Méthodologie d'aide à la programmation en modélisation architecturale paramétrique

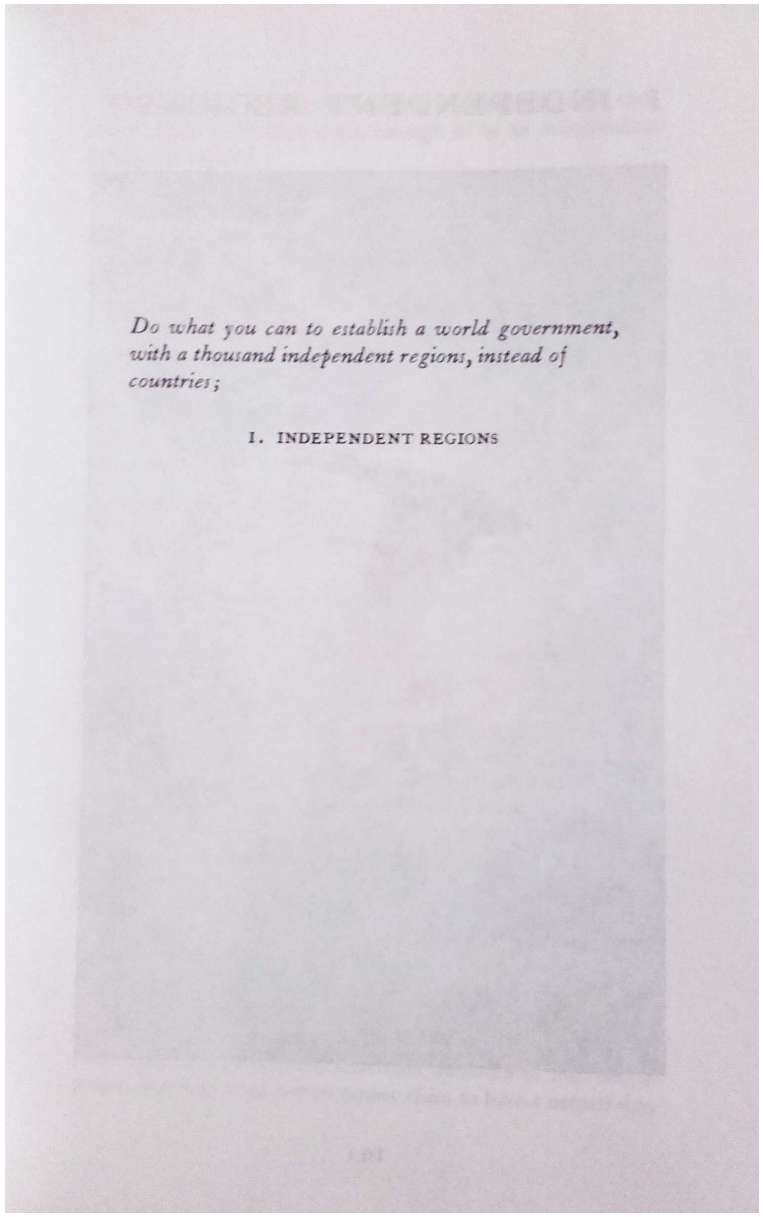
Auteur : Vervier, Charlotte
Promoteur(s) : Leclercq, Pierre
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil architecte, à finalité spécialisée en ingénierie architecturale et urbaine
Année académique : 2016-2017
URI/URL : http://hdl.handle.net/2268.2/2541

9. ANNEXES

Table des annexes :

1 - Exemple d'un pattern d'Alexander : Pattern 1 – Independent Region

D'après : 'A Pattern Language' de Christopher Alexander (1977, p. 9-15)



Do what you can to establish a world government, with a thousand independent regions, instead of countries;

1. INDEPENDENT REGIONS

I INDEPENDENT REGIONS**

Metropolitan regions will not come to balance until each one is small and autonomous enough to be an independent sphere of culture.

There are four separate arguments which have led us to this conclusion: 1. The nature and limits of human government. 2. Equity among regions in a world community. 3. Regional planning considerations. 4. Support for the intensity and diversity of human cultures.

1. There are natural limits to the size of groups that can govern themselves in a human way. The biologist J. B. S. Haldane has remarked on this in his paper, "On Being the Right Size":

> . . . just as there is a best size for every animal, so the same is true for every human institution. In the Greek type of democracy all the citizens could listen to a series of orators and vote directly on questions of legislation. Hence their philosophers held that a small city was the largest possible democratic state. . . . (J. B. S Haldane, "On Being the Right Size," *The World of Mathematics, Vol. II*, J. R. Newman, ed. New York: Simon and Schuster, 1956, pp. 962–67).

It is not hard to see why the government of a region becomes less and less manageable with size. In a population of N persons, there are of the order of N^2 person-to-person links needed to keep channels of communication open. Naturally, when N goes beyond a certain limit, the channels of communication needed for democracy and justice and information are simply too clogged, and too complex; bureaucracy overwhelms human processes.

And, of course, as N grows the number of levels in the hierarchy of government increases too. In small countries like Denmark there are so few levels, that any private citizen can have access to the Minister of Education. But this kind of direct access is quite impossible in larger countries like England or the United States.

We believe the limits are reached when the population of a region reaches some 2 to 10 million. Beyond this size, people become remote from the large-scale processes of government. Our estimate may seem extraordinary in the light of modern history: the nation-states have grown mightily and their governments hold power over tens of millions, sometimes hundreds of millions, of people. But these huge powers cannot claim to have a natural size.

They cannot claim to have struck the balance between the needs of towns and communities, and the needs of the world community as a whole. Indeed, their tendency has been to override local needs and repress local culture, and at the same time aggrandize themselves to the point where they are out of reach, their power barely conceivable to the average citizen.

2. Unless a region has at least several million people in it, it will not be large enough to have a seat in a world government, and will therefore not be able to supplant the power and authority of present nation-states.

We found this point expressed by Lord Weymouth of Warminster, England, in a letter to the *New York Times,* March 15, 1973:

WORLD FEDERATION: A THOUSAND STATES

. . . the essential foundation stone for world federation on a democratic basis consists of regionalization within centralized government. . . . This argument rests on the idea that world government is lacking in moral authority unless each delegate represents an approximately equal portion of the world's population. Working backward from an estimate of the global population in the year 2000, which is anticipated to rise to the 10,000 million mark, I suggest that we should be thinking in terms of an ideal regional state at something around ten million, or between five and fifteen million, to give greater flexibility. This would furnish the U.N. with an assembly of equals of 1000 regional representatives: a body that would be justified in claiming to be truly representative of the world's population.

Weymouth believes that Western Europe could take some of the initiative for triggering this conception of world government. He looks for the movement for regional autonomy to take hold in the European Parliament at Strasbourg; and hopes that power can gradually be transferred from Westminister, Paris, Bonn, etc., to regional councils, federated in Strasbourg.

I am suggesting that in the Europe of the future we shall see England split down into Kent, Wessex, Mercia, Anglia and Northumbria, with an independent Scotland, Wales and Ireland, of course. Other European examples will include Brittany, Bavaria and Calabria. The national identities of our contemporary Europe will have lost their political significance.

3. Unless the regions have the power to be self-governing, they

will not be able to solve their own environmental problems. The arbitrary lines of states and countries, which often cut across natural regional boundaries, make it all but impossible for people to solve regional problems in a direct and humanly efficient way.

An extensive and detailed analysis of this idea has been given by the French economist Gravier, who has proposed, in a series of books and papers, the concept of a Europe of the Regions, a Europe decentralized and reorganized around regions which cross present national and subnational boundaries. (For example, the Basel-Strasbourg Region includes parts of France, Germany, and Switzerland; the Liverpool Region includes parts of England and parts of Wales). See Jean-François Gravier, "L'Europe des regions," in 1965 Internationale Regio Planertagung, Schriften der Regio 3, Regio, Basel, 1965, pp. 211–22; and in the same volume see also Emrys Jones, "The Conflict of City Regions and Administrative Units in Britain," pp. 223–35.

4. Finally, unless the present-day great nations have their power greatly decentralized, the beautiful and differentiated languages, cultures, customs, and ways of life of the earth's people, vital to the health of the planet, will vanish. In short, we believe that independent regions are the natural receptacles for language, culture, customs, economy, and laws and that each region should be separate and independent enough to maintain the strength and vigor of its culture.

The fact that human cultures within a city can only flourish when they are at least partly separated from neighboring cultures is discussed in great detail in MOSAIC OF SUBCULTURES (8). We are suggesting here that the same argument also applies to regions —that the regions of the earth must also keep their distance and their dignity in order to survive as cultures.

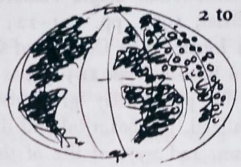
In the best of medieval times, the cities performed this function. They provided permanent and intense spheres of cultural influence, variety, and economic exchange; they were great communes, whose citizens were co-members, each with some say in the city's destiny. We believe that the independent region can become the modern polis—the new commune—that human entity which provides the sphere of culture, language, laws, services, economic exchange, variety, which the old walled city or the polis provided for its members.

