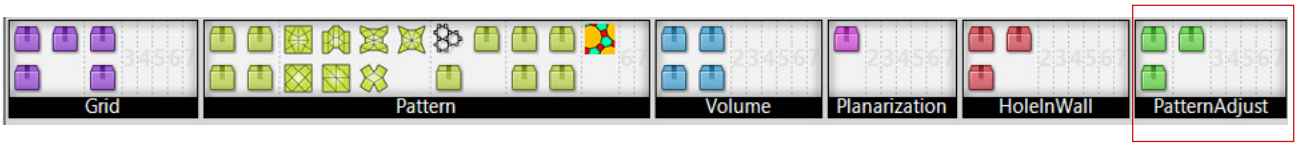


7. Volume créé suite au pattern reconstruit

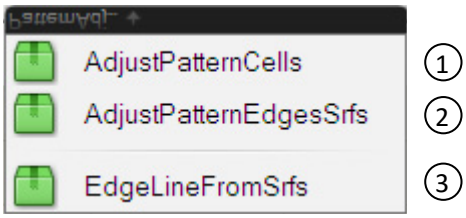


8. Schéma d'utilisation des 3 clusters de cette partie

g. AdjustPattern



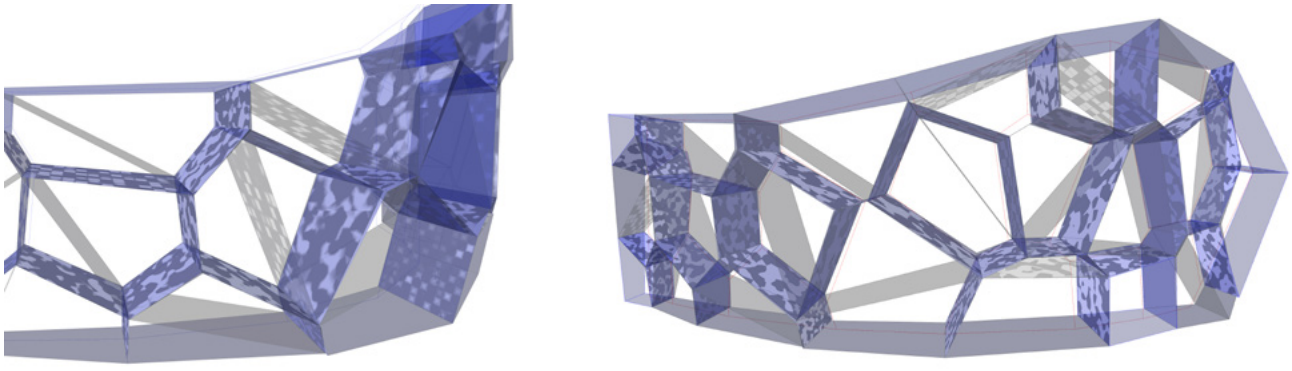
Comme nous l'avons vu précédemment, un mesh composé de quad et de triangle est nécessaire pour les parties «Volume» et «Planarization». Cette dernière partie a été créée afin de recréer les cellules de patterns à la géométrie plus complexe. Les clusters «AdjustPatternCells» et «AdjustPatternEdgesSrfs» (fig. 1) ont été conçus dans le but de reconstruire les cellules en fonction d'un pattern initial sous la forme de polylignes fermées. Ces dernières peuvent être disponibles à partir du cluster «MeshFromPolylines[2]».



1. Surface à l'extérieur du volume avec «PatternAdjust»

(1) - «AdjustPatternCells» est le composant permettant de reconstruire la totalité d'une cellule (fig. 3). Ce cluster a besoin des surfaces intérieures des cellules, des surfaces de leur cadre, ainsi que du pattern initial sous forme de polylignes fermées.

La première étape de ce composant est de regrouper toutes les cellules à fusionner en fonction du pattern initial. Nous avons donc pour chaque future cellule, une liste de cellules à fusionner (fig. 2). Nous pouvons ensuite détecter les cadres en trop, situés à l'intérieur du contour des futures cellules. Les surfaces des cadres en trop sont ensuite supprimées. Pour chaque cellule, il ne reste ainsi plus que les surfaces qui composent leur cadre. Une reconstruction de ce cadre au niveau des jonctions est ensuite souvent nécessaire afin d'obtenir une géométrie adéquate : un seul BREP.

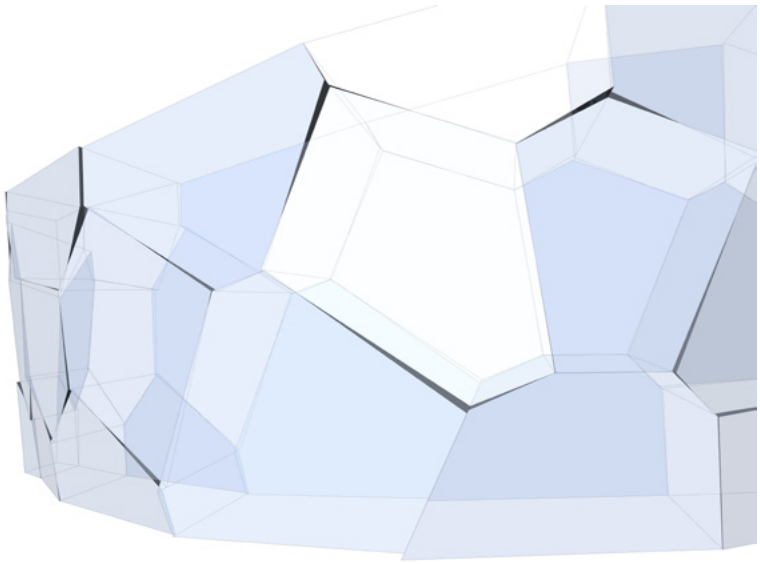


2. En bleu, les cellules reconstruites, en gris, les cellules initiales

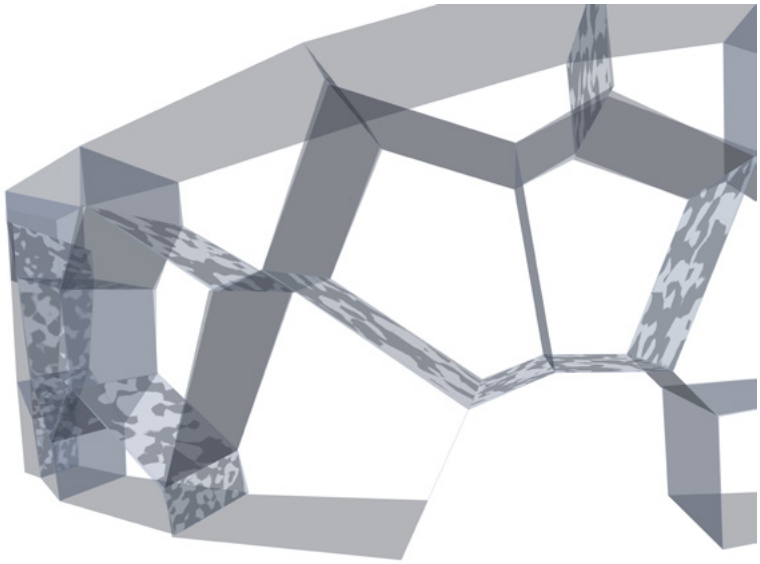
Il faut ensuite reconstruire les nouvelles faces intérieures et extérieures de ces nouvelles cellules. Cette technique est similaire à celle utilisé dans le cluster «Planarization»[3]. Cependant, cette partie produit de nombreuses erreurs, est assez longue à calculer et n'est pas encore totalement au point.

2 *Cluster présent dans la quatrième sous partie de «Pattern»*
3 *Couper le cadre par le plan médian à chaque bord*

② - «AdjutPatternEdgesSrfs» fonctionne de la même manière sauf qu'il ne comporte pas cette deuxième partie beaucoup plus longue à calculer et pas encore au point. Il permet ainsi simplement de récupérer les surfaces du cadre des cellules recomposées (fig. 4). Cependant, elle n'est utile que pour une visualisation rapide et ne permet pas d'avoir des géométries adéquates pour la suite du plugin.



3. Reconstitution du mur avec le cluster «AdjustPatternCell», toutes les faces sont plates



4. Reconstitution du mur avec le cluster «AdjustPatternEdgesSrfs». Les faces des cadres à l'intérieur des cellules sont supprimées

③ - «EdgeLineFromSrfs» permet simplement de récupérer les arrêtes du cadre de chaque cellule. Ce cluster nécessite simplement les surfaces intérieures et extérieures de chaque cellule correctement listée. En sortie, les arêtes sont disponibles avec le même niveau de liste qu'en entrée.

4. CONCLUSION

a. Fonctionnement général

Nous avons ainsi vu les six grandes parties qui constituent ce plugin. Elles correspondent aux différentes étapes de la conception d'un mur non-standard cellulaire en bois. Ainsi, un seul cluster de chaque catégorie est nécessaire pour construire son mur (bien que les parties 1, 5 et 6 soient facultatives). Pour chaque partie, l'utilisateur doit ainsi faire des choix de cluster mais aussi de paramètres d'entrées, (numériques, booléens, géométriques). En résumé, l'utilisateur doit :

⓪ Choisir une forme : modéliser une surface
② Choisir un type de pattern
- Paramètres liés au pattern
- Densité/taille du pattern (Peut être fait en fonction d'une grille ①)

Ou dessiner un pattern
- Type de projection sur la surface

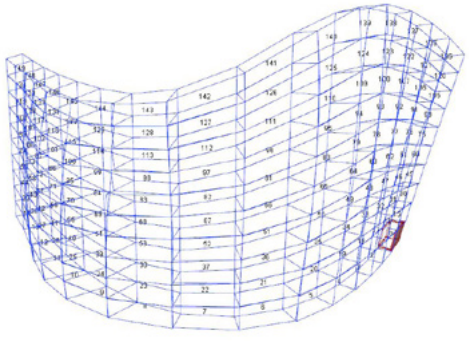
③ Choisir un type d'extrusion des cellules
- Épaisseur

④ Rendre la géométrie totalement plate
⑤ Choisir un type de Baie (Optionnel)
- Forme

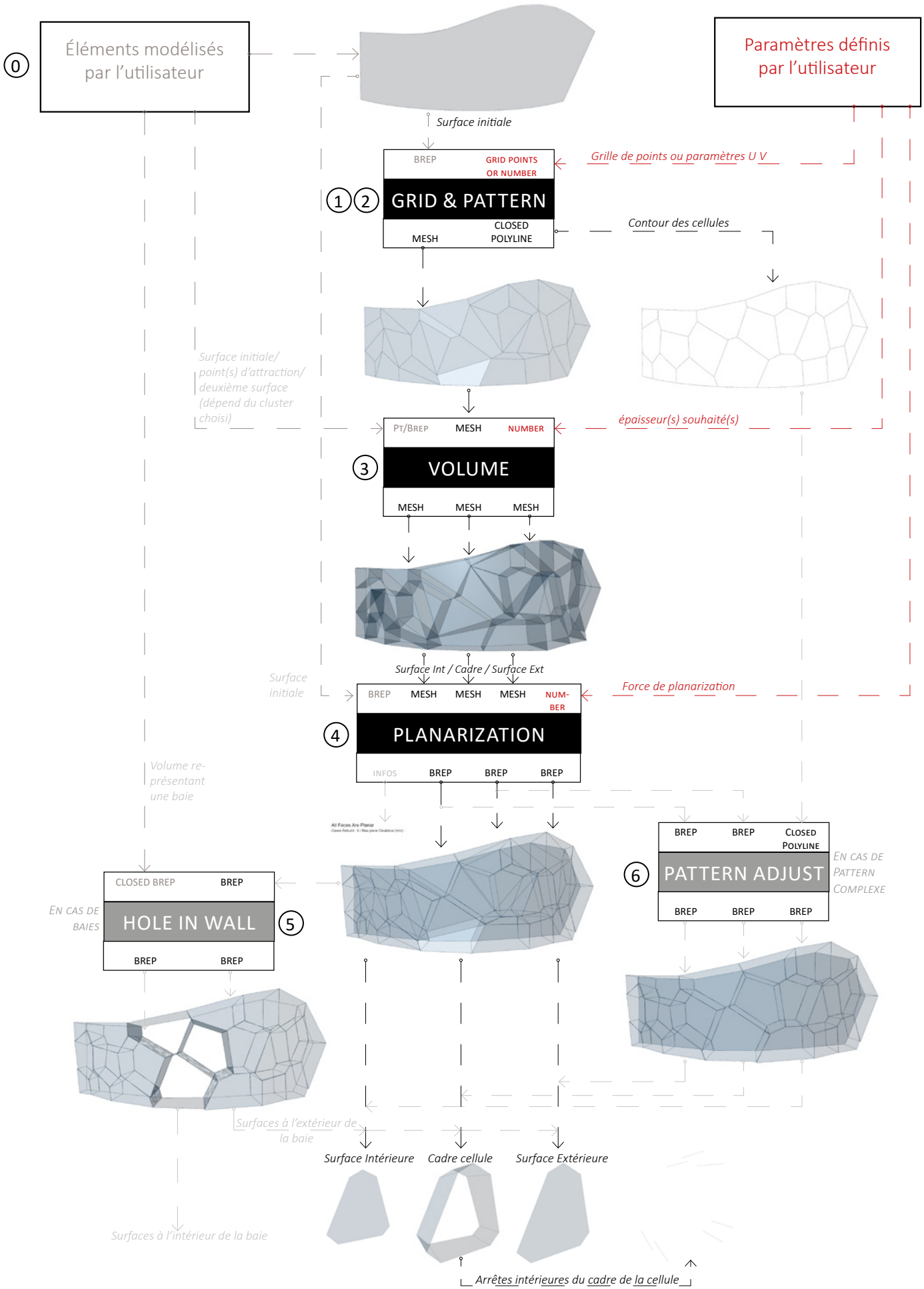
⑥ Choisir de reconstruire les cellules (Optionnel)

Pour permettre à l'utilisateur de choisir un cluster parmis tous ceux qui lui sont proposés, Il est très important de bien définir et d'harmoniser les paramètres d'entrées et de sorties des clusters de chaque partie. La fig. 2 ci-contre, montre le principe de fonctionnement général du plugin ainsi que l'ordre d'utilisation des différentes étapes. On peut y trouver les différentes données importantes à transmettre en entrée et sortie de chaque partie.

En suivant correctement ce schéma, toutes les parties du plugin fonctionnent correctement et les objectifs principaux ont été atteints. L'algorithme a permis de générer un mur complet avec des géométries adéquates, lisibles et identifiables pour la partie «assemblages». Chaque caisson du mur a ensuite pu être listé, et les assemblages ont pu être modélisés (fig. 1). L'objectif principal de créer un outil s'inscrivant dans un continuum numérique a donc été atteint.



1. Visualisation du mur et des différents caissons avec le fichier grasshopper «assemblage»



2. Principe de fonctionnement général

b. Ouverture

Avec du recul, les principales limites de ce plugin résident dans la partie pattern et la partie planarisation.

En effet, le temps très restreint de ce stage nous a obligé à nous focaliser directement sur le travail existant pour créer un outil fonctionnel; L'objectif principal étant de créer un outil qui soit inscrit dans un continuum numérique allant de la conception à la fabrication. Ainsi la première étape était de créer au moins un exemple d'un pattern subdivisant une surface et créant un mur qui serait compatible avec la partie assemblages. Certaines erreurs ont été quelques fois contournées afin de réussir à créer un exemple qui fonctionne.

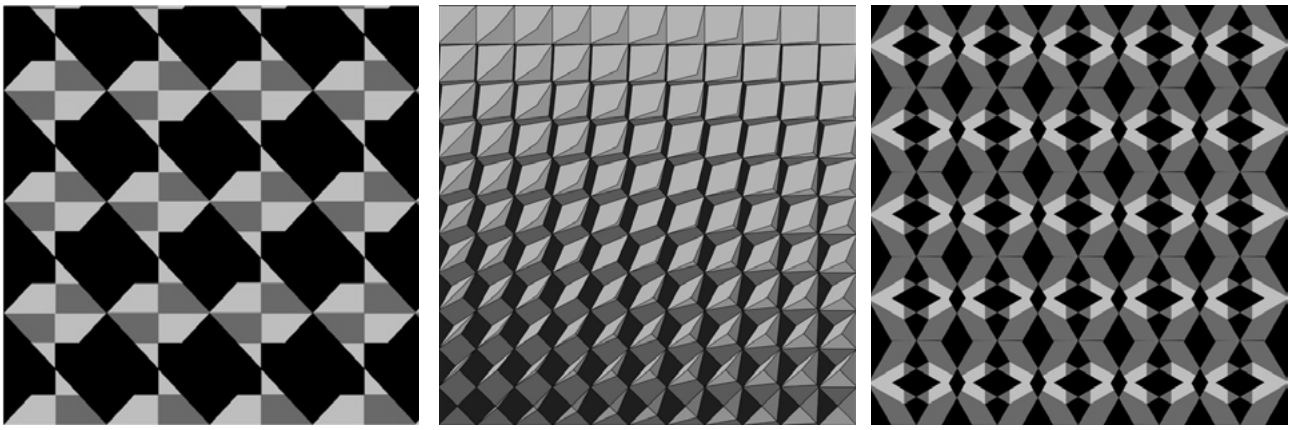
PLANARIZATION

Bien que cette partie soit actuellement fonctionnelle pour la plupart des patterns, elle ne l'est que pour un certain type de surface. En effet, pour des surfaces plus complexes, notamment des doubles courbures, cette étape provoque systématiquement des erreurs. Il devient alors impossible de planariser certains patterns. Actuellement, c'est à l'utilisateur de contrôler le fonctionnement de cette étape et de faire preuve de bon sens en modélisant une surface adéquate et pas trop complexe. On pourrait cependant imaginer une mise en place d'un test de la courbure de la surface avant cette étape. Ce test pourrait, soit détecter, soit directement modifier la surface lorsqu'elle est trop complexe.

PATTERN

La méthode initiale d'Oskar avait un réel avantage au niveau de la liberté de l'utilisateur. Cependant, cette méthode a été totalement abandonnée et remplacée par la création d'une bibliothèque de pattern. Là encore, les problèmes ont été contournés afin de continuer l'ensemble du plugin. Cependant d'un point de vue de la conception une telle approche est très intéressante. Il existe peut être un moyen d'automatiser la reconstruction de tous les vides présents dans le pattern lors de certains paramètres. Il existerait peut être également de nouveaux outils permettant d'aider ou guider la création de pattern, d'autres interfaces ou techniques, permettant de créer de nouveaux types de patterns (fig. 1).