Therefore:

Wherever possible, work toward the evolution of independent regions in the world; each with a population between 2 and 10 million; each with its own natural and geographic boundaries; each with its own economy; each one autonomous and self-governing; each with a seat in a world government, without the intervening power of larger states or countries.

each region
2 to 10 million population

1000 regions



✣ ✣ ✣

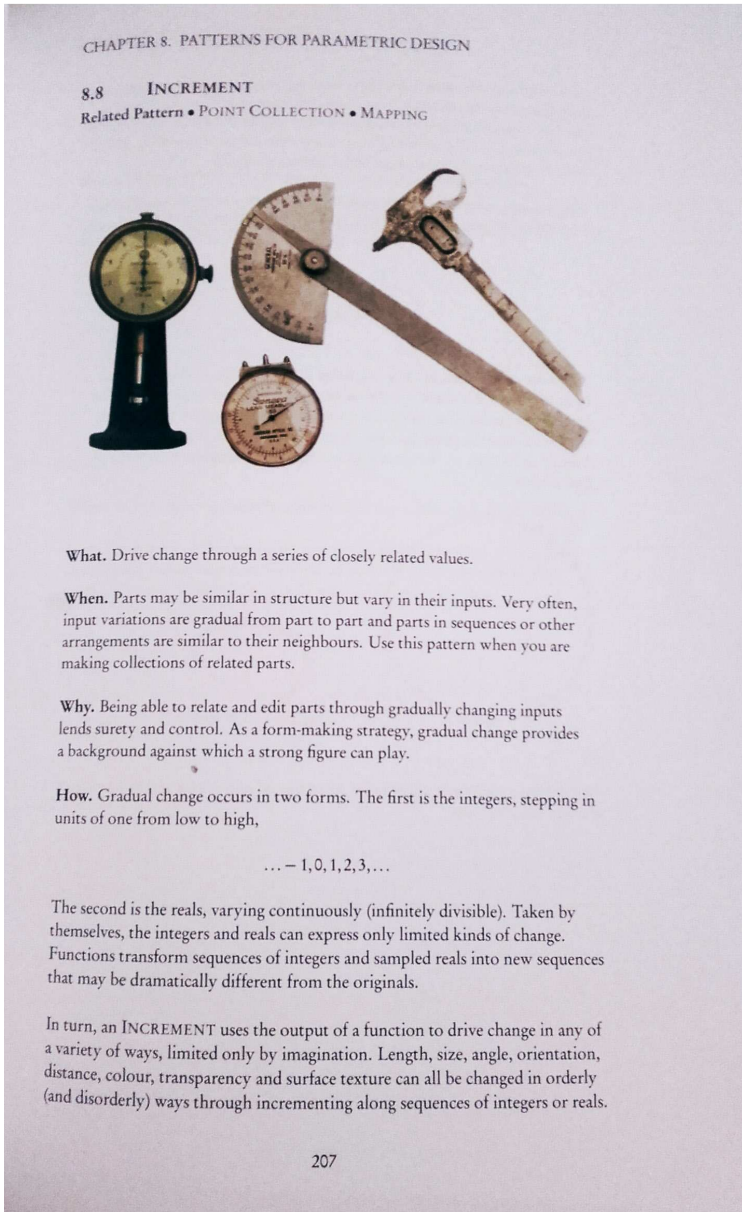
Within each region encourage the population to distribute itself as widely as possible across the region—THE DISTRIBUTION OF TOWNS (2). . . .

within each region work toward those regional policies which will protect the land and mark the limits of the cities:

2. THE DISTRIBUTION OF TOWNS
3. CITY COUNTRY FINGERS
4. AGRICULTURAL VALLEYS
5. LACE OF COUNTRY STREETS
6. COUNTRY TOWNS
7. THE COUNTRYSIDE

2 - Exemple d'un pattern de Woodbury : Pattern 4 – Increment

D'après : 'Elements of Parametric Design' de Robert Woodbury (2010, p. 207-211)



CHAPTER 8. PATTERNS FOR PARAMETRIC DESIGN

8.8 INCREMENT

Related Pattern • POINT COLLECTION • MAPPING

What. Drive change through a series of closely related values.

When. Parts may be similar in structure but vary in their inputs. Very often, input variations are gradual from part to part and parts in sequences or other arrangements are similar to their neighbours. Use this pattern when you are making collections of related parts.

Why. Being able to relate and edit parts through gradually changing inputs lends surety and control. As a form-making strategy, gradual change provides a background against which a strong figure can play.

How. Gradual change occurs in two forms. The first is the integers, stepping in units of one from low to high,

$$\ldots -1, 0, 1, 2, 3, \ldots$$

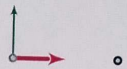
The second is the reals, varying continuously (infinitely divisible). Taken by themselves, the integers and reals can express only limited kinds of change. Functions transform sequences of integers and sampled reals into new sequences that may be dramatically different from the originals.

In turn, an INCREMENT uses the output of a function to drive change in any of a variety of ways, limited only by imagination. Length, size, angle, orientation, distance, colour, transparency and surface texture can all be changed in orderly (and disorderly) ways through incrementing along sequences of integers or reals.

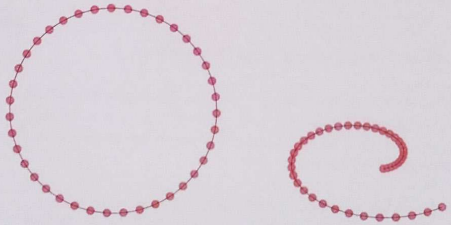
207

The samples in this pattern develop increasingly complex curves traced by a single point moving through space. Each successive sample increases both the number of parameters on which an increment applies and the complexity of the incrementing functions. Throughout each sample, the structure of the model remains constant; only the values of the parameters change.

Even a single point can demonstrate the basic structure of an increment. Start with a point in space, located as it must be with respect to a coordinate system.



The point can be thought of having either Cartesian (x, y, z) coordinates or cylindrical (r, θ, z), where r is the radius, θ is the azimuth angle and z is the height of the point. Use cylindrical coordinates and increment the azimuth angle θ to make the point trace out an arc. If the azimuth angle increments from 0° to 360° the arc becomes a circle. Increment the radius to turn the arc into a spiral.



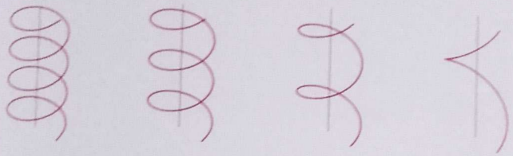
Incrementing the height of the point turns the arc into a helix and brings us to the first sample below.

INCREMENT Samples

Circular Helix

When. Move a point uniformly around a centre and upward in space.

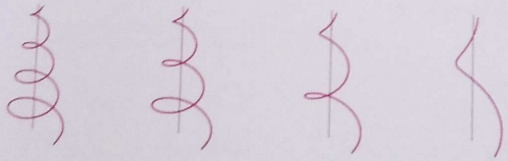
How. As a point moves around the circle, increment its height by a uniform amount. The result is to trace out a simple circular helix.



Conic Helix

When. Add a reducing radius increment to change a circular to a conic helix.

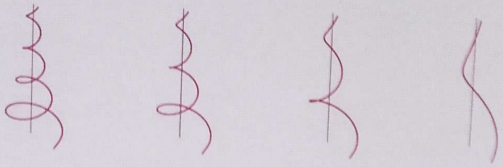
How. In addition to the two increments (angle and height) for a circular helix, reducing the radius incrementally from an initial value to a minimum value produces a conic helix, that is, a helix whose points lie on a cone.



Tapered Radius Spiral

When. Taper the radius of a conic helix to produce a spiral.

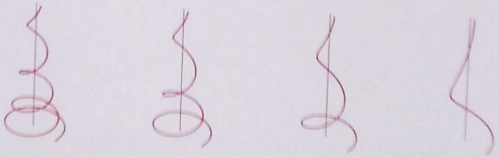
How. As a point on a conic helix moves upward its radius shrinks. The point can be imagined to have a parameter that is 1 at the helix base and 0 at the top. Squaring this parameter will still result in a series that goes from 1 to 0, but the series will taper across this interval. Mathematically, the curve changes from a helix to a spiral.



Tapered Height Spiral

When. Taper the height of a conic helix to produce a spiral.

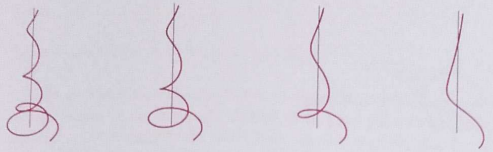
How. Instead of tapering the radius, taper the height with the same device, by squaring the parameter. In this case, the parameter is 0 at the helix base and 1 at the top. The helix, now a spiral, appears to have been differentially stretched from its base to its top.



Tapered Radius and Height Spiral

When. Combining increments yields unpredictable forms.

How. Combining both radius and height tapers can be done independently in the model. They do not affect each other computationally, but combine in the geometric result. They produce a spiral that would be hard to conceive of itself, but naturally emerges from the parameterization.

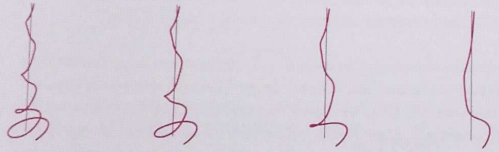


Elliptical Tapered Radius and Height Spiral

When. Change a circular spiral to an elliptical one.

How. In the prior samples, the radius, angle and height were independent in the model. In this sample, the radius becomes a function of the angle, by using a polar equation for the radius of an ellipse. If an ellipse has major axis of $r = 1$ and minor axis of $s = 0.5$, the radius as a function of θ is

$$\frac{rs}{\sqrt{r^2\cos^2\theta + s^2\sin^2\theta}} = \frac{0.5}{\sqrt{\cos^2\theta + 0.25\sin^2\theta}}$$



3 - Énoncés des modèles paramétriques réalisés au cours de MAN en 2015 et 2017 – TP1

Université de Liège
ARCHITECTURE
Modélisation Architecturale Numérique

Pierre LECLERCQ pierre.leclercq@ulg.ac.be
Med-Anis GALLAS ma.gallas@ulg.ac.be
Xavier LANGLOIS xavier.langlois@student.ulg.ac.be

A3 TP

COURS DE MODELISATION ARCHITECTURALE
TP - 3e BACH IR ARCHITECTES - 2014/15

MODELISATION PARAMETRIQUE [TP1]

Référence : Zubin Khabazi (http://www.morphogenesism.com/)

I. Objectifs

L'objectif de cet exercice est de tester le potentiel de construction géométrique de formes architecturales complexes proposé par le logiciel de modélisation paramétrique *Grasshopper*. Il s'agit d'analyser le projet, d'identifier les principaux paramètres contrôlant sa construction géométrique et enfin de proposer un scénario de modélisation mettant en œuvre les opérateurs géométriques intégrés au logiciel. Cet exercice introduit la dimension dynamique de la modélisation paramétrique qui permet d'explorer et d'évaluer rapidement plusieurs configurations architecturales possibles.

II. Scénario

Pour ce TD, le scénario est le suivant :

- forme végétale : le cornichon
- son interprétation : La Gherkin Tour de N.Foster

Les objets et paramètres à manipuler :

- des cercles: duplication, révolution et subdivision,
- des courbes: création et duplication
- des polygones: création et extrusion

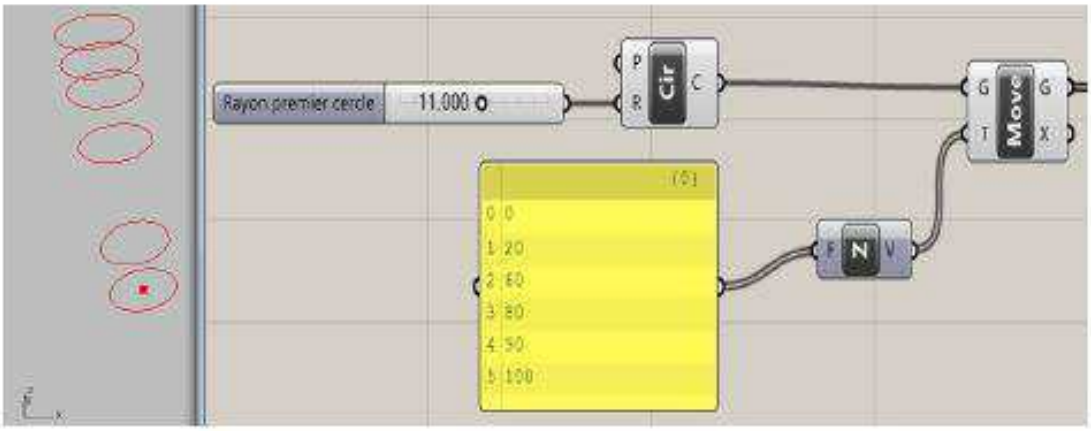
III. Dessiner la forme globale de la tour

Objectif : Une révolution de surface pour créer la forme globale de la tour et pouvoir modifier ses caractéristiques (hauteur, largeur...)

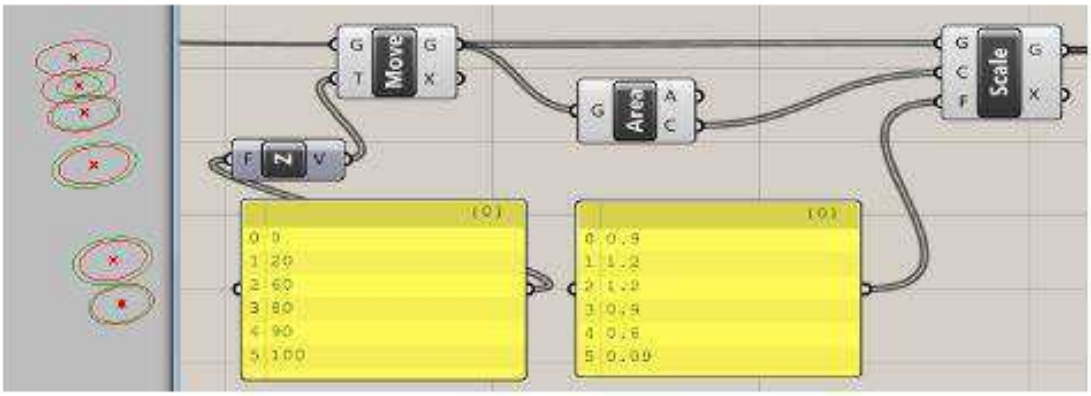
1. Pour commencer créons un ensemble de cercles qui décrivent la section de la tour à différents niveaux. Les composants *<Curve-Primitive-Circle>* et *<Params-*

Input-Number Slider> permettent de dessiner la base de la tour et de définir son rayon.

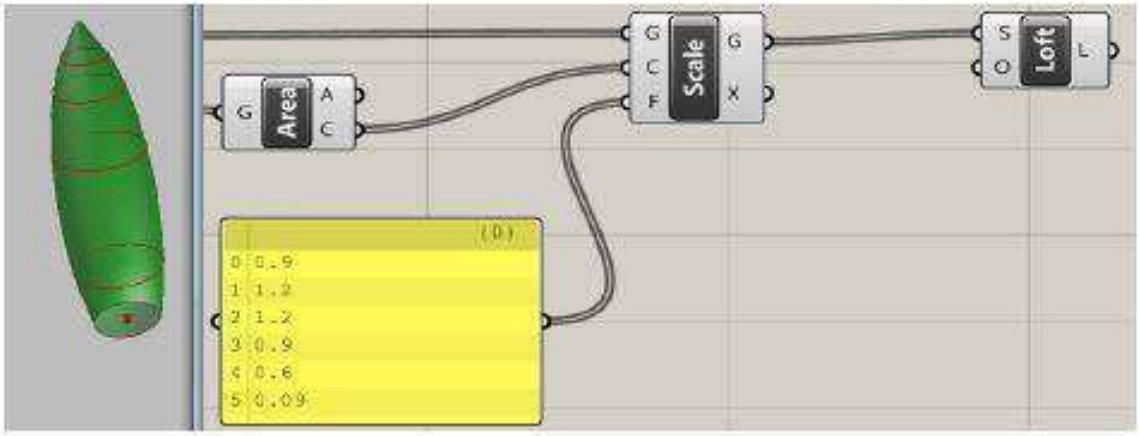
2. Dupliquons ce cercle avec le composant <*move*> selon la direction verticale en utilisant le composant <*Vector-Vector-Unit Z*>. Le vecteur doit être connecté à un composant de type <*Params-Input-Panel*>. Pour définir les différentes valeurs de translation, Il faut saisir les valeurs *(0, 20, 60, 80, 90, 100) dans le composant* <*Panel*> et ensuite effectuer un clic droit sur ce dernier pour choisir l'option <*Multiline Data*>.



3. Pour définir la forme profilée de la tour (non cylindrique), appliquons un opérateur de mise à l'échelle <*Transform-Affine-Scale*> aux différents cercles créés. Comme pour la translation, cette opération demande des valeurs d'échelles. Le composant <*Scale*> demande aussi le centre des géométries à redimensionner d'ou l'utilisation de composant <*Surface-Analysis-Area*> permettant de récupérer le centre des différents cercles (sortie <*C*>). Les facteurs d'échelles sont définis par le composant <*Params-Input-Panel*> en suivant le même schéma que pour les valeurs de translation.