Un grand travail de recherche aurait été nécessaire en amont pour se rendre compte de la complexité du monde des patterns. Ce travail aurait permis de prendre du recul sur la méthode d'Oskar et d'ouvrir le champ des possibilités d'un tel plugin. Bien que cet outil propose une bibliothèque de patterns conséquente, la méthode de construction de ces derniers reste plus ou moins la même. En effet, l'ensemble des patterns est créé à partir d'une grille de points découlant de paramètres UV d'une surface. Or, il existe une infinité de manière de concevoir ou de générer des patterns. On pourrait très bien imaginer générer des cellules à l'aide d'une image ou d'un son par exemple. Ce genre de technique de génération de pattern existe déjà dans l'art, et notamment en graphisme. C'est le cas de «l'art génératif» technique utilisé par l'artiste, designer et programmer Joshua Davis (fig. 2). Bien que ce dernier génère ces œuvres à l'aide de logiciel de programmation «Processing», de tels algorithmes peuvent surement être conçus sur Grasshopper. De plus en plus d'artistes s'inscrivent dans ce courant artistique. Le site internet www.chromeexperiments.com héberge de nombreux algorithmes créés par des artistes et propose aux utilisateurs de «jouer» avec ces algorithmes (fig. 3). L'interface est très simplifiée et permet a tout utilisateur lambda de s'approprier l'algorithme pour créer sa propre œuvre.



1. Exemples de patterns générés par Grasshopper conçus par des étudiants et disponibles sur www.designcoding.net

2. Travail de Joshua Davis, œuvres générées à partir de sons.

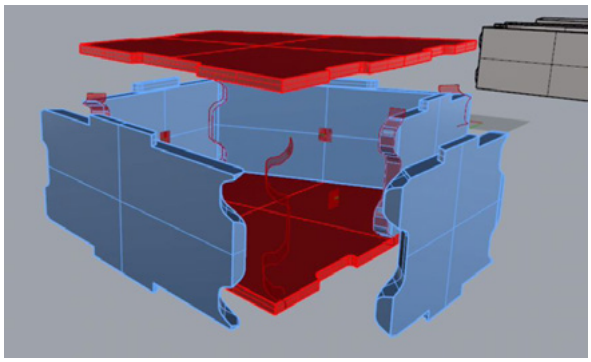
2. Algorithme «Particle Love» créé par Edan Kwan hébergé sur www.chromeexperiments.com

III. MODÉLISATION PARAMÉTRIQUE D'ASSEMBLAGES EN BOIS

1. INTRODUCTION

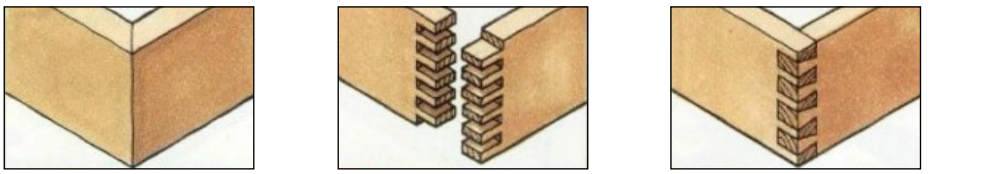
a. Approche générale

Le but de cet outil est de générer des assemblages paramétriques en bois qui permettent de lier les différents panneaux constituant une cellule. Lors de notre stage, ce travail avait déjà été commencé plusieurs mois avant par Oskar Gámez dans le cadre de sa thèse (fig. 1). Nous avons repris et poursuivi son travail sans réellement prendre de recul ou dresser un état de l'art sur ce qui existait, ce qu'il était possible de faire ou ce qui était souhaitable. Il existe effectivement un nombre presque infini de manières d'assembler et de lier plusieurs panneaux de bois entre eux pour former une cellule. C'est pourquoi les possibilités et les limites de cet outil sont à considérer dans un contexte de travail précis et n'ont pas forcément été décidé dès le début de notre travail mais apparaissent plutôt comme les limites d'un système en développement qui pourrait bien évidemment être amélioré.



1. Assemblage des panneaux d'un caisson par Oskar Gámez

Nous allons donc nous attacher à vous décrire les possibilités actuelles de cet outil et la manière dont sont générés les assemblages, étape par étape afin d'identifier clairement où sont les limites et quelle est leur nature afin de permettre et de faciliter un éventuel travail ultérieur sur cet outil par d'autres personnes.



Assemblage en biseau simple Assemble à queue droite Assemblage à queue d'arronde

2. Assemblages traditionnels. Extrait de «Good Wood Joints» par Albert Jackson & David Day

La nature de ces assemblages est directement inspirée des assemblages traditionnels utilisés en menuiserie (fig. 2) tels que l'assemblage en biseau simple, l'assemblage à queue droite ou l'assemblage à queue d'arronde. Le choix de ce type d'assemblage est dû à plusieurs facteurs :

- La réalisation de cette architecture sera effectuée par une méthode d'usinage soustractive
- L'usinage sera robotisé ce qui autorise des découpes complexes et «non standards»
- L'utilisateur doit pouvoir choisir entre plusieurs types d'assemblages
- La méthode de génération du motif de ces assemblages, expliqué par la suite, est également un facteur limitant

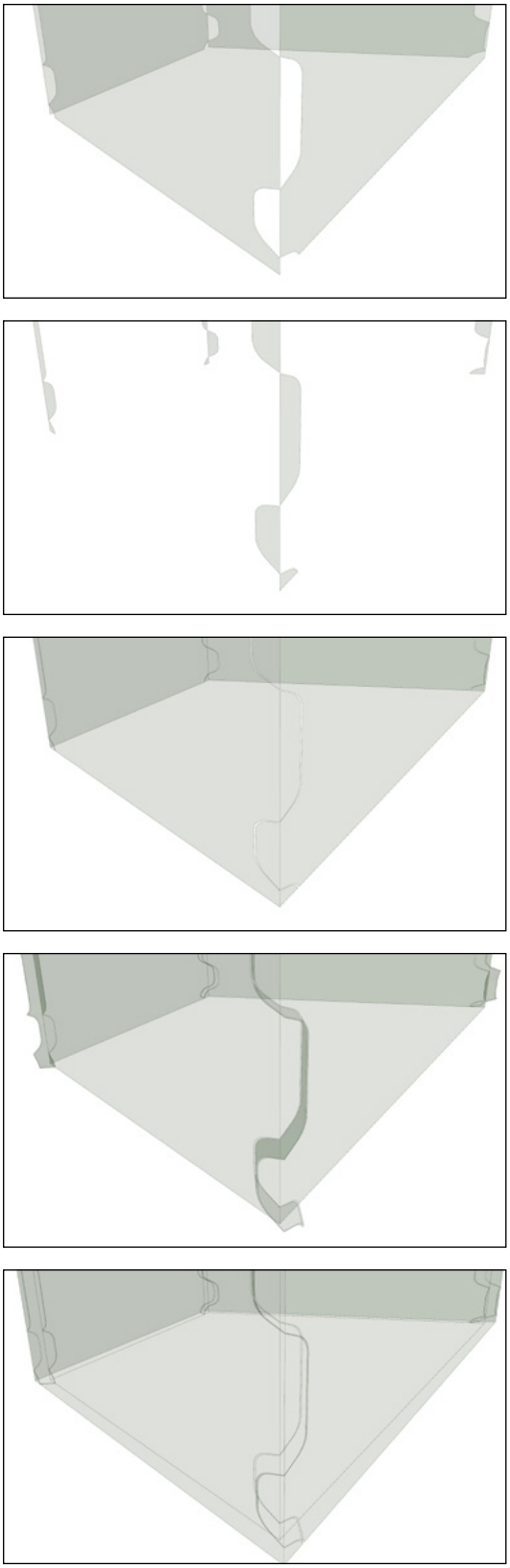
b. Travail pré-existant effectué par Oskar Gámez

Le travail débuté par Oskar Gámez sur la modélisation paramétrique d'assemblages en bois lui a permis de démontrer expérimentalement la faisabilité de son travail de recherche sur les murs non standards en bois. Le travail que j'ai pu effecté lors du stage a d'abord consisté à reprendre et améliorer le programme qu'il avait conçu mais qui comportait de nombreuses erreurs et ne permettait pas de traiter correctement une série de cellules différentes avec les mêmes paramètres. La méthode de construction des cellules mise en place par Oskar reste cependant très proche de celle suivie par l'algorithme actuellement et vous sera présentée dans les prochaines pages.

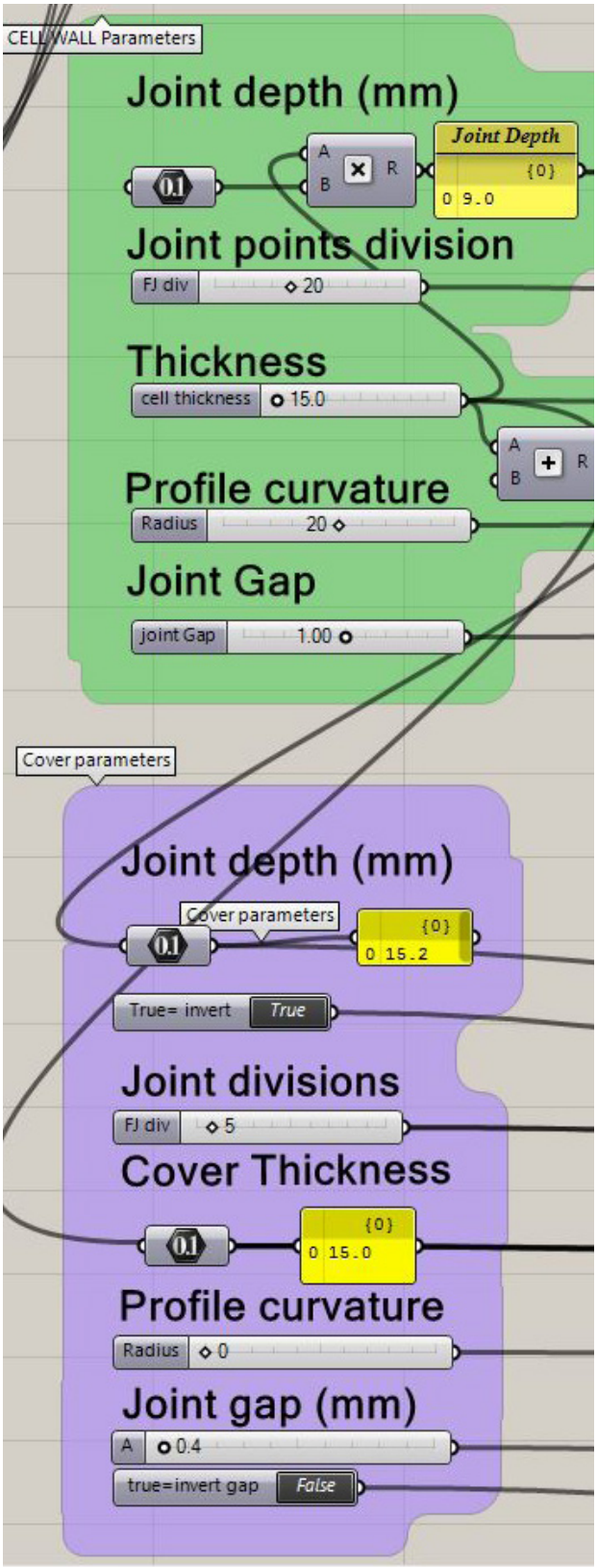
Un exemple de problème identifié et résolu par la suite est la méthode de construction de la géométrie des assemblages, qui consistait à découper puis assembler des surfaces entre elles pour former des volumes (fig. 1). Indépendemment, ces opérations étaient relativement rapides, mais l'enchaînement des opérations nécessaires ensuite pour reconstituer des objets solides et le nombre très important d'erreurs générées m'ont amené à utiliser plutôt des opérations booléennes de soustraction, a priori plus lourdes mais qui sont plus fiables et permettent de traiter une plus grande diversité de géométrie. Ce gain d'efficacité a finalement permis de gagner du temps de calcul.

Voici les principaux besoins et objectifs de ce travail :

- Améliorer la fiabilité :
    - Moins de bugs
    - Plus rapide
    - Fonctionner sur une plus grande diversité de géométrie de caissons
- Augmenter et améliorer les paramètres disponibles (fig. 2)
- Permettre de traiter une liste de caissons les uns après les autres
- Inscrire le programme dans un continuum numérique qui va de la conception à la fabrication
- Permettre à l'outil de prendre une forme plus universelle, à l'image d'un plugin Grasshopper.



1. Construction des assemblages par addition de surfaces.



2. Paramètres disponibles

2. DESCRIPTION DE L'ALGORITHME

a. Pré-requis

1. 2. 3.

Exemple de géométrie attendue en entrée

GÉOMÉTRIE

La géométrie nécessaire au fonctionnement de l'algorithme de génération des assemblages se compose de trois éléments :

1. Le cadre de la cellule
2. Les arêtes du cadre dont l'assemblage est à traiter
3. Les fonds

De manière générale :

- Toutes les faces sont plates et de type «Brep»
- Les arêtes de toutes les faces sont des lignes
- Le cadre est fermé, et les faces qui le constituent sont des quadrilatères

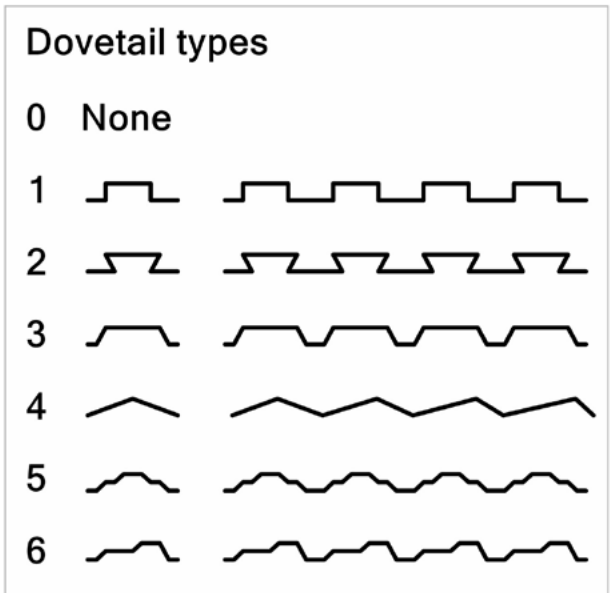
PARAMÈTRES

Les paramètres (fig. 4) permettent de régler :

- l'épaisseur (constante) des panneaux constituants la cellule
- le type d'assemblage parmi une liste proposée (fig. 5). Il est possible d'inverser le sens et de choisir qu'une arête soit traitée différemment (notamment pour des problèmes de montage)
- la largeur de l'assemblage
- la profondeur de l'assemblage
- le degré de courbure qui arrondi le profil d'assemblage
- l'épaisseur du jeu entre panneaux

Tous ces paramètres sont réglables de manière indépendante pour le cadre et les fonds.

4. Paramètres disponibles



5. Types d'assemblages disponibles

b. Fonctionnement général

Le résultat attendu à la sortie de cet algorithme est une modélisation 3D des différentes cellules, qui précise leur réalité constructive car elle sera ensuite utilisée pour la simulation puis l'usinage des éléments par soustraction de matière. On cherche donc à générer l'épaisseur et la géométrie des bords de chaque panneaux constituant une face de la cellule.

La géométrie de départ est constituée de surfaces et de lignes qui définissent (fig. 1):

1. - le cadre
2. - les arêtes entre chaque face du cadre
3. - les fonds (optionnel)

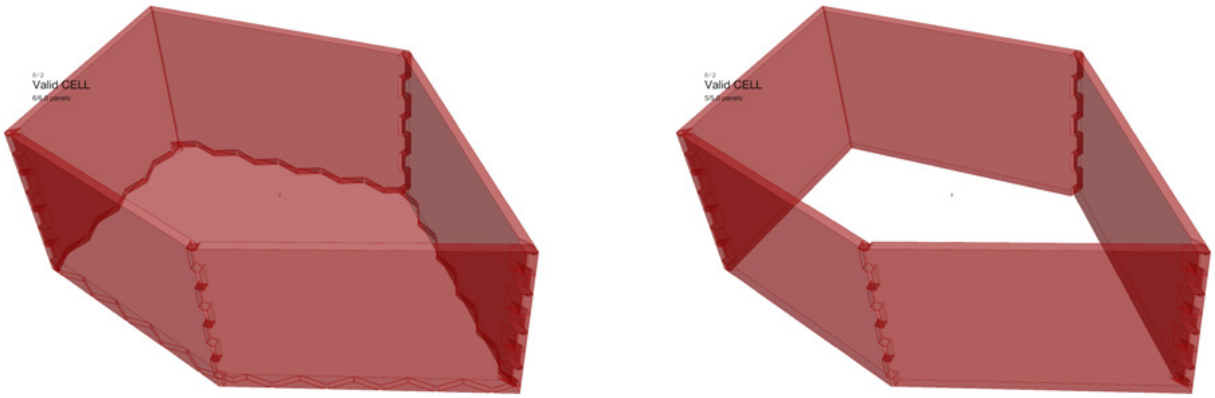
A partir de ces éléments l'algorithme construit trois types d'objets solides :

1. - Le cadre avec son épaisseur définitive
2. - Les assemblages qui vont servir d'emporte-pièce pour découper le cadre en panneaux
3. - Les fonds avec leurs épaisseurs définitives et leurs bords définitifs

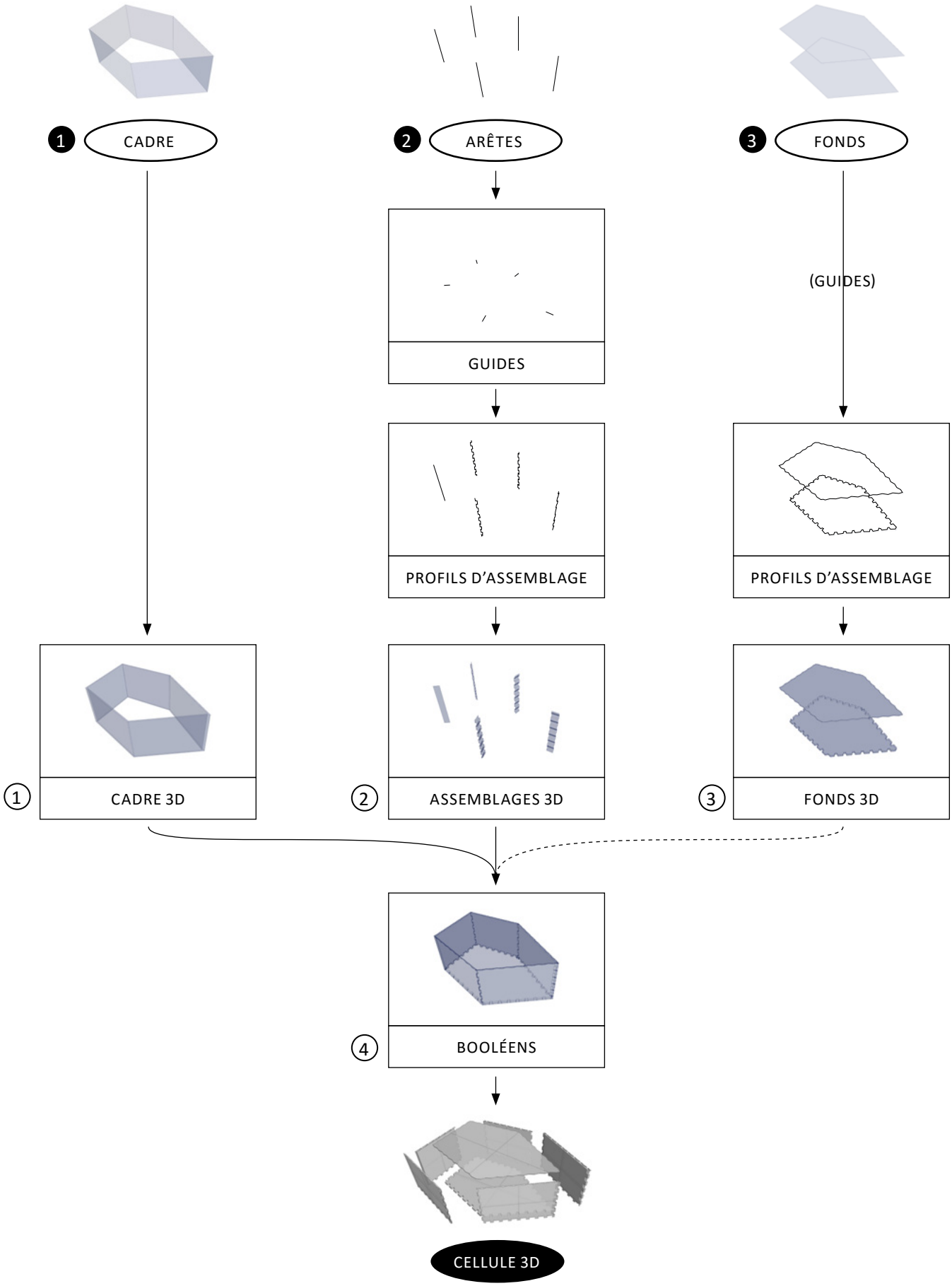
4. Enfin, les assemblages et les fonds sont soustraits du cadre, permettant d'obtenir le modèle 3D final.