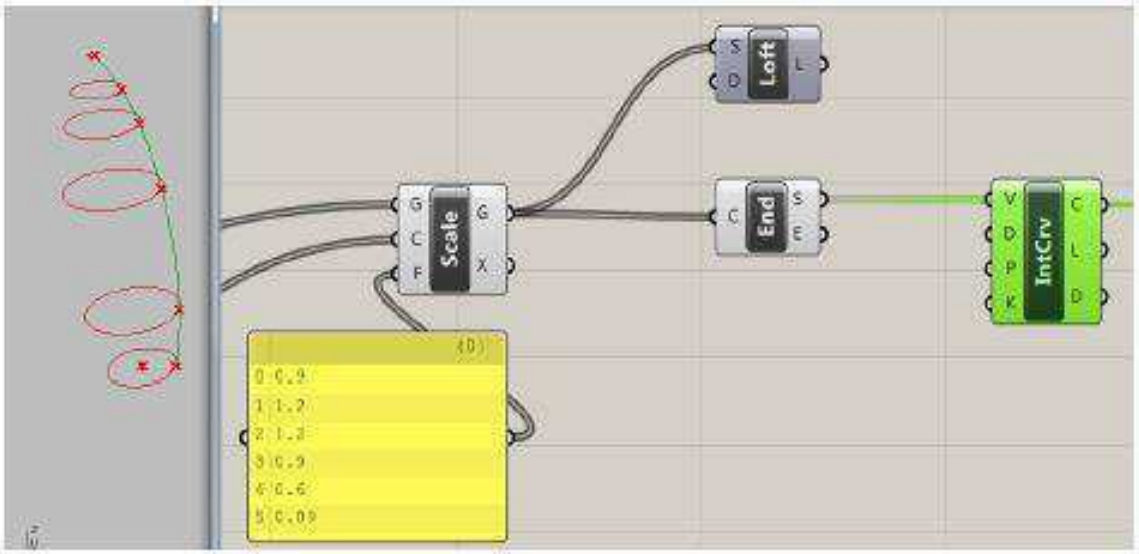


4. La dernière étape consiste à créer une surface avec le composant *<Surface-Freedom-Loft>* qui relie tous les cercles créés.

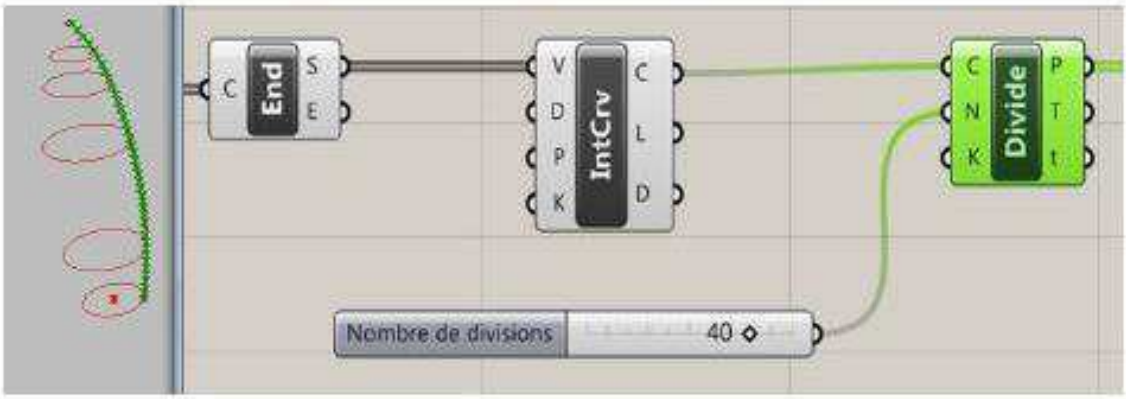


IV. Création des châssis de revêtement de la tour.

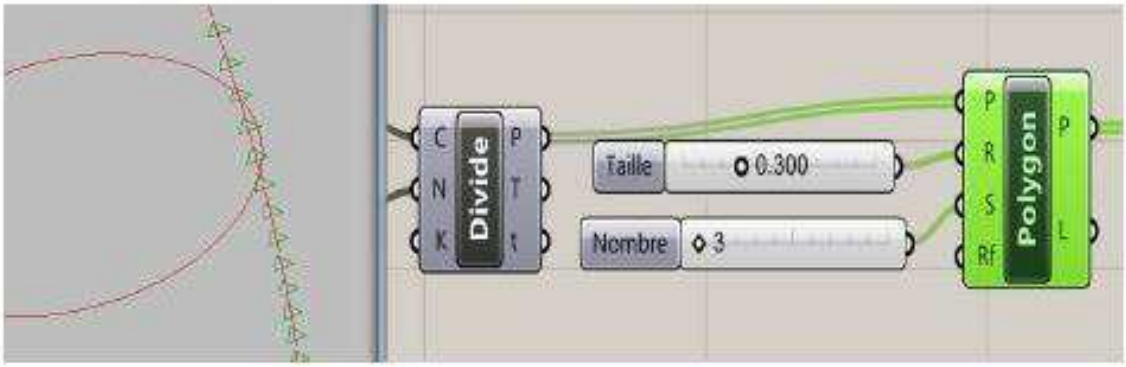
1. L'objectif est de créer des courbes qui parcourent la peau de la tour (créer des spirales). Pour commencer, cherchons le point de départ/d'arrivé de chacun des cercles en utilisant le composant *<Curve-Analysis-End Points>*. Ensuite, créons une première courbe qui représente le profil de la tour en connectant les sommets des cercles (sortie *<S>* du composant *<End>*) avec l'entrée *<V>* du composant *<Curve-Spline-Interpolate>*.



2. Divisons cette courbe en 40 parties repérées par des points sur les quelles nous positionnons des polygones. La courbe est divisée en utilisant le composant <*Curve-Division-Divide Curve*>. Le nombre de divisions est fixé par un composant <*Params-Input-Number Slider*> générant des valeurs de type entier.

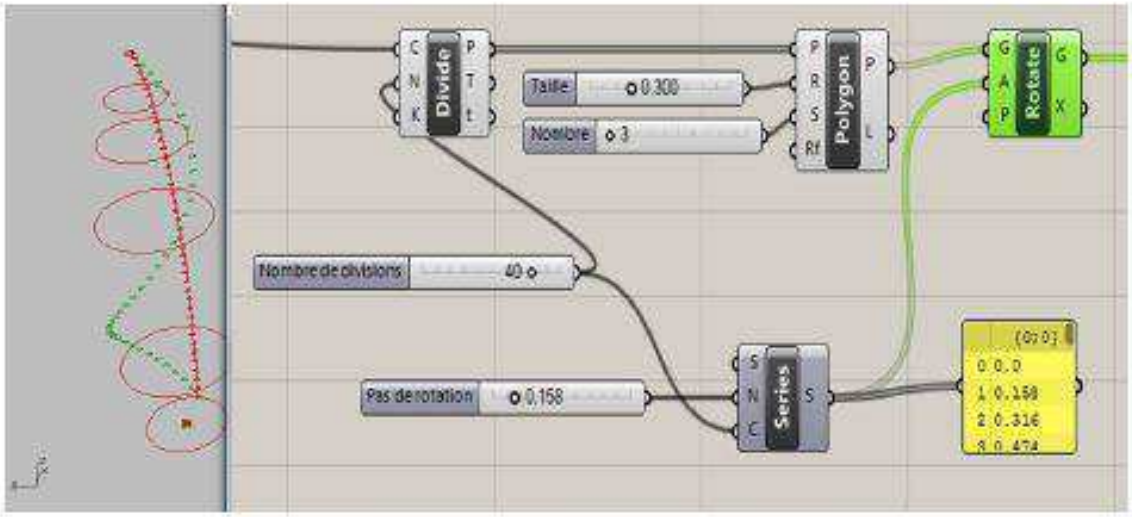


3. Le profil des châssis de revêtement est défini à partir de polygones. Utilisons le composant <*Curve-Primitive-Polygon*> pour créer ces polygones. Connectons le résultat de la division de la courbe à l'entrée <*P*> de ce composant pour définir les plans de positionnement des ces polygones. La taille du polygone <*R*> et le nombre de segments <*S*> sont définis par des composants de type <*Params-Input-Number sliders*> (le premier est de type réel et le deuxième de type entier).

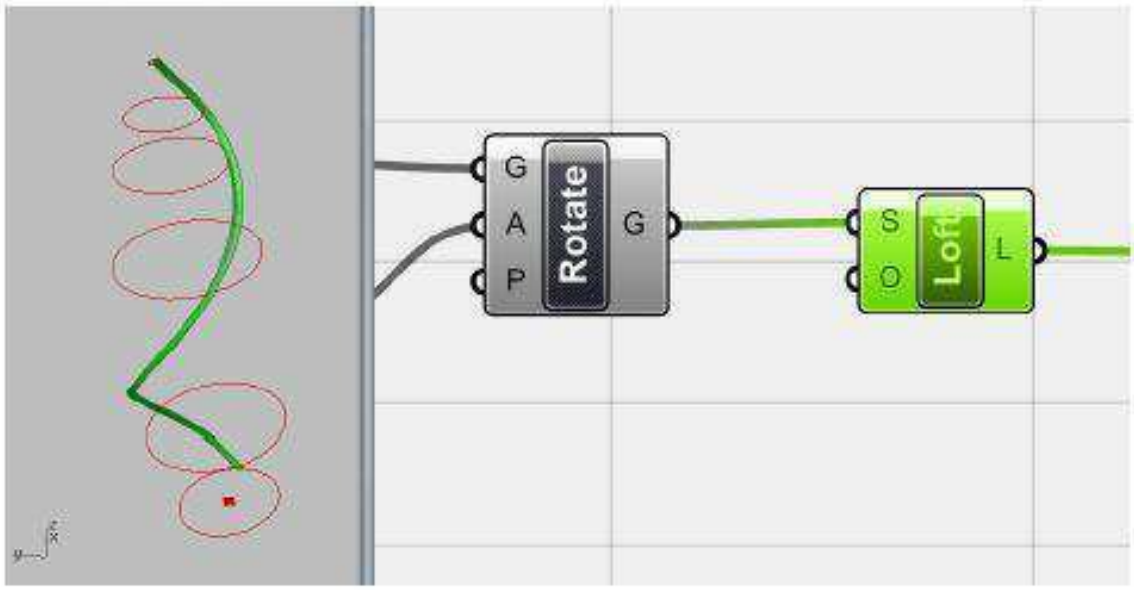


4. Les Châssis de la façade couvrent la surface totale de la peau de la tour. Pour arriver à ce résultat, utilisons le composant <*Transform-Euclidean-Rotate*>. Il faut connecter le l'entrée <*G*> du composant <*Rotate*> à la sortie <*P*> du composant <*Polygon*>. L'entrée <*A*> est liée à une série de valeurs correspondant à un angle de rotation en radian. Pour définir ces valeurs on utilise le composant <*Sets-Sequences-Series*> qui permet de créer une série de valeurs avec un pas bien déterminé. Dans ce cas la valeur de départ (entrée <*S*>) est de 0 (valeur par défaut, pas besoin de définir une valeur), le pas <*N*> est de 0,158 (valeur définie par un composant

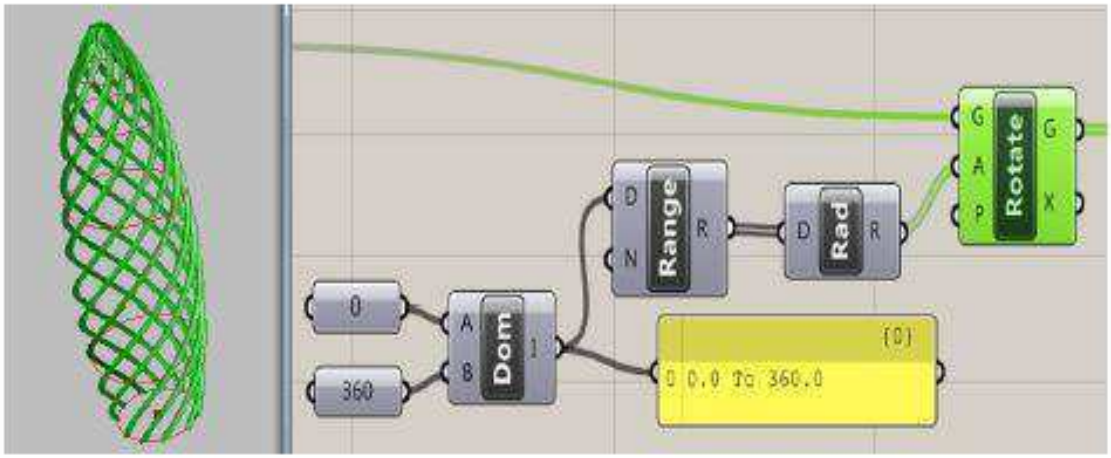
<Params-Input-Number Slider>) enfin le nombre totale de valeur *<C>* est le même que le nombre de subdivision utilisé pour diviser la courbe (entrée connectée au composant *<Nombre de division>*). Ce composant permet d'obtenir une série de 40 valeurs dont la première est 0 avec un pas d'avancement de valeur égale à 0.158.



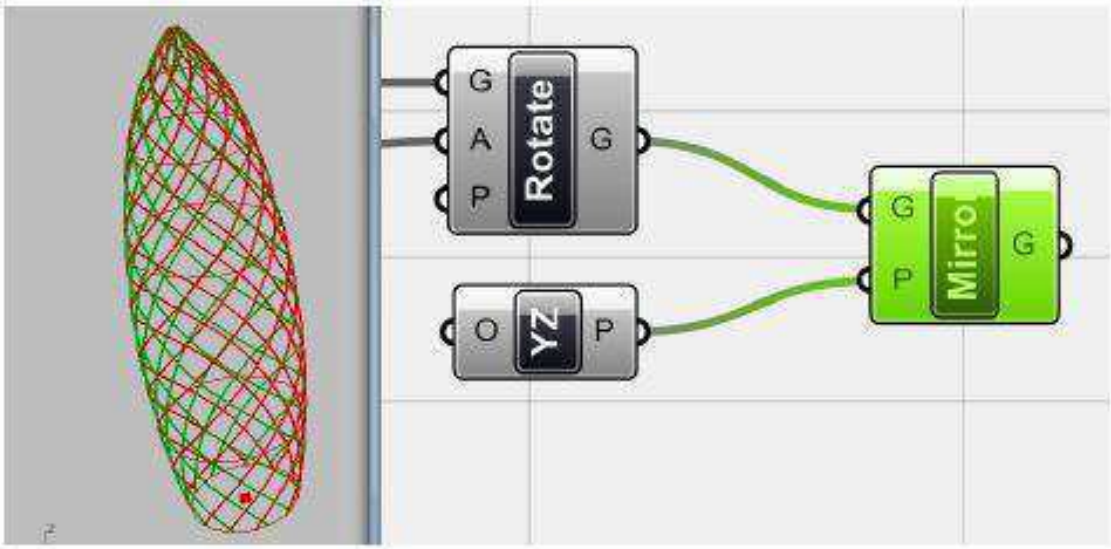
5. L'étape suivante consiste à lier ces différents polygones en utilisant le composant *<Surface-Freeform-Loft>* créant ainsi le volume d'un châssis.



6. Pour couvrir la totalité du volume de la tour, nous recopions cette géométrie 10 fois autour de la base de la tour. Nous utilisons le composant *<Transform-Euclidean-Rotate>* pour copier cette géométrie en effectuant à chaque fois une rotation d'un angle bien déterminé. Cet angle est défini par le composant *<Sets-Sequence-Range>* relié au composant *<Math-Domain-Construct Domain>* qui permet de créer une plage de valeurs variant de 0 à 360°. Comme les angles de rotation doivent être en Radian, nous utilisons le composant *<Math-Trig-Radians>* pour convertir les valeurs du Degré au Radian.