Les fonds permettent d'obtenir une cellule fermée, mais leur géométrie n'est pas nécessaire au fonctionnement du programme. S'il n'y a pas de fond, le programme n'en tiendra pas compte et ne construira pas la face correspondante (fig. 2).

Cette méthode permet de laisser une grande liberté à l'utilisateur et s'inscrit dans la philosophie du plugin Grasshopper. L'utilisateur manipule directement l'ensemble de la géométrie des fonds en créant son propre algorithme et ses propres règles de sélection, avant de générer les assemblages.



2. Cellule identique à celle présentée en fig. 1, générée avec un seul fond à gauche et sans fond à droite.



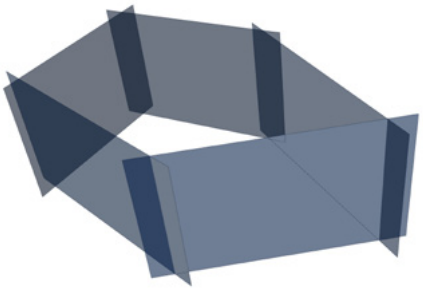
1. Le principe de fonctionnement du programme, qui s'appuie sur des soustractions d'objets solides.

① *Construction de l'épaisseur du cadre*

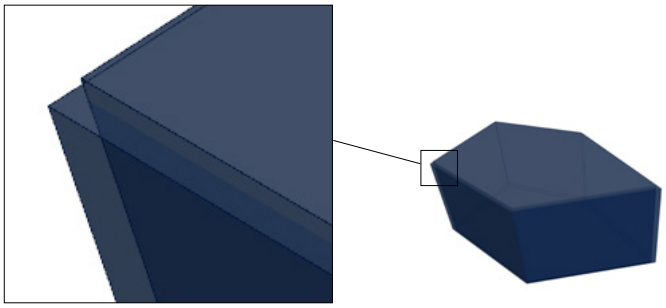
1. Principe de fonctionnement

Pour générer l'épaisseur du cadre, le programme décale chaque face vers l'intérieur de la cellule, trouve leurs intersections et reconstitue les futures faces intérieures. Le volume fermé des faces intérieures est ensuite soustrait du volume fermé des faces extérieures afin d'obtenir un seul solide (fig. 1).

Pour prendre en compte la plus grande variété possible de cellules notamment celles qui seraient évasées, les faces intérieures sont prolongées pour faciliter la recherche de leurs intersections (fig. 2). Elles sont ensuite recoupées en fonction des plans de chaque face afin de retirer les chutes. Le composant qui permet cette opération a été programmé par Oskar à l'aide du plugin GhPython. Il permet d'éviter de nombreux bugs mais contraint l'utilisateur à travailler avec des faces planes. De la même manière, le volume intérieur est légèrement prolongé afin de faciliter sa soustraction avec le volume extérieur, ce qui permet d'éviter de nombreuses erreurs (fig. 3).



2. Les faces sont prolongées dans leur plan.



3. Le volume intérieur dépasse légèrement du volume extérieur.

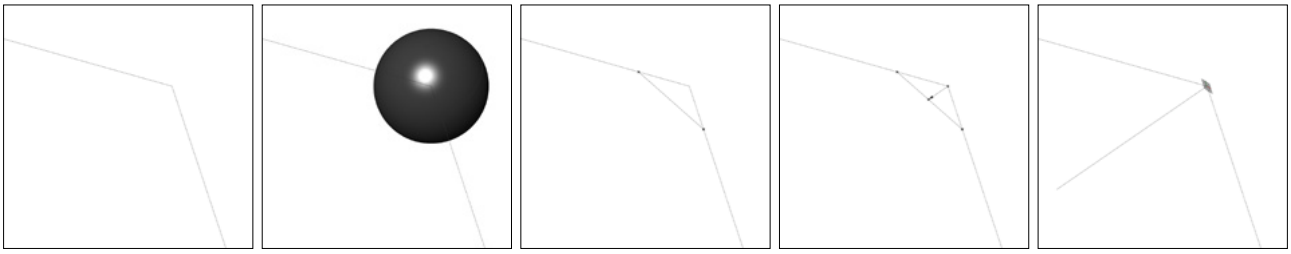
Cette méthode permet de résoudre les erreurs rencontrées auparavant, cependant elle reste relativement lente et gourmande en ressources, et limite en grande partie l'exploitation du programme avec d'autres géométries car elle est restreinte à des faces plates et des cellules fermées.

Une méthode plus rapide et plus efficace que je n'ai pas eu le temps de mettre en place ici consisterait à déplacer directement les sommets du cadre vers l'intérieur, de la même manière que l'on extrude un maillage vers l'intérieur, en utilisant comme guides les normales des faces. De cette manière d'autres géométries pourraient être traitées par le programme.

2 Construction des profils d'assemblage

GUIDES

Pour commencer, les guides sont créés à partir du contour des fonds. L'algorithme construit la bissectrice à chaque angle, ce qui permet d'obtenir la ligne guide et le plan de chaque futur profil d'assemblage (fig 1.).



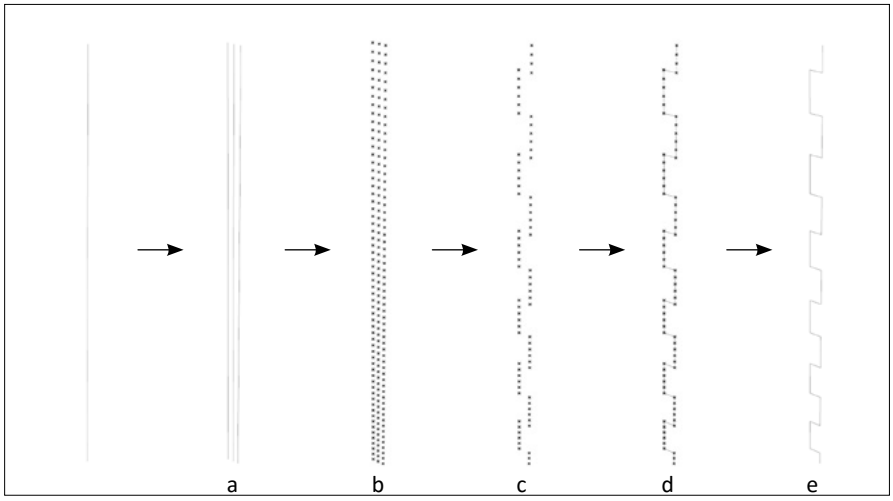
1. Les différentes étapes qui permettent la création des guides.

PROFILS D'ASSEMBLAGE

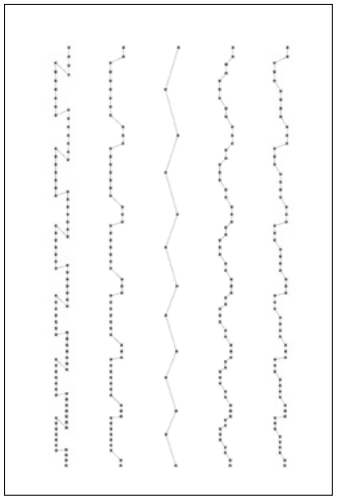
Pour créer les polylignes des profils, chaque arête du cadre est décalée[2] sur ses deux faces adjacentes (fig. 2 a). Les deux lignes obtenues ainsi que l'arête d'origine sont ensuite divisées par un nombre égal de points (fig. 2 b). Un filtre est ensuite appliqué sur les points de chaque ligne afin de ne conserver que les points souhaités (fig. 2 c). Ces points sont enfin connectés (fig. 2 d) pour former la polyligne du profil d'assemblage (fig. 2 e).

Cette méthode permet de construire une diversité de profils en jouant sur le rythme des points (fig. 3). Des profils supplémentaires peuvent bien sûr être ajoutés (fig. 4), et l'on peut également imaginer construire des lignes supplémentaires lors de la première étape (fig. 2 a) si l'on souhaite construire des profils plus complexes.

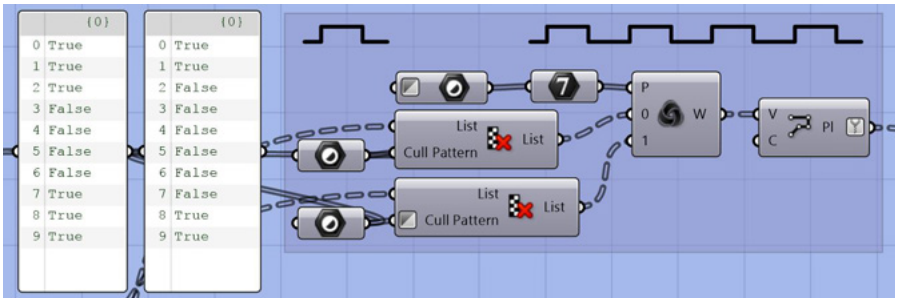
2 Outil «Offset» dans Grasshopper



2. Création des profils d'assemblage à partir d'un réseau de points.



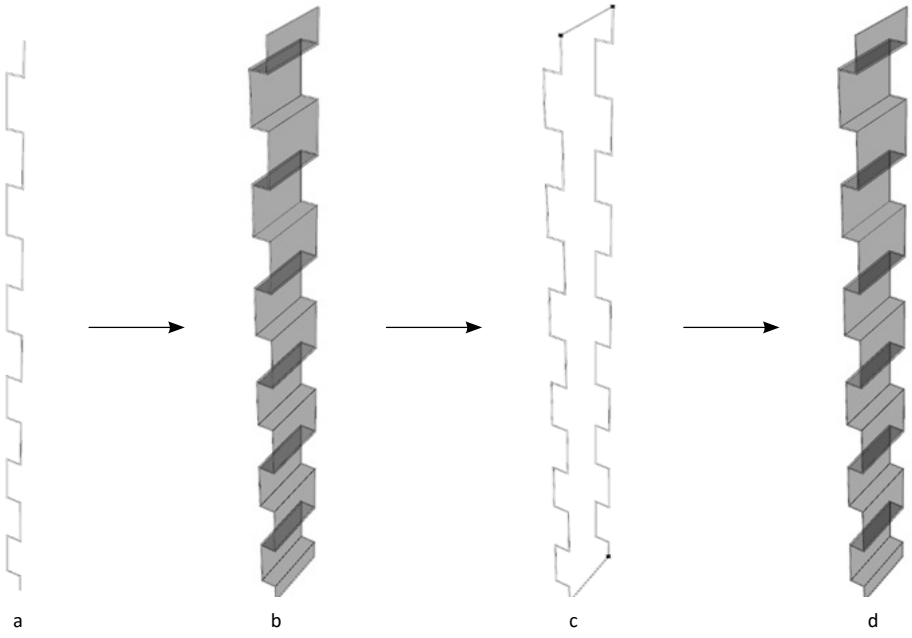
3. Exemples de profils possibles.



4. Exemple des paramètres d'un profil. Les deux listes à gauche donnent le rythme des points à conserver ou non sur les deux lignes extérieures.

ASSEMBLAGES 3D

Pour construire les profils en 3D, la polyligne précédemment obtenue est d'abord décalée de l'épaisseur de jeu souhaitée entre les pièces (fig. 5 a). C'est à cette étape que la complexité des polylignes se heurte parfois aux possibilités offertes par les outils disponibles dans Grasshopper.



5. Construction du profil d'assemblage en 3D à partir d'une polyligne.

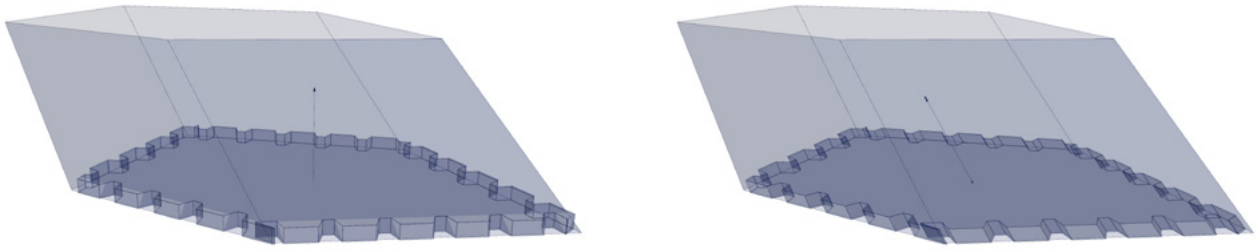
En effet, le plus simple serait ici de construire la surface entre les deux lignes décalées puis de l'extruder afin d'obtenir l'assemblage en trois dimensions. Cependant le décalage des polylignes (outil «offset) produit régulièrement des erreurs, tout comme l'outil de construction d'une surface entre deux polylignes. On peut également noter qu'il est possible de «lisser» la polyligne avant le décalage en créant des congés dans les angles, ce qui produit alors une courbe qui est une source supplémentaire d'erreurs. Cette étape serait certainement à retravailler dans le cadre d'un travail ultérieur sur le sujet, peut-être en passant par la programmation d'un nouvel outil permettant cette opération. C'est pourquoi la construction du profil d'assemblage suit une stratégie légèrement différente pour minimiser les erreurs trop fréquentes.

Suite au décalage, ces deux polylignes sont donc ensuite extrudées suivant le guide du profil correspondant (fig. 5 b). On construit les surfaces supérieures et inférieures à partir des arêtes créées lors de l'extrusion (fig. 5 c). Toutes ces surfaces sont soudées entre elles et le profil est rendu solide grâce au composant «cap holes» de Grasshopper (fig. 5 d)

GUIDE

Sur certaines cellules, les fonds ne sont pas perpendiculaires au cadre. Il est donc nécessaire que les fonds ne soient pas extrudés par rapport à leur normale mais par rapport à un guide qui suit l'inclinaison du cadre, afin d'éviter des erreurs par la suite lors de la soustraction des volumes. Pour construire les guides, on relie le centre de gravité de la face du fond au centre de gravité du volume de la cellule (fig. 1).

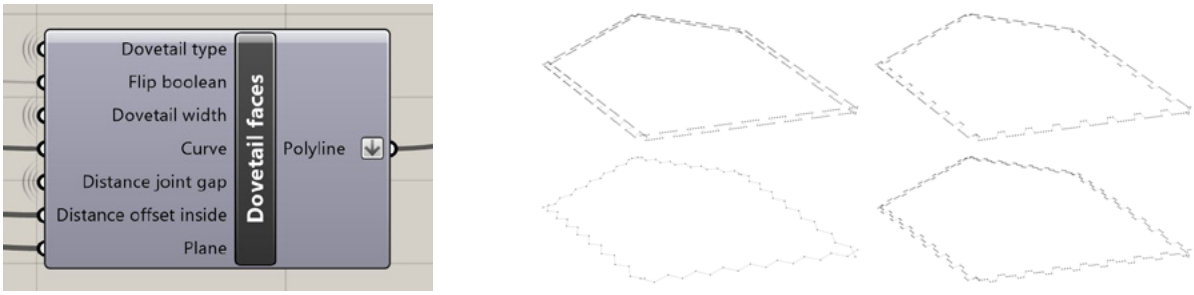
Cette méthode simple ne fonctionne plus dès lors que les cellules sont évasées. Cette étape de construction devrait donc être envisagée d'une autre manière dans le cadre d'un travail ultérieur visant à élargir les possibilités du programme.



1. A gauche le guide est la normale de la face. A droite le guide est obtenu en reliant le centre de la face au centre de la cellule

PROFILS D'ASSEMBLAGE

Les polylignes des profils d'assemblages des fonds sont générées de la même manière que pour les arêtes, voir pages précédentes (fig. 2).



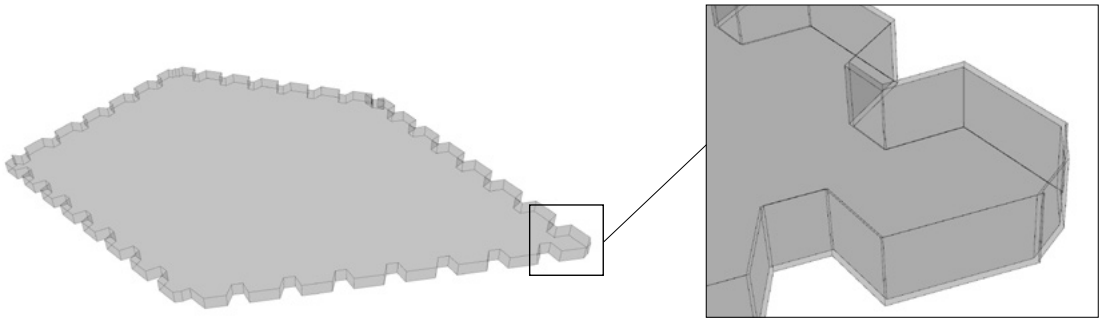
2. A gauche, le même cluster des profils d'assemblage est utilisé pour les arêtes et les fonds. A droite, exemples des profils possibles.

FONDS 3D

Une fois les polylignes obtenues, elles sont ensuite prolongées jusqu'à leur intersection (fig. 3 a). La surface du fond est découpée en fonction de ces polylignes (fig. 3 b) puis les chutes sont retirées (fig. 3 c). La surface obtenue est ensuite extrudée en suivant la direction du guide.

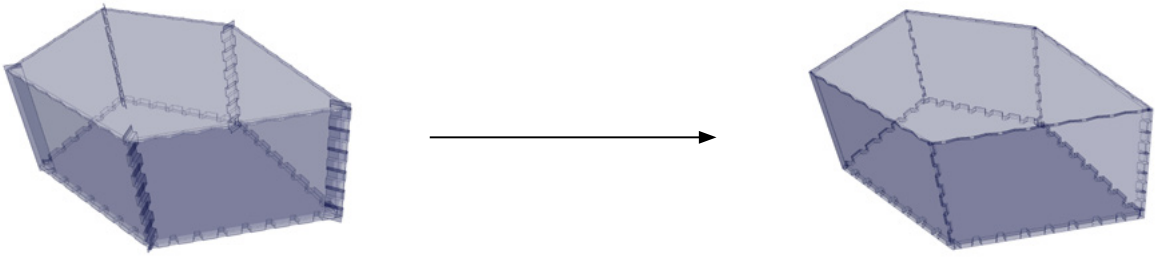
3. Les différentes étapes des polylignes jusqu'à la surface.

Pour assurer un jeu entre les pièces lors de l'assemblage, un second modèle 3D du fond est généré. Il est agrandit de la taille du jeu et servira d'emporte-pièce lors de la soustraction des volumes (fig. 4).



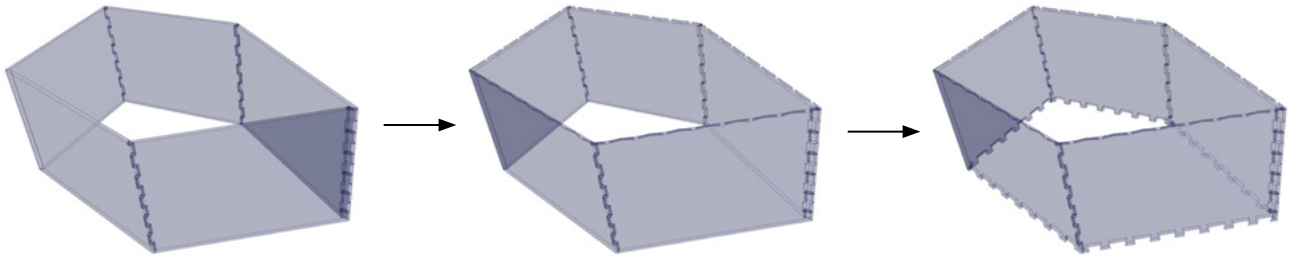
4. En superposition, on distingue le modèle agrandit de la taille du jeu qui servira pour l'opération booléenne.

4 *Opérations booléennes de soustraction*



Lors de cette dernière étape, les trois types d'éléments construits précédemment sont soustraits. Il est nécessaire de donner un ordre dans la soustraction (fig. 1) :

- Les assemblages 3D sont soustraits du cadre 3D
- Le fond supérieur (avec le jeu) est soustrait du résultat de l'opération précédente
- Le fond inférieur (avec le jeu) est soustrait du résultat de l'opération précédente



1. Les soustractions successives : les assemblages 3D sont soustraits du cadre, puis le fond supérieur et inférieur.

Enfin, les fonds définitifs sont ajoutés au modèle qui est alors complet. Cette opération peut sembler simple mais se révèle lourde et longue en calcul et amène souvent à des erreurs. De nombreux trous apparaissent sans raison apparente ainsi que des petites pièces de chutes qui sont supprimées. Pour obtenir des solides valides, il est bien souvent nécessaire de déconstruire puis reconstruire chaque panneau en supprimant des surfaces invalides apparues lors de la soustraction. Ces erreurs sont directement dues à l'outil de soustraction de Grasshopper qui n'est peut-être pas complètement optimisé. Ces opérations de reconstructions ont été intégrées à l'algorithme pour essayer de réparer ces erreurs fréquentes. Cependant, elles alourdissent encore le programme et ne sont pas tout le temps suffisantes.

Les opérations booléennes permettent une approche intuitive et à priori «propre». En réalité, il serait préférable de repenser une partie du programme afin de minimiser leur usage voir de s'en passer complètement car elles ne sont pas stables et sont très gourmandes en calcul. On pourrait par exemple imaginer une approche face par face qui générerait directement les assemblages en divisant les arêtes en points puis en déplaçant ces points selon des vecteurs à leur emplacement définitif. Une autre approche pourrait être de générer d'abord chaque panneau qui constitue la cellule en laissant vide les nœuds d'assemblage, puis d'ajouter une solution technique d'assemblage : ajout de matière sur les bords des panneaux ou construction d'une pièce supplémentaire spécifique pour l'angle.

c. Les « Outils »

Après avoir vu le fonctionnement de l'algorithme de construction des assemblages d'une cellule, voici maintenant les différents outils que j'ai ajouté au programme afin d'automatiser et de simplifier son utilisation pour permettre le traitement d'une grande quantité de données, par exemple lors de la conception d'une paroi constituée de plusieurs dizaines voir centaines de cellules à la géométrie non standard. Dans sa version originale, Grasshopper ne permet pas d'effectuer de calculs en utilisant des boucles. Cette fonction peut cependant être ajoutée en utilisant un composant de programmation tel que VB, C ou GhPython, ou encore à l'aide d'un plugin comme Anemone qui est utilisé dans les outils «List Loop» et «Debug Loop».

TEST RESULTS

Cet outil teste si la géométrie générée par l'algorithme est valide par comparaison du nombre d'éléments générés par rapport au nombre de panneaux attendus, et en vérifiant que les éléments générés sont solides. Le programme vérifie également le nombre et la nature des opérateurs (assemblages 3D et fonds) avant l'étape de l'opération booléenne afin que l'utilisateur puisse déterminer plus facilement d'où viennent les erreurs lorsqu'elles apparaissent. Le résultat «valide / invalide» de ces tests est utilisé par d'autres fonctions car il permet au programme de «comprendre» qu'il y a une erreur.

DISPLAY

Ce composant regroupe les fonctions qui permettent l'affichage et la visualisation du modèle dans Rhinocéros à l'aide du plugin Human (fig. 1 et 2) :

Dans l'espace 3D :

- Géométrie active en rouge
- Géométrie active non valide en jaune
- Contour de toutes les cellules de la liste en bleu
- ID des cellules en leur centre

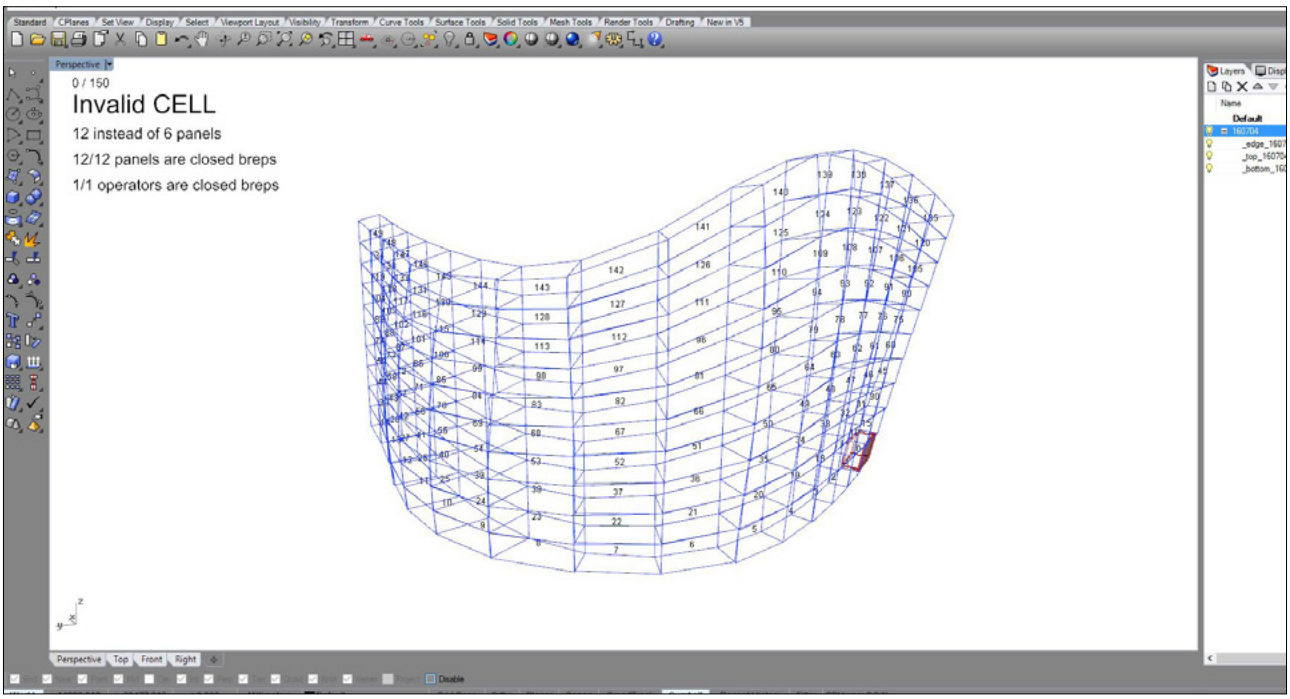
Au premier plan à l'écran en haut de la fenêtre 3D :

- ID de la cellule active et le nombre de cellules total
- Les résultats de Test Results : validité de la cellule, nombre de panneaux/opérateurs et nature

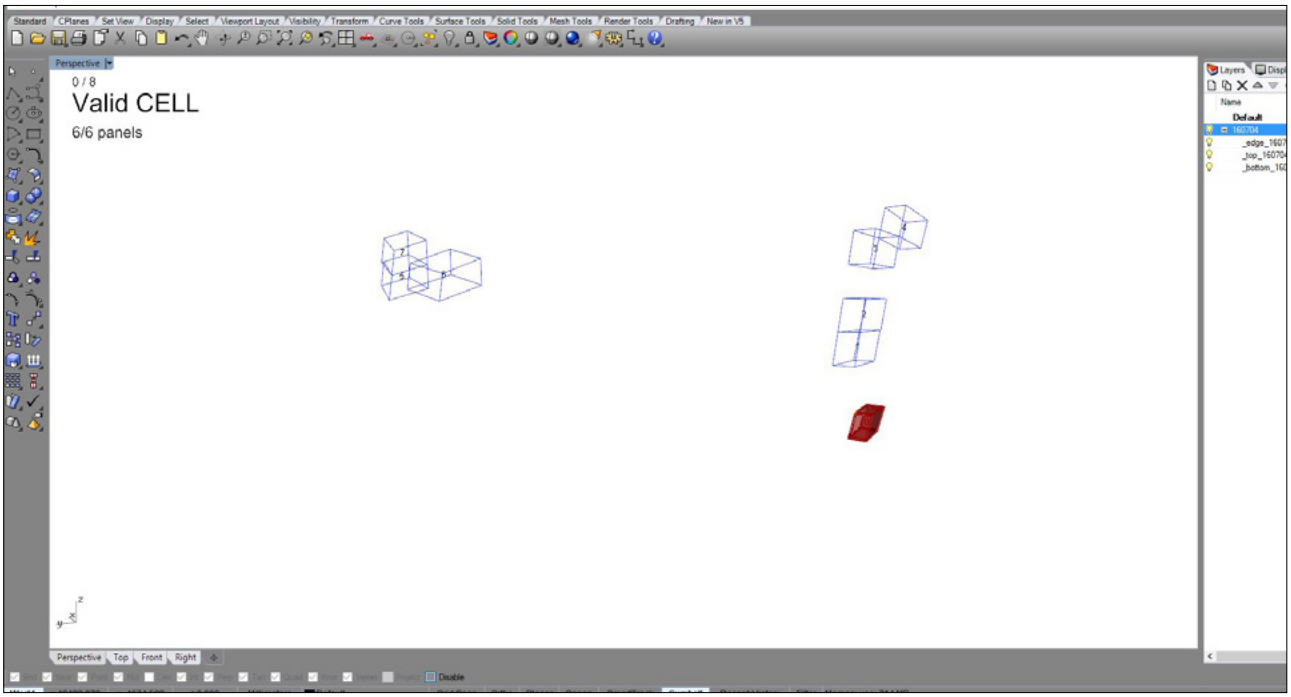
LIST LOOP

Cet outil permet de générer une série de cellules les unes après les autres car l'algorithme de génération des assemblages ne peut traiter qu'une cellule à la fois (fig. 3). L'utilisateur peut choisir un fonctionnement manuel (cellule active seulement), total (toutes les cellules, fig. 1) ou partiel en indiquant les ID de plusieurs cellules à traiter (fig. 2). Cette dernière possibilité est intéressante pour relancer un calcul sur plusieurs cellules qui présenteraient des erreurs, ou sur lesquels l'utilisateur souhaiterait appliquer des paramètres différents.

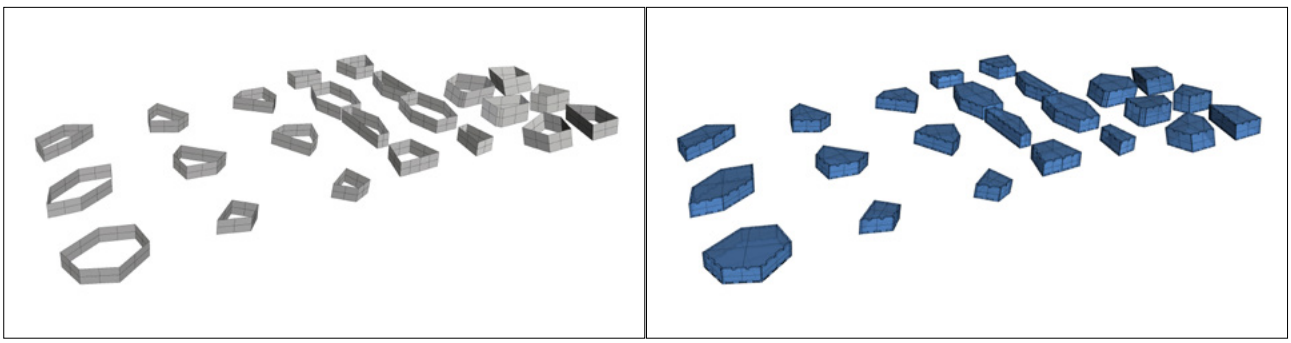
Cette fonction apporte une flexibilité dans le traitement en série et pourrait être associée avec un algorithme de sélection des cellules. Des sous-groupes seraient formés selon un motif ou des conditions (ouverture, ensoleillement, comportement thermique ou structurel…) et les cellules correspondantes pourraient être générés avec des paramètres différents d'un groupe à l'autre. On pourrait par exemple imaginer que l'épaisseur des panneaux évolue en fonction de leur sollicitation structurelle ou la nature des assemblages en fonction d'un motif ou d'un usage à l'échelle de la paroi.



1. Visualisation du modèle et de la validité de la cellule active dans Rhinocéros.



2. Modèle identique à la fig. 1, mais seulement 8 cellules sont sélectionnées pour le traitement en série.



3. Une série de cellule (à gauche) traitée automatiquement les unes à la suite des autres avec les mêmes paramètres (à droite).

DEBUG LOOP

Cette fonction permet de résoudre automatiquement les erreurs les plus courantes qui ont lieu lors de la génération des profils : des intersections entre profils impossibles et / ou le composant qui permet de donner une courbure à la polyligne qui ne fonctionnent pas. La boucle inverse donc tour à tour le sens des polylignes puis modifie leur rayon de courbure jusqu'à ce que la cellule soit valide.

Le traitement en série vu précédemment est suspendu lorsqu'une cellule est invalide puis reprend lorsque les erreurs sont corrigées. Afin d'éviter des bugs du programme, le nombre de corrections successives est limité. L'utilisateur peut choisir que le programme soit suspendu jusqu'à une intervention manuelle (fig. 4) ou qu'il passe automatiquement à la cellule suivante sans conserver la géométrie lorsque le nombre de corrections maximum est atteint sans solution valide.

BAKE

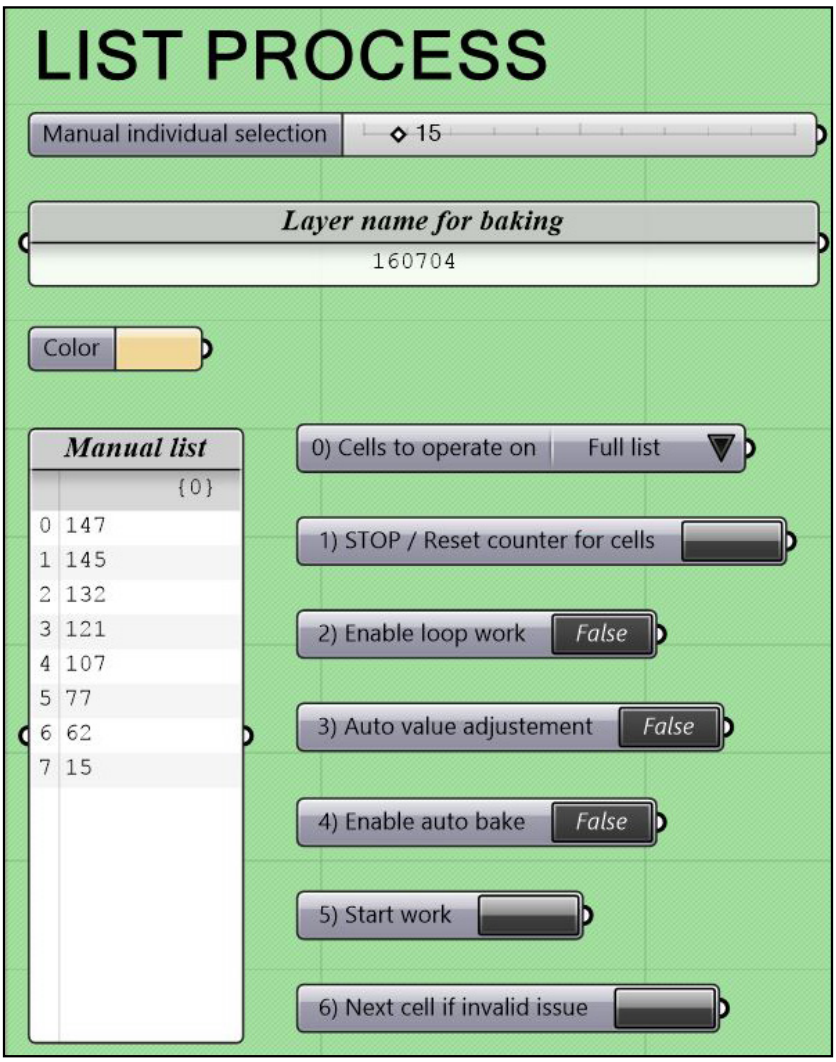
Cette fonction s'active à la fin du calcul de chaque cellule lorsque celle ci est indiquée comme valide par Test Results. Elle permet d'enregistrer automatiquement dans Rhinocéros la géométrie générée dans Grasshopper en lui attribuant des caractéristiques. L'utilisateur donne un nom de calque et une couleur pour la série de cellules qu'il s'apprête à calculer. La fonction Bake créé le calque s'il n'existe pas et établit automatiquement une nomenclature. Elle attribue un identifiant unique ainsi qu'une couleur aux éléments créés qui sont :

- triés dans des sous-calques par type
- nommés indépendamment par identifiant et type (fig. 5)
- groupés par cellule

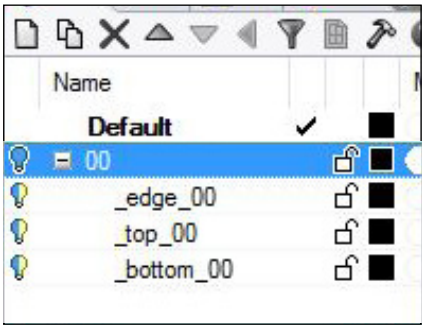
Cette identification facilite la manipulation dans Rhinocéros et permet de transférer la géométrie d'un algorithme à l'autre sans perdre la hiérarchie entre la paroi, les cellules et les panneaux qui constituent chaque cellule en s'inscrivant dans la logique de Grasshopper qui trie ses données par « arbre ».