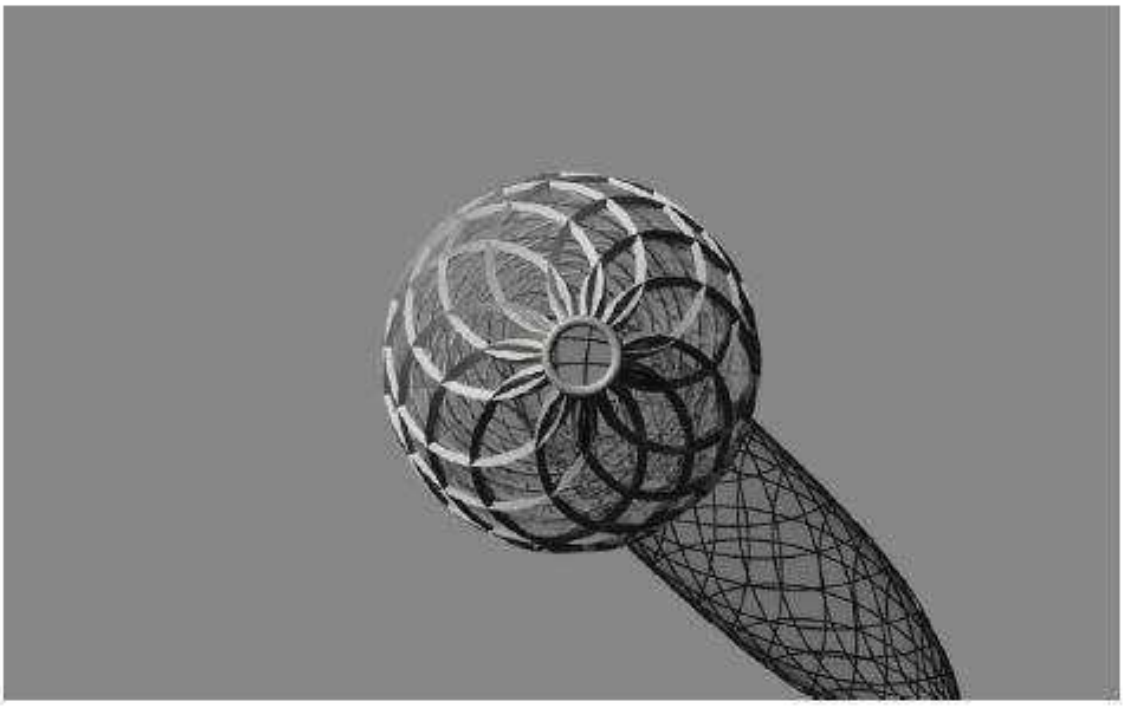


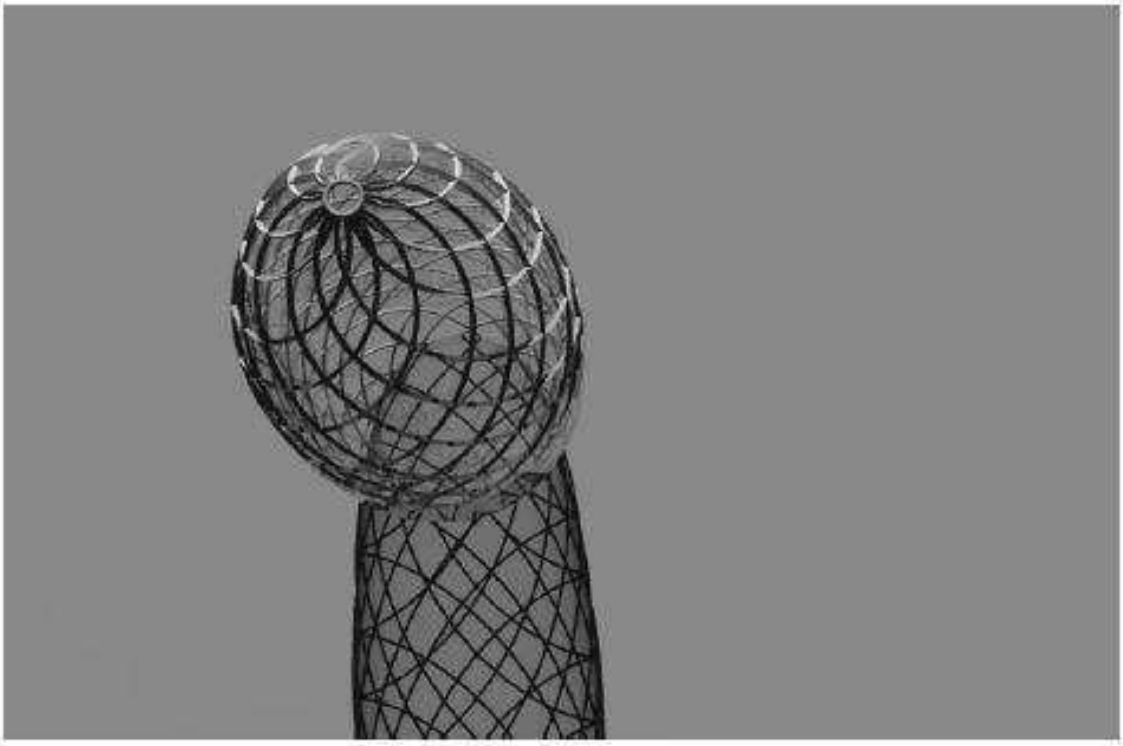
7. La dernière étape consiste à refaire la même opération mais dans le sens inverse afin de créer le croisement de maillage de la façade. Cette opération consiste à utiliser le composant *<Transform-Euclidean-Mirror>*. Il faut choisir le plan *<Vector-plane-YZ Plane>* comme plan de référence.



Initation à la Modélisation Paramétrique

V. Le résultat





3 - Énoncés des modèles paramétriques réalisés au cours de MAN en 2015 et 2017 – TP2

Université de Liège
ARCHITECTURE
Modélisation Architecturale Numérique

Pierre LECLERCQ pierre.leclercq@ulg.ac.be
Med-Anis GALLAS ma.gallas@ulg.ac.be
Xavier LANGLOIS xavier.langlois@student.ulg.ac.be

A3 TP

COURS DE MODELISATION ARCHITECTURALE
TP - 3e BACH IR ARCHITECTES - 2014/15

MODELISATION PARAMETRIQUE [TP2]

I. Objectifs

L'objectif de cet exercice est de tester les fonctions de traitements des surfaces proposées par le logiciel de modélisation paramétrique *Grasshopper*. Il s'agit de créer un maillage de type Voronoï, de l'appliquer sur des surfaces et enfin de lui donner une dimension architecturale (création de volumes).

II. Scénario

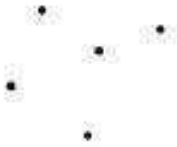
Pour ce TD, le scénario est le suivant :

- une surface *nurbs* constituée à partir de deux courbes sur laquelle nous projetons un diagramme de Voronoï

Les objets et paramètres à manipuler :

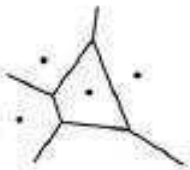
- des courbes: duplication, révolution, subdivision, extrusion et projection.

III. Le diagramme de Voronoi



1. A partir d'une série de points

2. Connection des points les plus proches



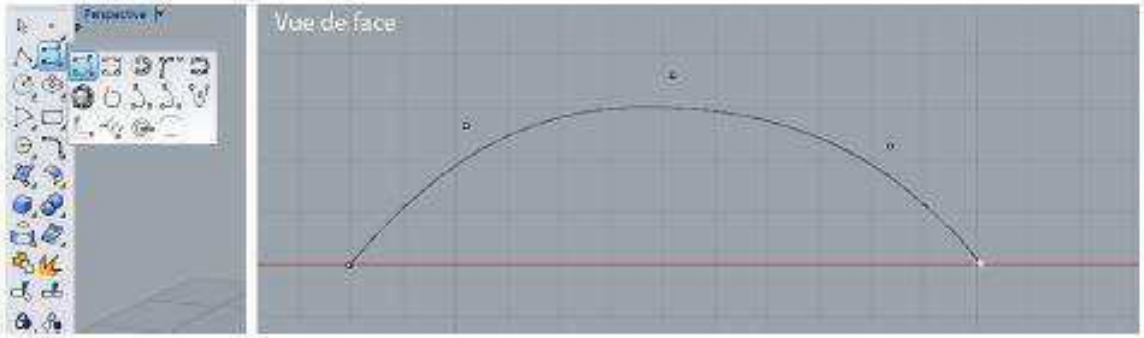
3. Construction de la médiatrice de chaque segment

4. Les médiatrices définissent ma cellule

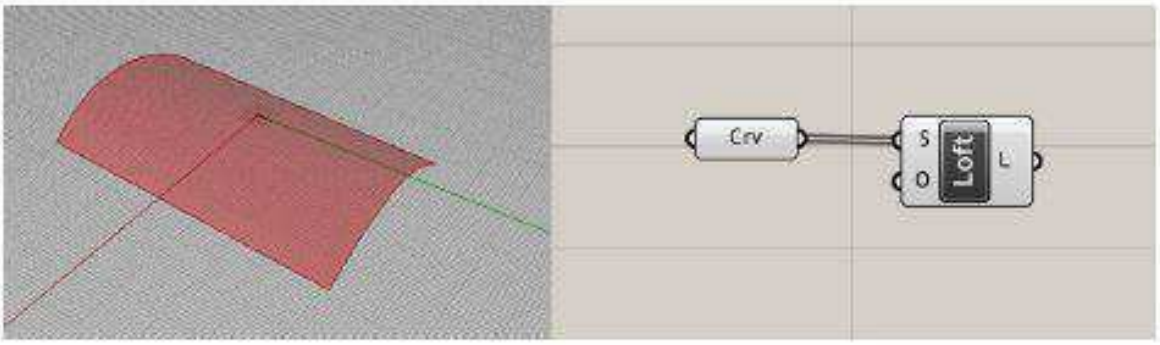
IV. Créer un digramme de Voronoï

Objectif : Créer un diagramme de Voronoï et le projeter sur une surface *nurbs* afin de créer un maillage.

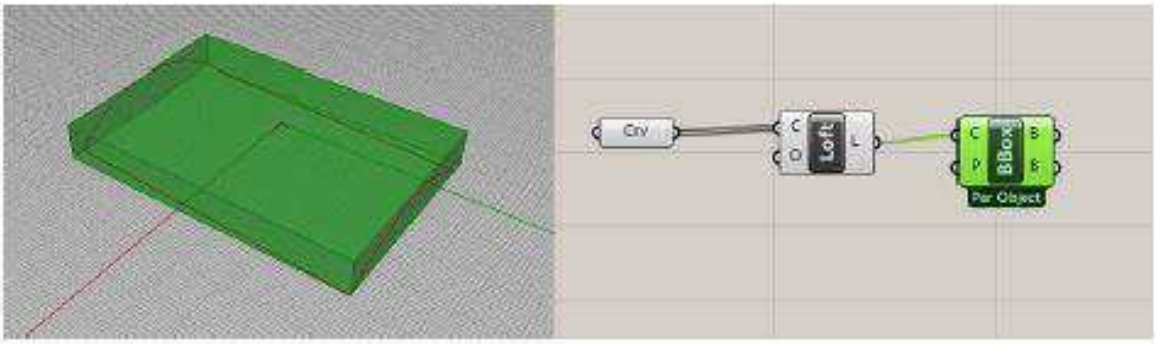
1. Pour commencer, crèons deux courbes sous *Rhino*.