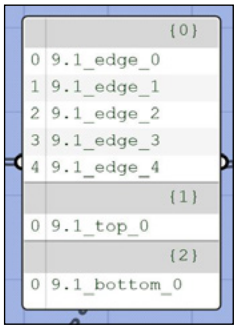
CHRONOMÈTRE

Enfin, un chronomètre qui affiche la différence entre l'heure de début du calcul et l'heure de fin permet de mesurer la durée totale de la série. Le programme reste relativement lent puisqu'à titre d'exemple, le calcul d'une cellule dure entre 5 et 30 secondes. La série de 150 cellules a duré environ 45 minutes, ce qui fait une moyenne d'environ 18 secondes par cellule. Ce temps inclut les itérations multiples d'une même cellule lorsque une erreur apparait. Dans sa globalité l'algorithme de génération des assemblages d'une cellule peut dont être considéré comme lent si l'on souhaite s'orienter vers une manipulation en temps réel d'une ou des cellules, mais il reste cependant très rapide et utile si l'on considère le nombre de tâches répétitives et complexes qui devraient être réalisées manuellement.



4. Interface qui permet d'activer et de paramétrer les outils. Le bouton 6) permet de passer à la cellule suivante manuellement.

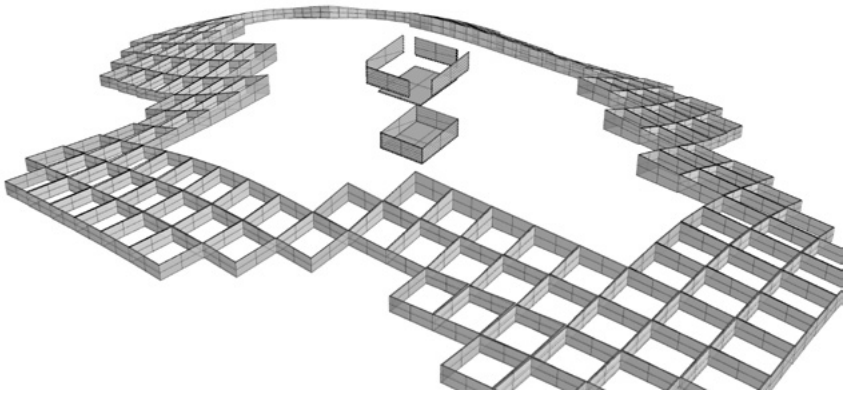




5. A gauche, nomenclature par type de géométrie dans le calque «00». A droite, identifiants générés pour les différents éléments de la cellule 9 qui appartient au groupe 1.

3. CONCLUSION

Les principaux objectifs de ce travail sur la modélisation paramétrique d'assemblages en bois ont été atteint. En effet, dans son état actuel le programme fonctionne dans le cadre précis du travail entamé par Oskar. La fiabilité, la rapidité et les paramètres disponibles ont été améliorés et il est désormais possible de générer une série de cellules automatiquement. De plus, l'algorithme s'inscrit dans un continuum numérique qui va de l'algorithme de tesselation d'une surface présenté précédemment à la simulation des trajectoires d'usinage de ses éléments constructifs qui est présenté dans les pages suivantes. Le programme accepte aussi des géométries générées de manière indépendante, comme a pu le montrer un test effectué à partir de la modélisation du PFE de Guillaume (fig. 1). Cependant, un travail supplémentaire de synthèse sera encore nécessaire sur le programme pour arriver à lui donner une forme de plugin pour Grasshopper. Il sera alors primordial de se demander quelles fonctions doivent être regroupées dans un même composant et quelles étapes doivent être séparées, de la même manière qu'il faudra certainement trouver d'autres stratégies de modélisation pour pouvoir gérer des géométries qui ne se limitent pas à des cellules fermées dont toutes les faces sont planes.



1. Génération des assemblages d'une structure cellulaire à partir d'une géométrie extérieure.

Ce travail permet aujourd'hui de faire le point sur ce qui fonctionne, ce qui a encore besoin d'être amélioré et propose une stratégie de modélisation qui sera peut-être remis en cause selon les objectifs que l'on souhaite atteindre. Par exemple, l'utilisation principale des opérations booléennes a permis de réduire de manière importante le nombre d'erreurs et d'augmenter la rapidité du programme. Cependant à l'utilisation on se rend compte que cette stratégie est loin d'être la plus rapide et la plus efficace et qu'il serait préférable de s'en passer si l'on souhaite encore alléger l'algorithme, la rapidité de chaque opération devenant un élément clé lorsqu'on traite de grandes quantités de données. Le continuum numérique tend aussi à ce que chaque étape soit la plus rapide possible afin de conserver une fluidité et une réactivité dans les modifications d'un bout à l'autre de la chaîne, sans quoi l'outil perd son effet moteur et stimulant pour la conception. Plutôt que d'utiliser des opérations booléennes pour créer les assemblages, on peut par exemple imaginer diviser les bords des panneaux de chaque cellule puis déplacer certains sommets afin de venir «sculpter» directement les assemblages sur les panneaux.

Une autre piste pour améliorer la rapidité et l'efficacité du programme se trouve dans l'utilisation des fenêtres de scripts dans Grasshopper qui permettent de coder en VB, C, Python… Cette méthode permet tout d'abord d'utiliser des fonctions qui ne sont actuellement pas accessible dans l'interface graphique de

Grasshopper, notamment des fonctions qui existent dans Rhinocéros. Ensuite, les fonctions développées dans ce travail font souvent appel à des plugins. Le codage permettrait de s'en passer dans de nombreux cas, surtout concernant les «outils» qui utilisent des boucles, des opérations logiques, et des fonctions de recherche. Ces opérations sont bien plus efficaces sous forme de code qu'en utilisant les composants de Grasshopper qui restent incomplets.

Le codage peut également permettre de créer une base de données alternative à celle utilisée par Grasshopper. Cela pourrait permettre d'utiliser une stratégie de modélisation moins lourde en calculs, qui ne reconstruit pas systématiquement toute la géométrie comme le fait Grasshopper mais qui fait appel à des modèles déjà construits qui seraient simplement modifiés et adaptés, à la manière d'un cube que l'on viendrait déformer pour qu'il corresponde à certaines proportions plutôt que de le construire à partir de rien.

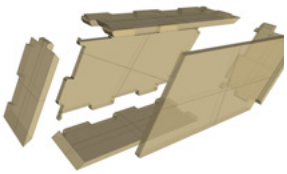
D'autre part, la question de l'assemblage des cellules entre elles n'a pas été abordée dans ce travail mais aurait certainement des conséquences sur la stratégie de modélisation des cellules dans leurs dimensions constructives.

Enfin, les assemblages réalisés par cet algorithme s'inspirent mais ne correspondent pas réellement à des assemblages à queue droite ou à queue d'aronde. Dans une perspective d'amélioration du programme, il y aurait donc d'abord un travail d'analyse et de comparaison des assemblages possibles et souhaitables pour se rapprocher d'une réalité constructive plus concrète qui tirerai parti du potentiel du mode de conception et de réalisation de cette architecture à l'aide d'outils numériques. On peut par exemple imaginer des assemblages autobloquants, à clavettes ou qui utilisent le tissage et qui ne nécessitent ni colle ni vis. Il existe beaucoup d'assemblages qui répondent souvent à des problématiques précises et qui dépendent autant des moyens dont on dispose que de la destination de ce que l'on construit : est-ce que cela doit être facilement montable ? démontable ? Les assemblages doivent-ils être rigides ou souples ? visibles ou invisibles ? Cela va-t-il être réalisé manuellement ou à l'aide d'une machine à commande numérique ? Cette problématique qui peut paraître à première vue anodine joue pourtant un rôle important dans l'esthétique de ce qui sera finalement construit et trouverai donc parfaitement sa place dans une démarche de conception qui s'appuie sur le continuum numérique (fig. 2).

2. Deux exemples d'assemblages innovants réalisés à l'aide d'une machine à commande numérique. Le premier est invisible de l'extérieur tandis que le second, très facilement montable et démontable, impacte beaucoup plus l'esthétique de l'objet.
Images : «CNC Cut Wood Joinery» disponible sur mkmra2.blogspot.fr

IV. PRÉPARATION À LA FABRICATION NUMÉRIQUE

1. INTRODUCTION



1. Calepinage automatisé

2. Simulation d'une trajectoire d'usinage

Comme nous l'avons vu précédemment, il est question de fabriquer des parois constituées de dizaines voir centaines de cellules, qui sont elles même constituées d'au moins trois panneaux mais souvent plus et dont la géométrie est dite non standard, ce qui signifie complexe et non répétitive d'une pièce à l'autre. Le travail de préparation d'un tel projet en vue de sa fabrication, même numérique, est donc long est fastidieux ce qui représente un coût de main d'œuvre élevé. En effet, dans l'industrie il est souvent nécessaire que quelqu'un redessine entièrement les pièces à usiner sur un logiciel spécifique à la machine à commande numérique qui va être utilisée. Durant cette étape du projet il n'est plus possible d'effectuer des modifications sans que cela ai un impact très important sur le temps et le coût de fabrication.

Afin de minimiser ces coûts et de permettre un maximum de souplesse dans la conception et la modification du projet, l'idée est ici de construire un algorithme qui s'inscrit dans la continuité du travail effectué précédemment et qui permette de générer automatiquement des fichiers d'usinages prêts à l'emploi suivant le principe du continuum numérique.

Pour y parvenir, le programme devra d'abord «démonter» numériquement la paroi modélisée en trois dimensions, calepiner les différentes pièces sur des panneaux de bois aux dimensions standardisées (fig. 1) puis construire les instructions qui devront être effectuées par la machine à commande numérique dans un format de fichier compatible (fig. 2).

L'enjeu est également d'utiliser le programme pour optimiser au maximum le calepinage des pièces afin de réduire les chutes et donc les coûts, et de simuler l'usinage afin de pouvoir anticiper des éventuels problèmes.

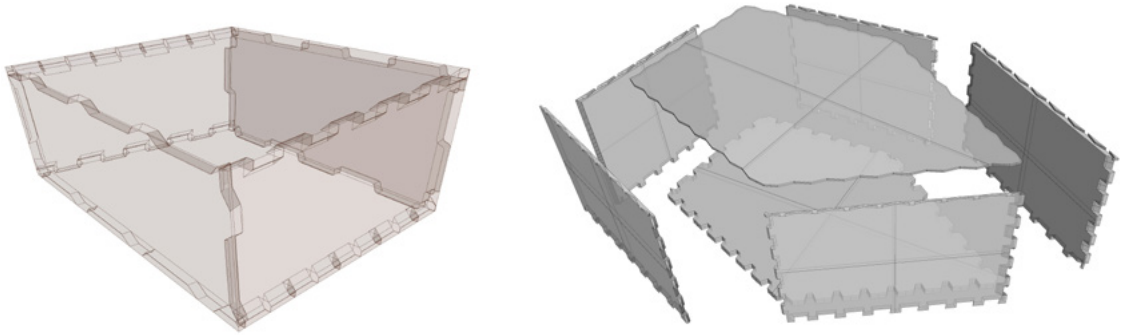
Actuellement, l'algorithme se décompose en deux parties :

1. Calepinage automatisé
2. Simulation d'une trajectoire d'usinage et export des instructions

Étant donné le temps très court de ce stage par rapport à l'ampleur du travail, il est évident que ces algorithmes ne sont actuellement qu'au stade d'esquisses. Ils sont pour l'instant loin d'être utilisables de manière fiable et précise mais permettent de comprendre quels seront les problèmes techniques à résoudre et constituent une première piste de réflexion.

2. CALEPINAGE AUTOMATISÉ

a. Pré-requis



1. Deux exemples de géométrie attendue en entrée

GÉOMÉTRIE

La géométrie traitée par l'algorithme de calepinage automatisé doit répondre aux critères suivants (fig. 1) :

- Chaque objet est un solide («Closed brep»)
- Chaque objet est référencé par un indentifiant unique (ID face) et par un identifiant qui correspond à la cellule auquel il appartient (ID cellule)

De manière générale :

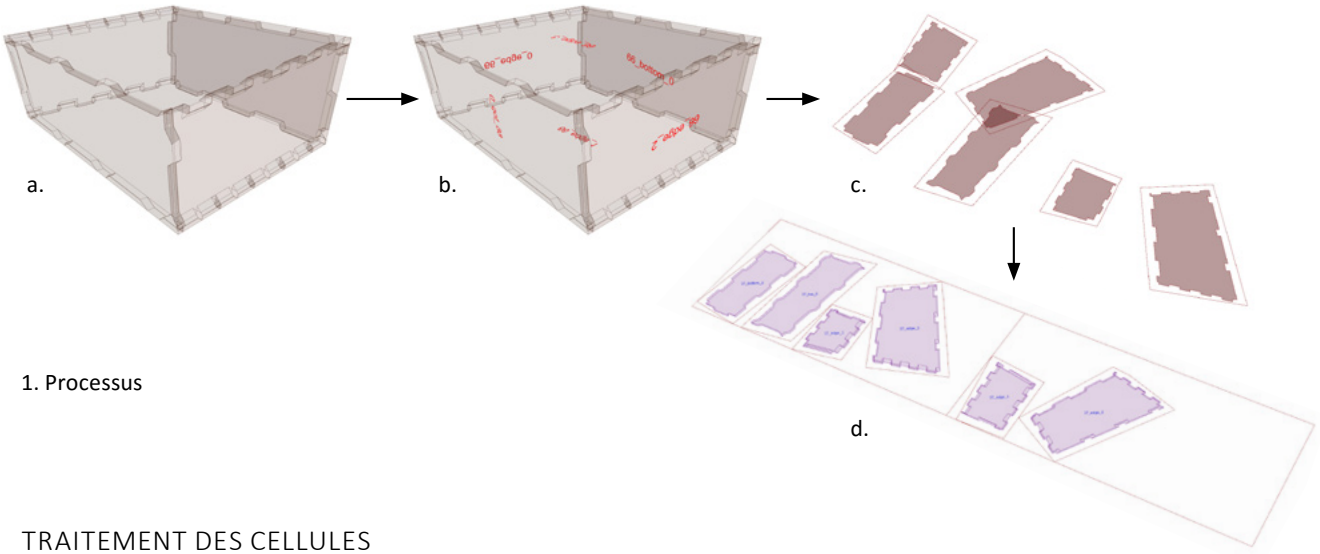
- Tous les objets correspondent à des panneaux en trois dimensions et leur épaisseur est constante

PARAMÈTRES

Les paramètres permettent de régler :

- les dimensions X et Y de la feuille
- la marge minimum entre les objets
- la qualité du calepinage

b. Fonctionnement général



1. Processus

TRAITEMENT DES CELLULES

Afin de concevoir l'algorithme sans être gêné par des ralentissements, le programme est prévu pour ne traiter pour l'instant qu'une seule cellule à la fois (fig. 1 a). Comme vu précédemment dans les outils de l'algorithme de modélisation des assemblages, il serait tout à fait possible dans le cadre d'un travail ultérieur d'utiliser ici la fonction permettant de traiter une série de cellules à la suite, cependant cela entraînerait une grande quantité de chutes car les objets déjà placés ne serais pas pris en compte d'une cellule à l'autre. Il faudrait alors traiter plusieurs voir toutes les cellules en même temps, ce qui risque d'être très long comme nous le verrons par la suite. Une autre solution serait de placer petit à petit les nouveaux objets en tenant compte des objets précédemment placés sur les panneaux.

IDENTIFICATION DES FACES

L'algorithme commence d'abord par lire les identifiants des objets actifs et les affiche sur le modèle 3D (fig. 1 b). Ces identifiants pourraient par la suite être gravés sur les objets lors de la fabrication et le modèle 3D annoté pourrait servir de plan de montage.

MISE À PLAT

La cellule est ensuite décomposée et les objets sont mis à plat dans le même plan que la future feuille de calepinage (fig. 1 c).

CONSTRUCTION DE L'EMPRISE

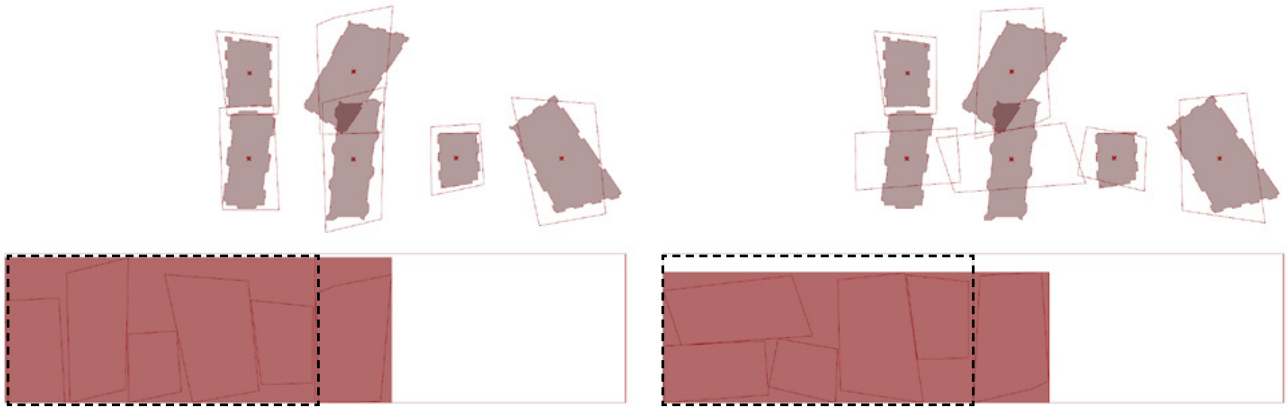
Une fois à plat, les objets sont décomposés en surfaces. La polyligne de contour de la plus grande surface est ensuite simplifiée à l'extrême, puis décalée vers l'extérieur de l'objet afin de laisser une marge tout autour de la pièce (fig. 1 c). C'est cette polyligne qui sera utilisée par le composant de calepinage automatique.

Le programme calcule ensuite le plan de référence de la polyligne, qui servira par la suite d'origine à l'objet pour le replacer correctement sur la feuille de calepinage en lieu et place de la polyligne (fig. 1 d).

CALEPINAGE

La fonction de calepinage automatique est assurée par le composant «Nesting» du plugin Generation. Ce composant permet de calepiner des courbes sur une feuille rectangulaire selon plusieurs qualités et en tenant compte d'une marge entre les courbes. Le composant créé autant de feuilles que nécessaire pour positionner toutes les courbes.

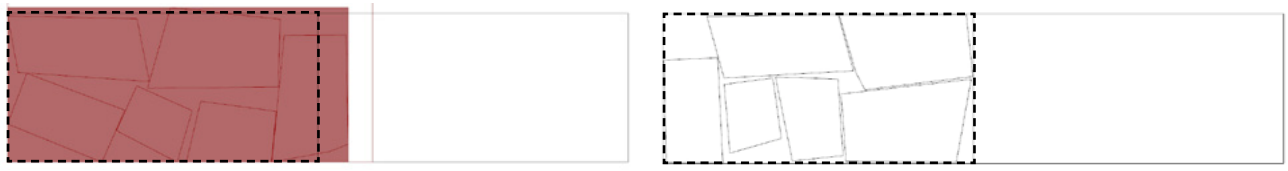
Le composant «Nesting» fonctionne, cependant il présente plusieurs défauts. Tout d'abord il est très lent et peut facilement calculer pendant plusieurs minutes. Le temps de calcul augmente beaucoup dès que l'on demande dans les paramètres une qualité de calepinage supérieure, même avec des courbes simples et peu nombreuses. Une autre caractéristique du composant est qu'il conserve au maximum l'orientation initiale des courbes lors du calepinage, ce qui peut être un avantage lorsque l'on traite par exemple des courbes qui ont une forme semblable, allongée et qui sont dans le même sens, mais qui s'avère être un défaut majeur lorsque les courbes sont toutes différentes et n'ont pas d'orientation prédéfinie. Les résultats s'améliorent lorsque la qualité est augmentée mais sont souvent loin d'être la solution la plus optimisée (fig. 2).



2. On remarque que le plugin Nesting conserve au maximum l'orientation initiale des objets lors du calepinage.

Pour essayer d'améliorer les résultats de Nesting et parce que les composants de calepinage automatique m'ont semblé rares et relativement complexes à mettre au point, j'ai choisi d'utiliser le composant Galapagos. Présent dans Grasshopper (ce n'est pas un plugin), c'est un Solver qui permet de tester de nombreuses possibilités automatiquement en cherchant à converger vers un résultat précis.

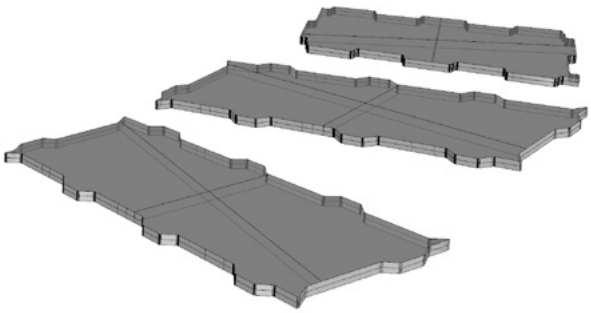
Galapagos est utilisé pour changer l'orientation des objets indépendamment les uns des autres avant qu'ils ne soient transmis à Nesting pour calculer leur calepinage. Une fois le calepinage effectué, la surface de l'emprise des courbes sur la feuille est calculée (en rouge sur les figures 2 et 3). Galapagos est programmé pour retenir la solution qui permet que la surface d'emprise obtenue soit la plus petite possible. Le résultat obtenu de cette manière est plus optimisé mais dépasse encore de la feuille et reste beaucoup plus long qu'un calepinage effectué manuellement (fig. 3).



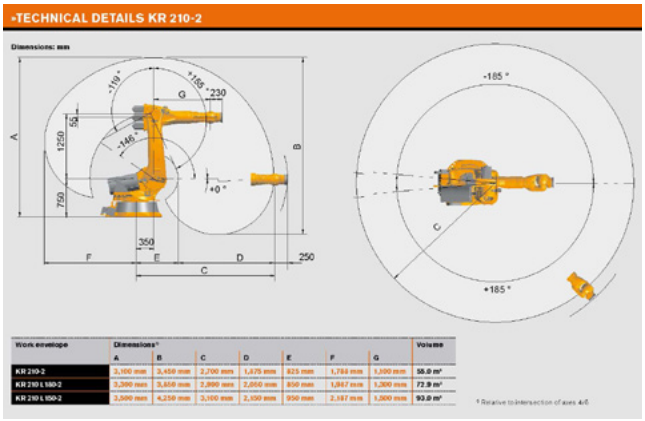
3. A gauche, résultat après 10 minutes de calcul à l'aide de Galapagos. A droite, résultat après 2 minutes de calepinage manuel.

3. SIMULATION D'UNE TRAJECTOIRE D'USINAGE

a. Pré-requis



1. Exemples de géométrie attendue en entrée



2. Le modèle précis du robot utilisé est un paramètre important qui conditionnera la taille maximale des objets à usiner

GÉOMÉTRIE

Les objets traités par l'algorithme de simulation d'une trajectoire d'usinage doivent être des solides («Closed brep»). De manière générale tous les objets correspondent à des panneaux en trois dimensions et leur épaisseur est constante (fig. 1).

CHOIX D'UN ROBOT D'USINAGE

Pour simuler l'usinage et produire un fichier prêt à l'emploi, il est nécessaire que le modèle précis de la machine à commande numérique soit connu à l'avance car chaque outil a ses propres caractéristiques matérielles et logicielles. Dans le cadre de ce travail le choix s'est porté sur les robots d'usinage de la marque Kuka, qui sont largement répandus et utilisés sur les chaînes de montage dans l'industrie automobile ou dans différents laboratoires de recherche en architecture et ingénierie comme par exemple à l'ITKE de Stuttgart. Cette popularité est due à l'abondante documentation qui accompagne ces machines et la relative facilité de prise en main de cet outil.

De plus, le plugin Kuka PRC pour Grasshopper permet la programmation des robots de cette marque ce qui correspondait à nos attentes en terme de continuum numérique. Enfin un modèle est disponible à l'ENSTIB et pourrait permettre un test à grande échelle.

PARAMÈTRES

Les paramètres permettent de choisir précisément le modèle du robot Kuka et les outils d'usinage (fig. 2).

b. Fonctionnement général

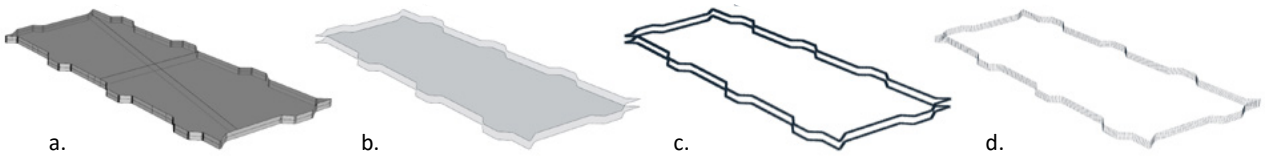
FONCTIONNEMENT DU PLUGIN KUKA PRC

L'algorithme se base sur l'utilisation du plugin Kuka PRC. L'utilisateur peut choisir le type de robot ainsi que l'outil d'usinage qu'il souhaite utiliser parmi une liste (fig. 1, 02 Virtual Robot et 03 Virtual Tool). Différents composants (01 - Core) permettent le calcul des mouvements du robot à partir d'une série de plans successifs qui définissent la trajectoire et l'orientation de l'outil, tandis que d'autres (04 - Toolpath Utilities et 05 - Utilities) offrent des fonctions qui s'appliquent à la géométrie de la trajectoire ainsi qu'au contrôle du robot.

1. Outils proposés par le plugin Kuka PRC.

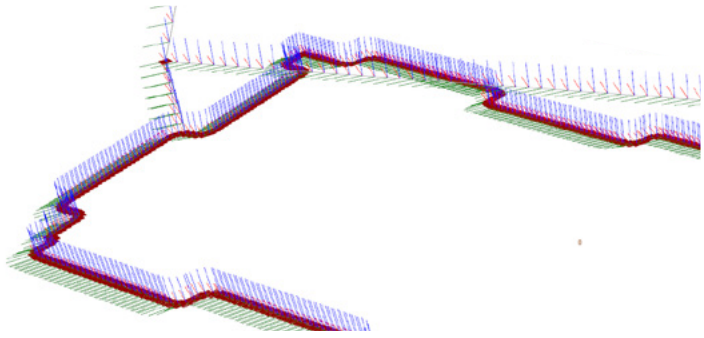
CONSTRUCTION DE LA TRAJECTOIRE

L'algorithme, qui permet de traiter plusieurs objets en même temps, commence d'abord par trier les panneaux en fonction de leur proximité afin de définir une trajectoire optimisée. Les objets sont ensuite décomposés en surfaces qui sont triées selon leur aire afin de ne conserver que les deux plus grandes (fig. 2 b). Les polylignes des contours de ces deux surfaces sont chacune divisées en un même nombre de points (fig. 2 c). Les points sont ensuite reliés ensemble afin de former une suite de lignes (fig. 2 d), qui permettent de définir une série de plans successifs qui définissent la trajectoire.



2. Construction de la trajectoire d'usinage d'une pièce simple.

Afin d'éviter les collisions entre les déplacements, les premiers et derniers plans sont dupliqués et décalés sur leur axe Z pour assurer une marge de manœuvre (fig. 3).



3. Décalage des premiers et derniers plans sur la trajectoire d'usinage.

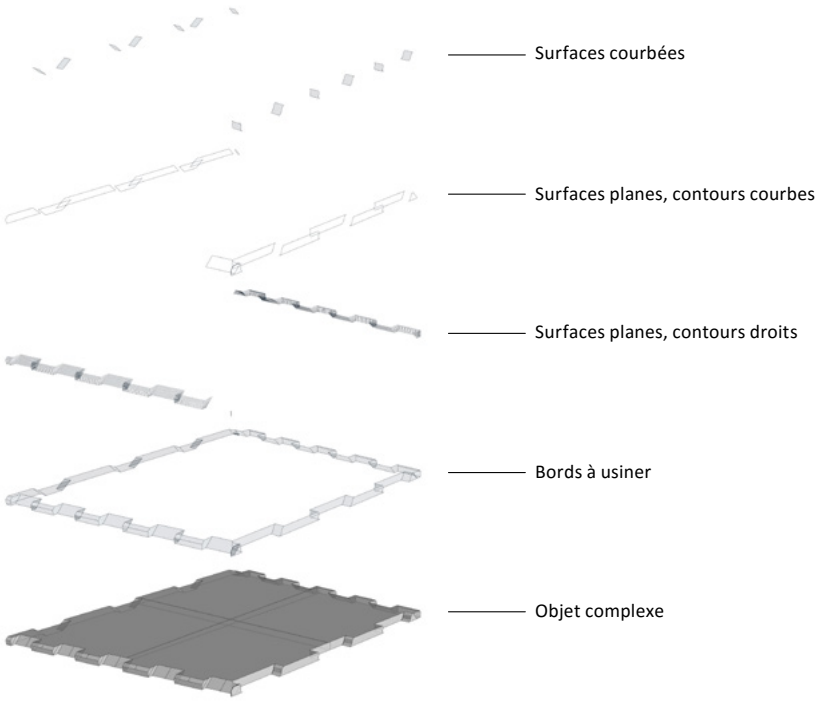
SIMULATION ET EXPORT D'UN FICHIER D'USINAGE

Le plugin calcule les mouvements du robot et les instructions à partir de la succession de plans. L'utilisateur peut alors simuler la position du robot à tout moment et exporter un fichier d'usinage au format standard XML (fig. 4). L'algorithme permet pour l'instant un contrôle visuel mais pourrais par la suite être couplé à une détection des collisions avec les objets ou l'environnement.

4. Simulation de la trajectoire d'usinage et export du fichier d'instructions au format standard XML.

OBJETS COMPLEXES

L'algorithme de modélisation paramétrique d'assemblages en bois génère deux types de géométrie à usiner: les objets simples dont la surface des bords est continue (fig. 2) et les objets complexes dont les bords sont fragmentés et ne forment pas une surface continue (fig. 5). Les différentes surfaces peuvent être courbes ou planes et leurs contours droits ou courbés. La géométrie complexe n'est pas complètement traitée par l'algorithme qui se contente pour l'instant de trier les différents types de surfaces et ne gère que les surfaces planes aux contours droits (fig. 5). Le principe consiste à réaliser plusieurs passages avec l'outil en fonction de la nature des surfaces. Les objets complexes sont beaucoup plus difficile à traiter que les objets simples car cela implique de tenir compte des collisions de l'outil et l'ordre dans lequel découper les surfaces.



5. Les différents types de surfaces d'un objet complexe nécessitent chacun un mode opératoire différent.

4. CONCLUSION

CALEPINAGE AUTOMATISÉ

Bien que l'algorithme de calepinage des panneaux démontre une faisabilité de l'automatisation de cette tâche il reste peu efficace pour l'instant. De nouvelles stratégies sont à explorer et de nouveaux composants à trouver et concevoir pour gagner en rapidité et en qualité. Les paramètres disponibles pourraient également être améliorés car par exemple, le programme ne prend pas en compte le sens de la fibre du bois. On peut relativiser cette étape de calepinage dans la mesure où aujourd'hui dans un contexte industriel les fichiers sont calepinés à l'échelle de la production de l'usine afin d'optimiser au maximum l'exploitation des matériaux. Cela reste cependant un frein important à la mise en place du continuum numérique imaginé.

SIMULATION D'UNE TRAJECTOIRE D'USINAGE

L'algorithme présente une première stratégie possible concernant la préparation automatique de l'usinage de ce type d'architecture. La construction de la trajectoire sur des objets simples semble être efficace, cependant les objets complexes posent beaucoup de problèmes qui risquent d'alourdir l'algorithme, et qui nécessiteraient d'approfondir ce travail. Le programme ne prend pas en compte le calepinage des panneaux et travaille donc sans le matériau brut. Inclure cette contrainte induirait une réduction des mouvements possibles, et apporte ses problématiques : le matériaux brut peut-il être utilisé dans son format standard ou doit-il être découpé grossièrement au préalable ? Les efforts qui agissent lors de l'usinage ainsi que le maintient de la pièce et son repérage ne sont pas abordés et sont pourtant complexes.

Des logiciels très complets disponibles sur le marché sont capables de réaliser cette étape en prenant en compte toutes ces contraintes mais leur coût et la difficulté d'utilisation reste prohibitif pour des projets de taille modeste. L'interopérabilité entre ces logiciels et la géométrie construite dans Grasshopper reste également à démontrer.

V. CONCLUSIONS

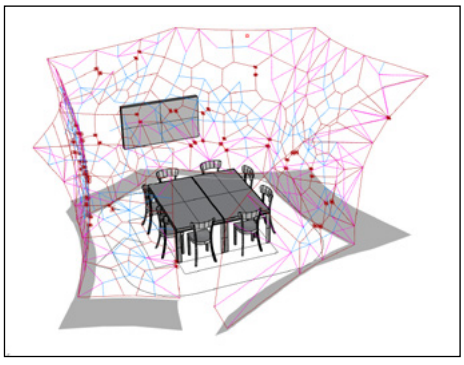
1. CONCLUSION GÉNÉRALE

Le travail mené lors de ce stage a permis de répondre à une grande partie des objectifs et problématiques énoncées initialement, notamment concernant l'amélioration des algorithmes de tessellation de surfaces, qui en plus de fonctionner correctement a pu prendre la forme d'un plugin, et de l'algorithme de génération des assemblages. La dernière étape de préparation à la fabrication numérique est loin d'être complète mais permet néanmoins d'appréhender le programme du début à la fin et constitue un prototype.

De nombreux aspects restent bien évidemment à améliorer dans le cadre d'un travail ultérieur, à commencer par la rapidité de calcul et la variété de la géométrie acceptée par le programme qui est actuellement restreinte à des maillages et des faces plates. Cette contrainte réduit également les types de patterns disponibles à une même famille qui reste relativement simple. L'enjeu ici est de trouver une stratégie permettant de générer des patterns personnalisables simplement, et d'ouvrir à d'autres familles de patterns moins réguliers comme ceux que l'on peut retrouver dans l'art génératif évoqué en conclusion de la partie «II. Tesselation de surfaces». Les types d'assemblages devront quant à eux évoluer pour correspondre à des solutions techniques précises comme vu en conclusion de la partie «III. Modélisation paramétrique d'assemblages en bois». De plus, il reste encore un travail important de synthèse pour donner une forme de plugin aux deux dernières parties.

D'autre part, des points n'ont pas pu être abordés dans ce travail faute de temps, à l'image de l'assemblage des cellules entre elles qui nécessiterai peut-être de remettre en question toute la stratégie de modélisation établie, selon l'importance et la nature des solutions techniques envisagées. On peut également envisager un outil permettant d'affecter des paramètres personnalisés aux cellules en fonction de critères extérieurs. Par exemple, une analyse des sollicitations structurelles, de l'ensoleillement, du comportement thermique ou acoustique aurait une influence sur la déformation d'un pattern, la forme ou l'épaisseur de certaines cellules, la nature des assemblages ou encore même la matérialité. Le projet FabPod (fig. 1) développé par Daniel Davis à l'aide de Grasshopper et présenté sur son blog[2] est un exemple de paroi cellulaire non standard en bois dont la morphologie et la matérialité des cellules sont directement influencées par le comportement acoustique de la paroi au sein de l'outil de conception (fig. 2).