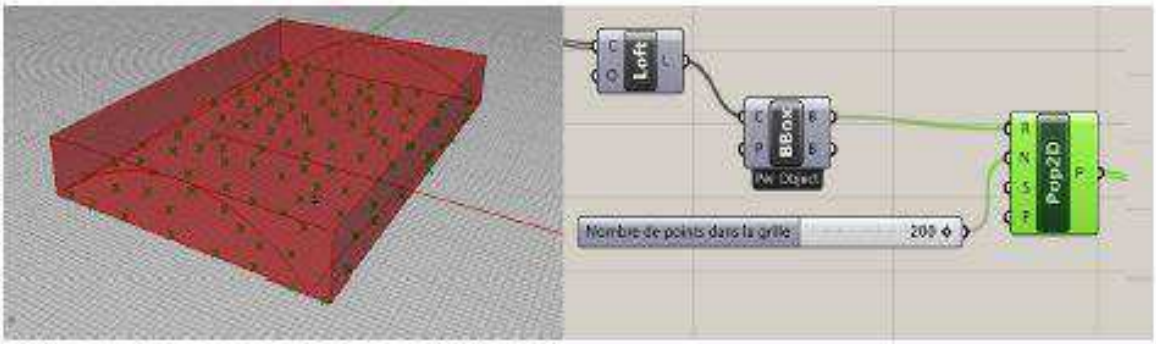
2. Les courbes sont intégrées dans *Grasshopper* grâce au composant <*Params-Geometry-Curve*>. Pour réaliser cette opération, Clic droit sur le composant <*Crv*>, choisir l'option <*Set Multiple Curves*> et enfin sélectionner (sous *Rhino*) les deux courbes. Le composant <*Crv*> change de couleur indiquant ainsi que les deux courbes sont chargées. Nous créons ensuite une surface *Nurbs* qui relie ces deux courbes. La surface est créée en utilisant le composant <*Surface-Freedom-Loft*>.



3. Créons une boite qui englobe cette surface en utilisant le composant *<Surface-Primitive-Bounding box>* dont l'entrée <C> est connectée à la sortie *<L>* du composant *<Loft>*.

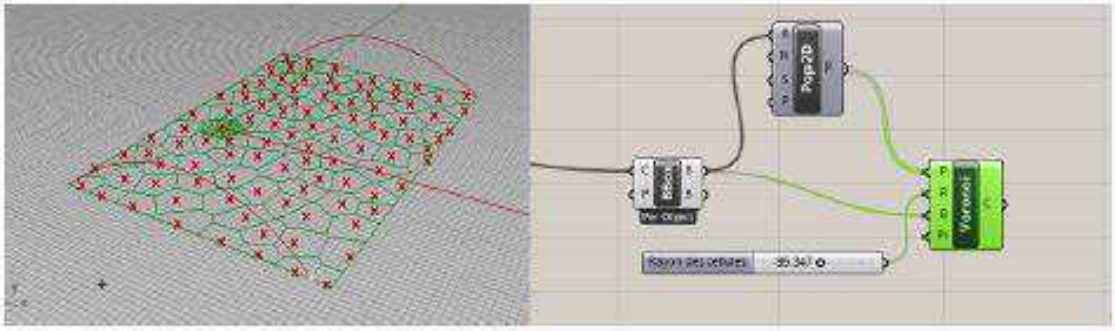


4. L'étape suivante consiste à créer une grille de points sur toute la base de cette boite englobante. Nous utilisons le composant *<Vector-Grid-Populate 2D>*. Le nombre de points constituant la grille est défini par un composant *<Params-Input-Number Slider>*.



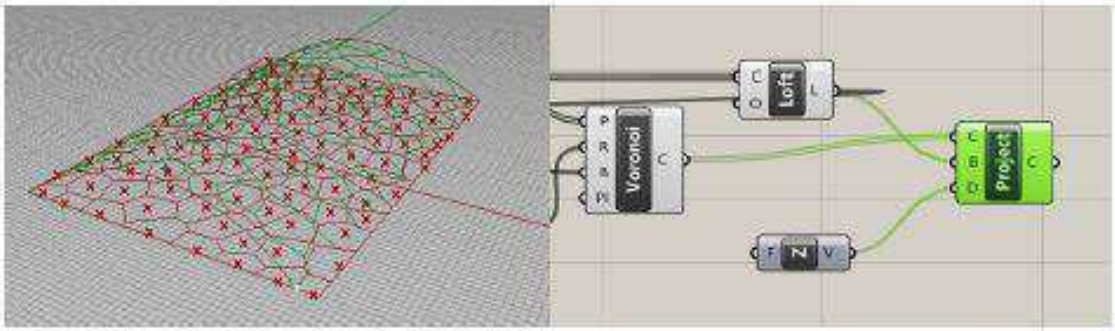
5. Cette grille de points est utilisée pour générer le diagramme de Voronoï grâce au composant *<Mesh-Triangulation-Voronoi>*. L'entrée *<P>* de ce composant est connectée à la sortie *<P>* du composant *<Pop2D>* intégrant ainsi la liste des points de base pour la création du diagramme de Voronoï. Le rayon des cellules à créer (entrée *<R>* du composant *<Voronoï>*) est défini par un composant le *<Params-Input-Number Slider>* intégrant des valeurs de type réel. Le digramme à générer est

délimité par la boite englobante créée par le composant <*BBox*> ainsi nous connectons la sortie <*B*> de ce dernier à l'entrée <*B*> du composant <*Voronoï*>.



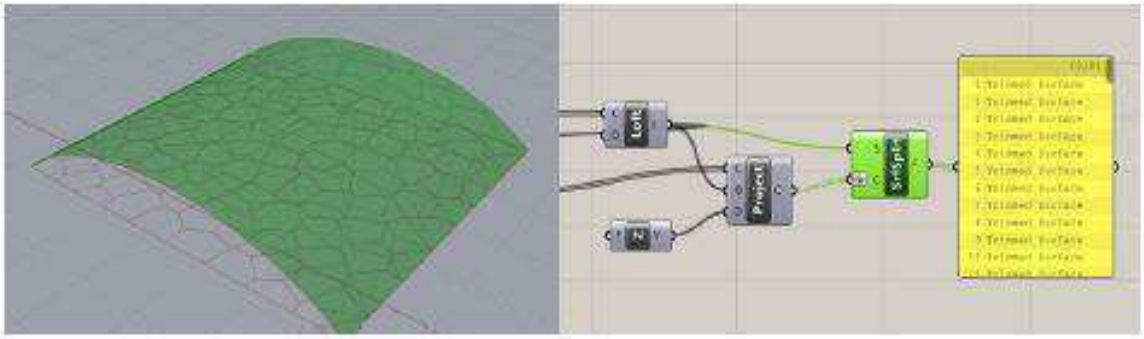
V. Projeter et exploiter le diagramme de Voronoï

1. L'étape suivante consiste à projeter ce diagramme sur la surface *nurbs* en utilisant le composant <*Curve-Util-Project*>. L'entrée <*C*> de ce dernier est connectée à la sortie <*C*> du composant <*Voronoï*>, l'entrée <*B*> est connectée à la sortie <*L*> du composant <*Loft*> intégrant ainsi les courbes à projeter et la surface de destination. L'entrée <*D*> du composant <*Project*> est connectée au composant <*Vector-Vector-Unit Z*> définissant la direction de cette projection.

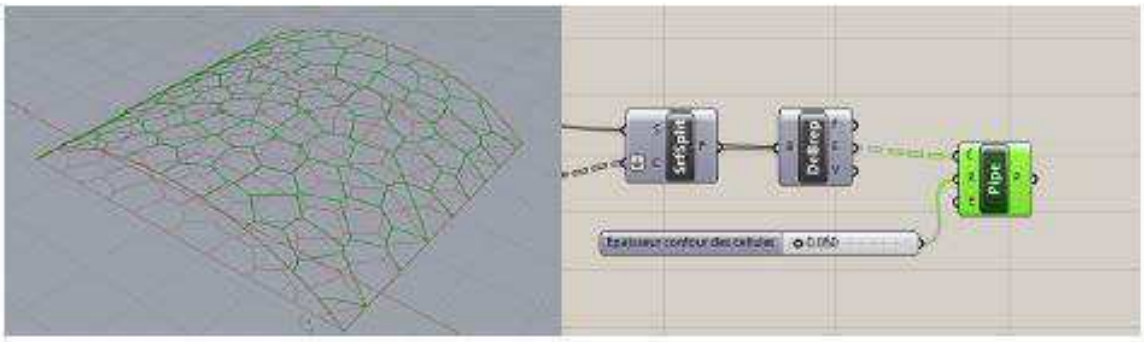


2. Le diagramme projeté nous permet de découper la surface de base en surfaces de subdivisions qui correspondent aux limites des cellules de Voronoï générées. Utilisons le composant <*Intersect-Physical-Surface Split*> pour créer ces surfaces de subdivisions. L'entrée <*S*> de ce composant est connectée à la sortie <*L*> du composant <*Loft*>. L'entrée <*C*> du composant <*SrfSplit*> est reliée à la sortie <*C*> du composant <*Project*>.

Afin de normaliser la structure des données (même structure de données entre les deux entrées <*S*> et <*C*> du composant <*SrfSplit*>), clic droit sur cette entrée <*C*> et choisir l'option <*Flatten*> (flèche vers le bas).