1. Le projet FabPod développé par Daniel Davis.

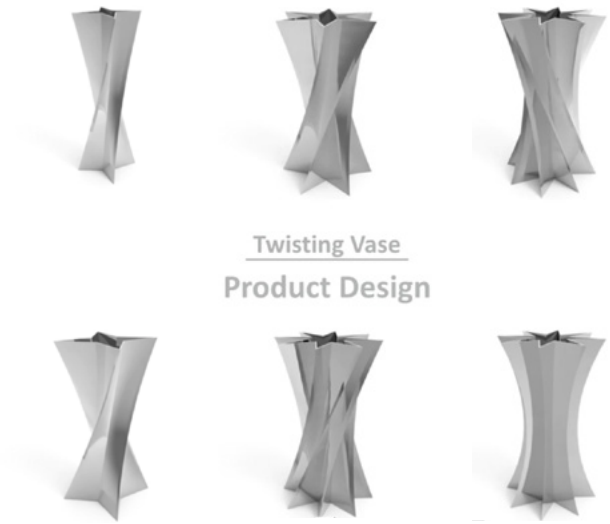


2. Etude acoustique du projet.

2 Davis, Daniel. *DanielDavis.com* [en ligne]. juin 2013 [consulté le 7 juillet 2016]. FabPod. Disponible sur : http://www.danieldavis.com/fabpod/

3. Impression d'écran de l'interface WebGL imaginée par BIG permettant de créer son propre pavillon de la Serpentine Gallery

4. Deux vues de l'application mobile Rat Lab.



5. Différentes versions réalisables avec cette application

Ces améliorations pourraient être effectuées dans le cadre d'un éventuel travail ultérieur conséquent, lors duquel l'implication d'une personne compétente en programmation informatique standard (C#, VB, Python) serait un réel atout. Néanmoins, le travail réalisé démontre la faisabilité et l'intérêt d'un continuum numérique qui permet l'émergence de nouveaux outils de conception. Cette méthode, en favorisant les allers-retours entre réalité constructive et concept, autorise l'exploration rapide de nombreuses variantes et choix jusque dans le détail avec la possibilité de modifier et d'adapter rapidement le projet, voir d'aboutir à des propositions que le concepteur n'avait pas imaginé. Cette dernière caractéristique est certainement l'aspect le plus intéressant et stimulant de l'outil dans le domaine de la conception architecturale.

Pour gagner en crédibilité et imaginer une utilisation professionnelle de ce travail, la création d'outils d'analyse et d'auto-correction du projet en fonction de critères structurels, thermiques, acoustiques ou d'ensoleillement nous semble essentielle. La question des échanges et de l'interopérabilité avec des logiciels plus conventionnels de type BIM comme Archicad ou Revit, mais aussi la possibilité de travailler de manière collaborative sur le projet devront également être étudiées.

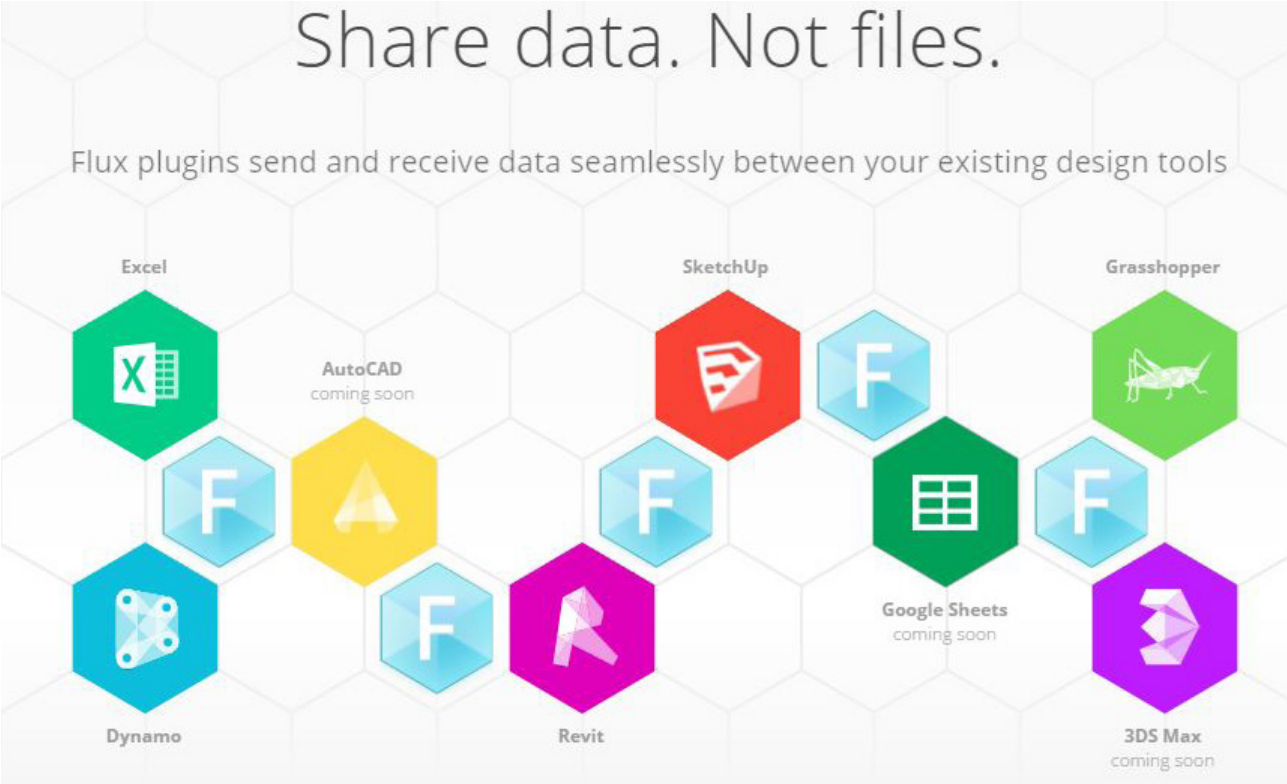
Même si à l'avenir le programme s'améliore et offre des fonctionnalités puissantes, son accès restera fermé à la plupart des concepteurs. Il faudra alors réfléchir à une interface utilisateur plus attractive, intuitive et rapide pour étendre l'utilisation du programme. Les interfaces web par exemple peuvent être envisagées comme des solutions d'avenir, car elles sont accessibles depuis n'importe quel ordinateur ou smartphone et sans logiciel supplémentaire à installer. De plus, La puissance des navigateurs et la rapidité des connexions internet grandissantes a permis le développement de nombreuses applications artistiques basées sur l'art génératif et dont l'intérêt repose souvent sur l'interaction avec le modèle via des curseurs ou des gestes. Les nombreuses pages interactives référencées sur le site Chrome Experiments[2] permettent d'imaginer les différents usages possible de ce type d'interface. Célèbre pour son innovation en matière de communication, l'agence d'architecture BIG a d'ailleurs récemment utilisé une interface WebGL (fig. 3) qui permet à l'utilisateur de «concevoir» en ligne sa propre version de l'édition 2016 de la Serpentine Gallery, dont le projet réel a été conçu grâce à la modélisation paramétrique[3]. Un autre projet, «You're The Designer», est une application pour smartphone disponible sur Android (fig. 4 et 5), développée par le laboratoire de recherche en architecture et technologies rat[LAB][4]. Elle permet à des concepteurs de partager des projets de design dont les utilisateurs peuvent personnaliser les paramètres en temps réel en jouant avec des curseurs.

---

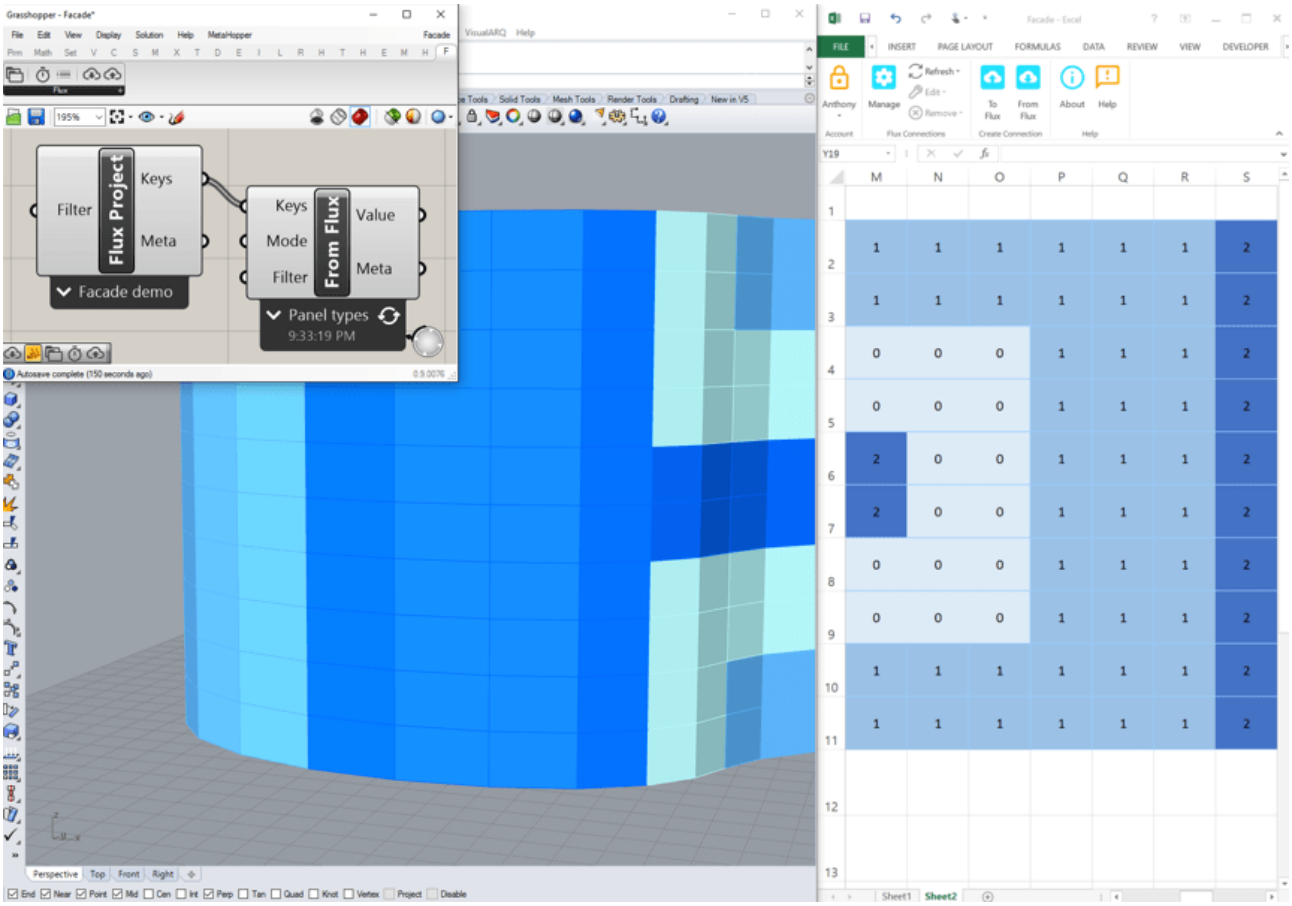
2 *Chrome Experiments* [en ligne]. [consulté le 28 août 2016]. Disponible sur : https://www.chromeexperiments.com/

3 *ArchDaily* [en ligne]. Archilogic, 7 juin 2016, [consulté le 10 juin 2016]. Play With a Parametric Version of BIG's Serpentine Pavilion in this Model. Disponible sur : http://www.archdaily.com/789013/play-with-a-parametric-version-of-bigs-serpentine-pavilion-in-this-model

4 *Rat[LAB]-Research in architecture and technology* [en ligne]. 2012 [consulté le 12 juillet 2016]. Disponible sur : http://www.rat-lab.org/



6. Principe de fonctionnement de la plate forme d'échange Flux.



7. Exemple d'échange réalisé à l'aide de Flux entre Rhinocéros-Grasshopper et Excel.

Le but de ces interfaces n'est pour l'instant pas un usage de conception mais plutôt un moyen de communiquer sur des projets et d'initier le grand public aux possibilités offertes par la modélisation paramétrique. Il ne serait peut-être pas intéressant d'imaginer ouvrir l'utilisation de cet outil d'aide à la conception de parois non standards au grand public, mais il est plutôt question ici de s'inspirer de la dimension pédagogique et intuitive de ces interfaces.

Des solutions à ces problématiques d'interopérabilité, de travail collaboratif et d'amélioration de l'interface émergeront certainement dans un avenir proche, et certains projets intéressants sont déjà en cours. On peut par exemple citer l'arrivée en 2016 de la connexion en temps réel entre Rhinocéros, Grasshopper et Archicad qui est loin d'être aboutie mais ouvre des perspectives intéressantes[5]. L'interface offerte par Dynamo dans Revit présent depuis plusieurs années continue de se développer[6], et d'autres projets émergent à l'image de la plateforme d'échange Flux[7] qui vise à centraliser en temps réel les données de nombreux logiciels comme Excel, Grasshopper, Revit, Sketchup ou encore 3DSmax (fig. 6 et 7). Ce dernier projet semble avoir le potentiel de résoudre les problèmes d'interopérabilité, de travail collaboratif et d'interface en adoptant une forme proche de ce qu'est actuellement Google Drive[8] pour le domaine de la bureautique.

Les grands groupes éditeurs de logiciels ainsi que les architectes commencent à prendre conscience de l'intérêt et du potentiel de ces outils qui arrivent sur le marché, qui restent aujourd'hui à l'état de prototype mais pourraient bien se développer rapidement et occuper une place importante dans le domaine de l'architecture, de la conception et de la construction dans les années à venir.

---

5 Graphisoft [en ligne]. Connection Archicad-Rhinocéros-Grasshopper [consulté le 28 août 2016]. Disponible sur : http://www.graphisoft.com/archicad/rhino-grasshopper/

6 DynamoBim [en ligne]. DynamoBIM pour Revit [consulté le 28 août 2016]. Disponible sur : http://dynamobim.org/

7 Flux [en ligne]. *Design tools and teams, working together* [consulté le 28 août 2016]. Disponible sur : https://flux.io/

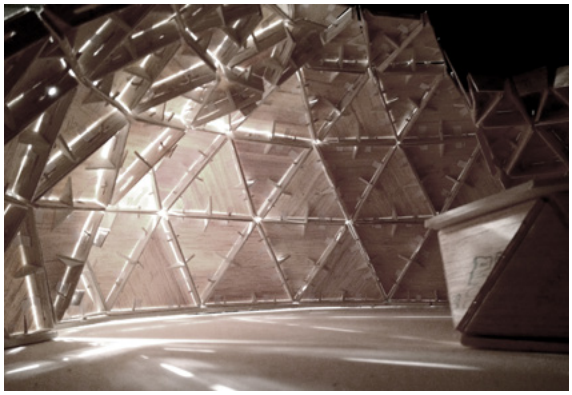
8 GoogleDrive [en ligne]. Google Drive [consulté le 28 août 2016]. Disponible sur : https://www.google.com/intl/fr_fr/drive/

2. BILAN PERSONNEL : THOMAS EHRHARDT

Ce stage de recherche au sein du laboratoire Map-CRAI a été pour moi une expérience très riche, tant au niveau personnel que professionnel, pour l'année en cours, ainsi que pour ma future carrière. En effet, j'ai pu découvrir une nouvelle façon de penser un projet. Re-questionner un projet permet de comprendre les forces et les faiblesses de ce dernier, pour y trouver des améliorations. Des exemples de travaux existants sur le sujet ainsi que des travaux dans d'autres domaines permettent également de nourrir le projet. La cadre de travail d'un laboratoire de recherche encourage les échanges d'informations et les discussions ce qui stimule continuellement la réflexion.

Le sujet de cette recherche sur les parois non-standard en bois est un domaine de l'architecture qui m'a intéressé depuis la première année d'études. En effet, dès le projet de pavillon bois encadré par J.C Bignon en 2ème année (fig. 1), c'est instinctivement que je me suis tourné vers une ossature cellulaire en bois. Ce projet a été principalement conçu en maquette au vu de sa géométrie particulière. L'apport de la modélisation paramétrique m'aurait permis d'explorer de nouvelles formes et de gagner un temps considérable. D'autres projets en collaboration avec Guillaume ont pu être réalisés à l'aide de Rhinocéros et Grasshopper, comme le concours acier de 4ème année ou le projet de 1er semestre de 2ème année de master. Nous avons surtout appris à utiliser ces logiciels lors de ces projets, sans réelle méthode. Cependant, lors de stage nous avons découvert une nouvelle approche de ce couple de logiciels. En effet, pour concevoir un algorithme de cette ampleur, une étude préalable est nécessaire afin de bien fixer les enjeux, ainsi que toutes les étapes de l'algorithme. Ce travail en amont aurait pu nous faire gagner un temps considérable lors de la conception de ces projets.

Nous avons eu la chance de travailler en même temps sur notre projet de PFE sur une problématique similaire (fig. 2). C'est ainsi que des allers-retours ont pu avoir lieu entre projet et recherche. Ils ont permis d'enrichir les deux aspects et de nous faire gagner beaucoup de temps.



1. Maquette du projet du Pavillon bois pour Gipeblor

2. Perspective du PFE encadré par S. Rinckel et R. Rouyer

Cependant, comment pourrais-je appliquer ces compétences dans une future agence ? Il est vrai qu'aujourd'hui seulement très peu d'agences utilise ce genre d'outils numériques. Cependant pour répondre à certaines problématiques, la modélisation algorithmique peut être très utile, notamment dans l'automatisation de processus techniques. Même si cette modélisation paramétrique n'est pas directement utilisée dans une agence, la manière de concevoir peut, quant à elle, être facilement appliquée. En effet, pour un algorithme comme pour un projet, il faut avoir pleinement conscience des objectifs, des contraintes et des différentes étapes nécessaires au travail de conception.

3. BILAN PERSONNEL : GUILLAUME GINEFRI

Ce stage constitue pour moi une approche du travail de recherche en architecture et ingénierie intéressante et enrichissante. J'ai pu approfondir mes compétences et découvrir les possibilités offertes par des logiciels de modélisation paramétrique et algorithmique comme Rhinocéros et Grasshopper. Le travail réalisé durant ce stage a été également enrichissant sur les aspects professionnels et personnels lors des échanges que j'ai pu avoir avec nos encadrants Gilles Duchanois et Oskar Gámez, ainsi que toute l'équipe des chercheurs, doctorants et stagiaires du CRAI.

La thématique du stage sur les parois non standards en bois fait écho à des projets auxquels j'ai pu participer durant mes études à l'école d'architecture de Nancy, comme par exemple le projet de pavillon en bois pour le stand du Gipeblor en 2ème année qui était également constitué de cellules non standards en bois (fig. 1) ou le projet de fin d'études qui s'en rapproche (fig. 2). J'ai aussi eu l'occasion d'utiliser la modélisation paramétrique lors du projet de concours acier en 4ème année ou de l'atelier de projet cultures constructives au premier semestre de 5ème année, et je pense que les compétences que j'ai pu acquérir durant ce stage auraient une grande influence sur ces projets si j'avais à les refaire. Les allers-retours que nous avons pu effectuer entre le travail de PFE et ce stage qui se sont déroulés en parallèle et sur des problématiques proches a été lui aussi enrichissant et nous a permis notamment de capitaliser du temps de travail sur des stratégies de modélisation.

1. Projet de 2ème année. Pavillon pour le stand du Gipeblor.

2. PFE. Pavillons pour les espaces publics de Lunéville.

Au terme de mon parcours à l'école d'architecture, ma perception du travail de conception évolue grâce aux problématiques abordées par ce stage, que je peux confronter à mon travail de recherche de 3ème année sur le BIM, les différents stages effectués en agence d'architecture ainsi que la formation AME suivie en double cursus cet année. Ces différentes expériences m'amènent à me questionner sur la manière d'échanger et de travailler en collaboration en respectant le continuum numérique dans un contexte professionnel, faisant intervenir des acteurs dont les compétences et les objectifs sont différents. Par exemple est-il possible d'utiliser la modélisation algorithmique en lien avec des logiciels BIM plus conventionnels pour permettre ce travail collaboratif ? Cela peut-il fonctionner et être économiquement viable dans le cadre de projets et d'agences de taille modeste ? Quels avantages mais aussi quels inconvénients peuvent présenter ces nouveaux outils de conception ? D'autres outils et méthodes sont certainement encore à inventer pour permettre le continuum numérique tout au long du projet, qui reste pour l'instant expérimental.

BIBLIOGRAPHIE

LIVRES

Jackson, Albert & Day, David. *Good Wood Joints*. Harpercollins, 1995. 128 p.

Tedeschi, Arturo. *AAD Algorithms-Aided Design : Parametric Strategies Using Grasshopper.* Le Penseur, 2014. 495 p.

WEBOGRAPHIE

SITES INTERNET

Chrome Experiments [en ligne]. [consulté le 28 août 2016]. Disponible sur : https://www.chromeexperiments.com/

Davis, Daniel. *DanielDavis.com* [en ligne]. juin 2013 [consulté le 7 juillet 2016]. Disponible sur : http://www.danieldavis.com/

Designcoding [en ligne]. [consulté le 7 juillet 2016]. Disponible sur : http://www.designcoding.net/

DynamoBim [en ligne]. DynamoBIM pour Revit [consulté le 28 août 2016]. Disponible sur : http://dynamobim.org/

Flux - Design tools and teams, working together [en ligne]. [consulté le 18 juillet 2016]. Disponible sur : https://flux.io/

GoogleDrive [en ligne]. Google Drive [consulté le 28 août 2016]. Disponible sur : https://www.google.com/intl/fr_fr/drive/

Marc Fornes & Theverymany, practicing at the intersection of art, architecture and computation [en ligne]. [consulté le 12 juillet 2016]. Disponible sur : https://theverymany.com/

Rat[LAB]-Research in architecture and technology [en ligne]. 2012 [consulté le 12 juillet 2016]. Disponible sur : http://www.rat-lab.org/

BLOGS

Meier, Mark. *Digital Fabrication for Designers* [en ligne]. 2010 [consulté le 7 juillet 2016]. Disponible sur : http://mkmra2.blogspot.fr/

Tessmann, Olivier. *Computation in design* [en ligne]. [consulté le 30 août 2016]. Disponible sur : https://olivertessmann.wordpress.com/

THÈSES ET RAPPORTS SCIENTIFIQUES

Davis, Daniel. *Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture*. School of Architecture and Design, College of Design and Social context, RMIT University, février 2013. 243 p. [consulté le 11 juillet 2016]. Disponible sur : http://www.danieldavis.com/papers/danieldavis_thesis.pdf

Sutherland, Ivan Edward. *Sketchpad : A man-machine graphical communication system*. University of Cambridge Computer Laboratory, septembre 2003. 149 p. [consulté le 11 juillet 2016]. Disponible sur : http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-574.pdf

ARTICLES

ArchDaily [en ligne]. Archilogic, 7 juin 2016, [consulté le 10 juin 2016]. Play With a Parametric Version of BIG's Serpentine Pavilion in this Model. Disponible sur : http://www.archdaily.com/789013/play-with-a-parametric-version-of-bigs-serpentine-pavilion-in-this-model

ArchiRADAR [en ligne]. ArchiRADAR ltd, [consulté le 8 juillet 2016]. Grasshopper-Archicad connection. Disponible sur : http://www.archiradar.it/en/books/tutorials-archicad-artlantis/45-tutorials/rhino-grasshopper-archicad.html

Autodesk Research [en ligne]. Autodesk, [consulté le 8 juillet 2016]. Project Dreamcatcher. Disponible sur : https://autodeskresearch.com/projects/dreamcatcher

Bernstein, Phil. *Line/Shape/Space* [en ligne]. Autodesk, 27 avril 2015 [consulté le 8 juillet 2016]. Créer vos propres règles grâce à l'architecture génératrice. Disponible sur : https://lineshapespace.fr/2015/04/27/creer-vos-propres-regles-avec-larchitecture-generatrice/

Cad-magazine [en ligne]. CIMAX, [consulté le 8 juillet 2016]. Modélisation : paramétrique VS directe. Disponible sur : http://www.cad-magazine.com/article/modelisation-parametrique-vs-directe

CADAZZ [en ligne]. 2003 [consulté le 8 juillet 2016]. CAD Software - History of CAD CAM. Disponible sur : http://www.cadazz.com/cad-software-Sketchpad.htm

Core 77 [en ligne]. Rain Noe, 18 novembre 2015, [consulté le 5 juillet 2016]. Reference: The Ultimate Wood Joint Visual Reference Guide. Disponible sur : http://www.core77.com/posts/43001/Reference-The-Ultimate-Wood-Joint-Visual-Reference-Guide

Davis, Daniel. *DanielDavis.com* [en ligne]. 6 août 2013 [consulté le 7 juillet 2016]. A History of Parametric. Disponible sur : http://www.danieldavis.com/a-history-of-parametric/

Davis, Daniel. *DanielDavis.com* [en ligne]. juin 2013 [consulté le 7 juillet 2016]. FabPod. Disponible sur : http://www.danieldavis.com/fabpod/

Flajolet, Philippe & Parizot, Etienne. *Interstices* [en ligne]. INRIA, 24 février 2004 [consulté le 8 juillet 2016]. Qu'est-qu'un algorithme ? Disponible sur : https://interstices.info/jcms/c_5776/qu-est-ce-qu-un-algorithme

Graphisoft [en ligne]. Connection Archicad-Rhinocéros-Grasshopper [consulté le 28 août 2016]. Disponible sur : http://www.graphisoft.com/archicad/rhino-grasshopper/

Sandworks [en ligne]. 35Degree Team, [consulté le 8 juillet 2016]. Material. Disponible sur : https://sandworks.org/material-2/

Senses Lost [en ligne]. 30 mai 2012, [consulté le 28 août 2016]. Video Interview with Joshua Davis. Disponible sur : http://senseslost.com/2012/05/30/video-interview-with-joshua-davis/

Swenson Gordon, Kylee. *Line/Shape/Space* [en ligne]. Autodesk, 30 juin 2016 [consulté le 8 juillet 2016]. Qu'est-que la conception générative ? Disponible sur : https://lineshapespace.fr/2016/06/30/definition-conception-generative/?linkId=26678694

TABLE DES ILLUSTRATIONS

À l'exception des illustrations référencées ci-dessous, toutes les images présentées dans ce travail de recherche ont été réalisées par nos soins, qu'il s'agisse de schémas ou d'impressions d'écran des différents programmes.

I. 3. a. 1. Oskar Gámez, Thèse. Réalisation d'un prototype en 2014

I. 3. a. 2. Oskar Gámez, Thèse. Réalisation d'un prototype en 2014

I. 3. a. 3. Oskar Gámez, Thèse. Réalisation d'un prototype en 2014

I. 3. a. 4. Oskar Gámez, Thèse. Réalisation d'un prototype en 2014

II. 1. a. 1. Mosaïque grecque (musée de Delphes). In *Oktana* [en ligne]. [Consulté le 19 juillet 2016]. Disponible sur: http://oktana-gr.blogspot.fr/2014/10/blog-post_67.html

II. 1. a. 2. Mosaïque romaine à Carmona (Séville). In *123RF* [en ligne]. [consulté le 19 Juillet 2016]. Disponible sur: http://it.123rf.com/photo_31202096_mosaico-romano-carmona-provincia-di-siviglia-andalusia-spagna.html

II. 1. a. 3. Mosaïque de l'Alcazar (Séville). In *Pinterest* [en ligne]. [consulté le 19 juillet 2016]. Disponible sur : https://fr.pinterest.com/marwanhi101/mud%C3%A9jar/

II. 1. b. 1. Oskar Gámez, Thèse. Chapter 2. Digital morphogenesis : machines, patterns and architecture, 2016.

II. 1. c. 2. Oskar Gámez, Thèse. Chapter 2. Digital morphogenesis : machines, patterns and architecture, 2016.

II. 1. c. 3. Véronique Sagliet, Façade de la Philharmonie, La Villette, Paris. in *Fotolia* [en ligne]. [consulté le 12 Août 2016]. Disponible sur : https://fr.fotolia.com/tag/%22philharmonie%22%22Paris%22

II. 1. c. 4. Oskar Gámez, Thèse. Chapter 2. Digital morphogenesis : machines, patterns and architecture, 2016.

II. 4. b. 1. Gauche et Droite. Exemples de pattern généré par Grasshopper conçus par des étudiants. In *DesignCoding* [en ligne]. [Consulté le 18 Août 2016]. Disponible sur : http://www.designcoding.net/seamless-patterns-2/

II. 4. b. 1. milieu. Exemples de pattern généré par Grasshopper conçus par des étudiants. In *DesignCoding* [en ligne]. [Consulté le 18 Août 2016]. Disponible sur : http://www.designcoding.net/pattern-deformation/

II. 4. b. 2. Gauche. Travail de Joshua Davis. In *Instagram* [en ligne]. [Consulté le 20 Août 2016]. Disponible sur : http://blog.instagram.com/post/130883359187/151010-praystation

II. 4. b. 2. Milieu et Droite. Travail de Joshua Davis. In *Pinterest* [en ligne]. [Consulté le 20 Août 2016]. Disponible sur : https://fr.pinterest.com/praystation/

II. 4. b. 3. Impression d'écran de l'algorithme «particle Love» de Edan Kwan. In *ChromeExperiments* [en ligne]. [Consulté le 22 Août 2016]. Disponible sur : http://edankwan.com/experiments/the-spirit/

III. 1. a. 1. Oskar Gámez, Modélisation paramétrique d'assemblages non standards en bois, 2016.

III. 1. b. 1. Oskar Gámez, Modélisation paramétrique d'assemblages non standards en bois, 2016.

III. 1. b. 2. Oskar Gámez, Modélisation paramétrique d'assemblages non standards en bois, 2016.

III. 1. a. 2. Assemblages traditionnels. Extrait de «Good Wood Joints» par Albert Jackson & David Day. In *Core 77* [en ligne]. Rain Noe, 18 novembre 2015, [consulté le 5 juillet 2016]. Reference: The Ultimate Wood Joint Visual Reference Guide. Disponible sur : http://www.core77.com/posts/43001/Reference-The-Ultimate-Wood-Joint-Visual-Reference-Guide

III. 3. 2. Deux exemples d'assemblages innovants réalisés à l'aide d'une machine à commande numérique. In *Digital Fabrication for Designers* [en ligne]. Meier, Mark. CNC Cut Wood Joinery, 24 août 2014 [consulté le 22 juillet 2016]. Disponible sur : http://mkmra2.blogspot.fr/2014/08/cnc-cut-wood-joinery.html

IV. 3. a. 2. Extrait de la documentation technique Kuka, Series 2000, KR 210-2.

V. 1. 1 et 2. Photo du projet FabPod et étude acoustique. In Davis, Daniel. *DanielDavis.com* [en ligne]. juin 2013 [consulté le 7 juillet 2016]. FabPod. Disponible sur : http://www.danieldavis.com/fabpod/

V. 2. 3. Impression d'écran de l'interface WebGL de création de pavillon de la Serpentine Gallery. In *ArchDaily* [en ligne]. Archilogic, 7 juin 2016, [consulté le 10 juin 2016]. Play With a Parametric Version of BIG's Serpentine Pavilion in this Model. Disponible sur : http://www.archdaily.com/789013/play-with-a-parametric-version-of-bigs-serpentine-pavilion-in-this-model

V. 2. 4. et 5 aperçu de l'application RatLab. In *Rat[LAB]-Research in architecture and technology* [en ligne]. 2012 [consulté le 12 juillet 2016]. Disponible sur : http://www.rat-lab.org/

V. 2. 6. et 7 Images tirés du Site internet de Flux. In *Flux* [en ligne]. *Design tools and teams, working together* [consulté le 28 août 2016]. Disponible sur : https://flux.io//

V. 2. 1. Thomas Ehrhardt. Projet de 2ème année. Maquette pour le pavillon du stand du Gipeblor, 2012.

V. 2. 2. Thomas Ehrhardt. Projet de fin d'études. Pavillons pour les espaces publics de Lunéville, 2016.

V. 3. 1. Guillaume Ginefri. Projet de 2ème année. Maquette pour le pavillon du stand du Gipeblor, 2012.

V. 3. 2. Guillaume Ginefri. Projet de fin d'études. Pavillons pour les espaces publics de Lunéville, 2016.

LOGICIELS

- Rhinocéros® par Robert McNeel & Associates, version : 5 SR12
Disponible sur (version d'évaluation) : http://www.rhino3d.com/

- Grasshopper® par Robert McNeel & Associates, version : 0.9.0076
Disponible sur : http://www.grasshopper3d.com/

TABLE DES PLUGINS

- Anemone par Mateusz Zwierzycki, version : 0.4
Disponible sur : http://www.food4rhino.com/project/anemone?etx

- GENERATION par Antonio Turiello, version : 1.0.0.0
Disponible sur : http://www.food4rhino.com/project/generation?etx

- GhPython par McNeelEurope, version : 0.6.0.3
Disponible sur : http://www.food4rhino.com/project/ghpython?etx

- Heteroptera par Amin Bahrami, version : 0.1.0.6
Disponible sur : http://www.food4rhino.com/project/heteroptera?etx

- Human par Andrew Heumann, version : 0.6.2.0
Disponible sur : http://www.food4rhino.com/project/human-ui?etx

- Kangaroo par Daniel Piker, version : 2.0.2
Disponible sur : http://www.food4rhino.com/project/kangaroo?etx

- KUKA|PRC par Sigrid & Johannes @ Association for Robots in Architecture, version : 1.0.0.9
Disponible sur (version d'évaluation) : http://www.food4rhino.com/project/kukaprc?etx

- LunchBox par Nathan Miller, version 2016.3.21
Disponible sur : http://www.food4rhino.com/project/lunchbox?etx

- Mesh+ par David Mans, version : 2.1.00
Disponible sur : http://www.food4rhino.com/project/mesh?etx

- PanelingTools for Rhino 5.0 par Rajaa Issa, version : 2016-01-20-00
Disponible sur : http://www.food4rhino.com/project/panelingtools?etx

- StarFish par Michael Weizmann, version : 0.1
http://www.food4rhino.com/project/starfish?etx

- TT Toolbox par The Core Studio Team, version : 1.7
Disponible sur : http://www.food4rhino.com/project/tttoolbox?etx

- Viper par Tangchi, version : 20160119
Disponible sur : http://www.food4rhino.com/project/tangchi?etx

GLOSSAIRE

BREP (Boundary representation) : *Une BREP est une technique de modélisation 3D par les bords. On l'appelle également modélisation surfacique. Un BREP est un volume représenté par sa «peau» qui est composée de faces (surface NURBS), de sommets et d'arrêtes (NURBS, B-Splines). Il existe aujourd'hui deux manières de modéliser un solide en CAO, la BREP et la CSG (Constructive Solid Geometry, ou modélisation volumique). Communément, on appelle «un BREP» un volume représenté par ses bords.*

Booléen : *Un booléen est une variable à deux états. Ces derniers sont représentés sous la forme de «true» et «false» dans Rhinocéros.*

Cluster : *Du verbe anglais «se rassembler». Il représente dans Grasshopper un ensemble de composants. Les Clusters permettent d'une part d'alléger visuellement un programme Grasshopper et d'autre part de répéter cet ensemble de composants plusieurs fois sans alourdir un fichier.*

Composant : *Dans Grasshopper, un composant est une fonction et est représentée par une «boîte». Il est possible de créer de nouveaux composants, soit par la programmation classique (VB, C# ou Python), soit en regroupant plusieurs fonctions de bases dans un seul cluster.*

Continuum numérique : *Un continuum est un ensemble d'éléments tel que l'on passer de l'un à l'autre de façon continue. Ainsi dans un continuum numérique, les données numériques peuvent passer d'une étape à une autre de la conception à la fabrication sans intervention manuelle d'un opérateur. Un continuum numérique induit une résolution de tous les problèmes d'interopérabilités.*

Interopérabilité : *L'interopérabilité est la capacité de logiciels ou matériels différents à fonctionner ensemble et à partager des informations ensemble sans aucune perte de données.*

Mesh (Maillage en Français) : *Un mesh est un ensemble de polygones et de points permettant de définir la forme d'un objet. Les meshs de Rhinocéros ne peuvent être créés qu'à partir de quads et de triangles.*

Opération Booléenne : *Les opérations booléennes sont des opérations logiques reprenant les principes de l'algèbre de Boole AND, OR, NOT, XOR (union, intersection et différence). Les opérations booléennes sont, dans notre cas, considérées comme des opérations entre solides.*

Quad (abréviation d'un quadrilatère) *: Un Quad est un polygone possédant quatre cotés reliés par des sommets. Les quads sont les polygones les plus utilisés dans la modélisation 3D par maillage.*

Surface NURBS : *les NURBS sont utilisés pour représenter mathématiquement des objets géométriques, pouvant décrire avec une grande précision toute forme, allant de la simple ligne 2D, un cercle, une surface ou un volume organique à la géométrie très complexe. Une surface NURBS est définie par un réseau de points de contrôle disposés selon ses directions U et V, et par un degré de courbure.*

Vertice : *on appelle «vertice» tout point constituant un maillage.*

Cet ouvrage est un mémoire de travaux réalisés lors d'un stage de recherche au laboratoire du MAP-CRAI à l'école nationale supérieure d'architecture de Nancy.

En lien direct avec le travail de thèse d'Oskar GAMEZ, ce mémoire porte sur la création d'un outil de conception et de fabrication de parois cellulaires non-standards en bois, réalisé principalement sur les logiciels Rhinocéros® et Grasshopper®.