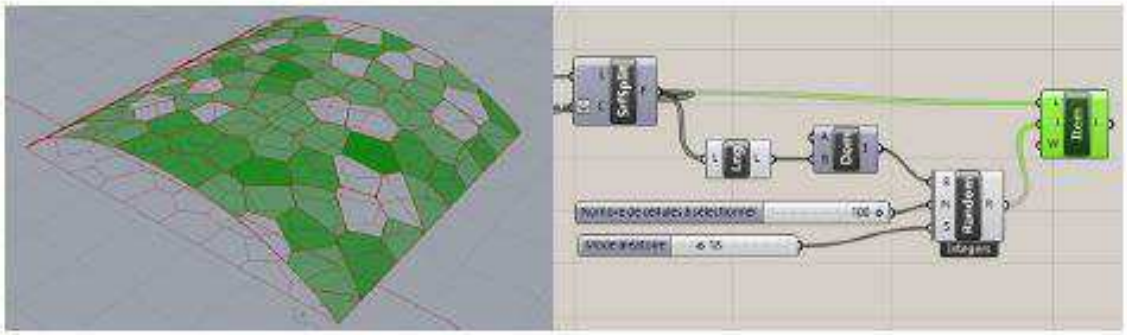


3. Pour affirmer les contours de ces surfaces de subdivisions, nous utilisons le composant *<Surface-Freeform-Pipe>*. Ce dernier permet de créer des volumes cylindriques (très faible épaisseur) qui marquent les limites des surfaces de subdivisions générées. Le rayon de ce volume est défini à partir du composant *<Params-Input-Number Slider>* qui est connecté à l'entrée *<R>* du composant *<Pipe>*.

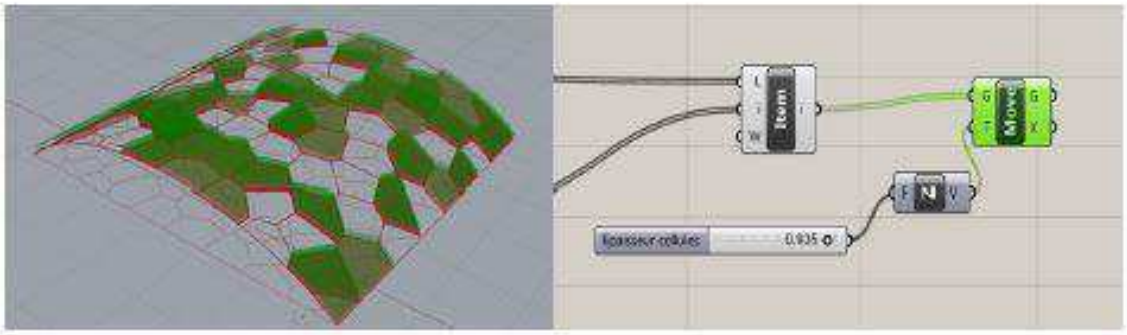


4. Cette étape vise à sélectionner une partie des surfaces de subdivisions créées selon un mode aléatoire. Afin de définir ce mode de sélection aléatoire, nous utilisons un composant *<Sets-List-List Item>* permettant de sélectionner un ou plusieurs éléments d'une liste. La sélection se fait en saisissant l'identifiant des éléments à choisir. Pour créer ces identifiants (liste d'entiers), nous commençons par chercher le nombre total de surfaces de subdivision en utilisant le composant *<Sets-List-List Length>*. Ce dernier est connecté à la sortie *<F>* du composant *<SfrSplit>*. Définissons ensuite un domaine dont les limites varient entre *0* et le nombre total de surfaces de subdivisions. Ce domaine est créé par le composant *<Maths-Domain-Construct Domain>*. Nous créons ensuite une liste de valeurs de type entier de manière aléatoire. Cette liste est générée par le composant *<Sets-Sequence-Random>*. Le nombre de valeurs à créer est défini par un composant *<Params-Input-Number Slider>* (valeur de type entier) qui est connecté à l'entrée *<N>* du composant *<Random>*. Pour générer une liste aléatoire de valeurs de type entier, il faut effectuer un clic droit sur le composant *<Random>* et choisir l'option *<Integer Numbers>*. Nous pouvons changer de mode aléatoire en connectant un composant *<Params-Input-*

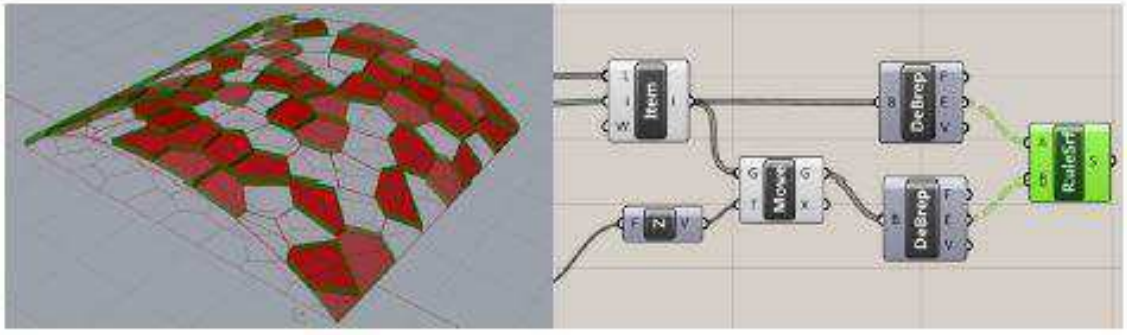
Slider Number> (valeurs de type entier variant de 1 à 100) à l'entrée <*S*> du composant <*Random*>.



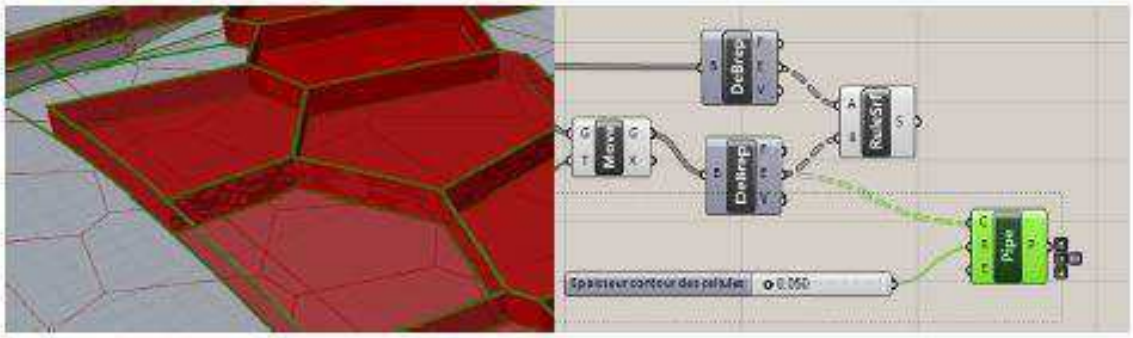
5. Pour créer des volumes à partir des surfaces de subdivisions sélectionnées, nous utilisons le composant <*Transform-Euclidean-Move*>. L'entrée <G> du composant <*Move*> est connectée à la sortie <*I*> du composant <*Item*>. Pour donner une direction à cette translation, nous utilisons le composant <*Vector-Vector-UnitZ*>. La valeur de la translation est définie par un composant <*Params-Input-Number Slider*>.



6. Pour créer l'enveloppe des ces volumes (relier les deux niveaux de surfaces), nous utilisons deux fois le composant <*Surface-Analysis-Deconstruct Brep*> qui permet d'isoler les arrêtes des surfaces de subdivisions (supérieures et inférieures). Le composant <*Surface-Freeform-Ruled Surface*> permet de relier ces bords par des surfaces enveloppantes.

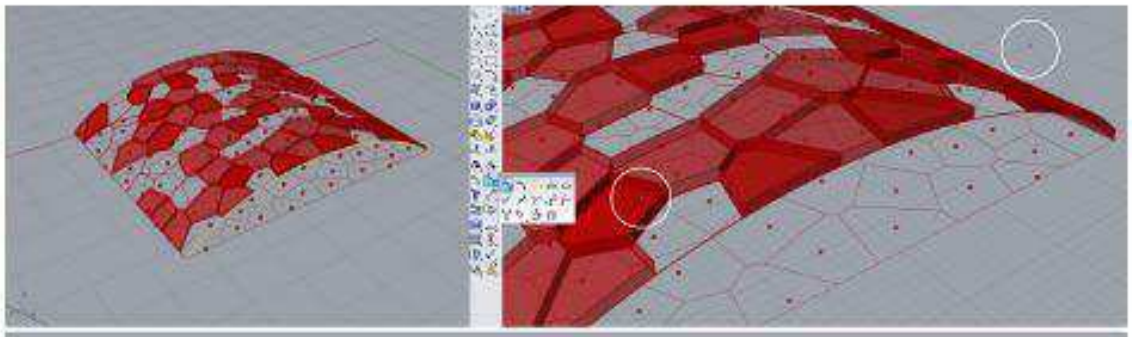


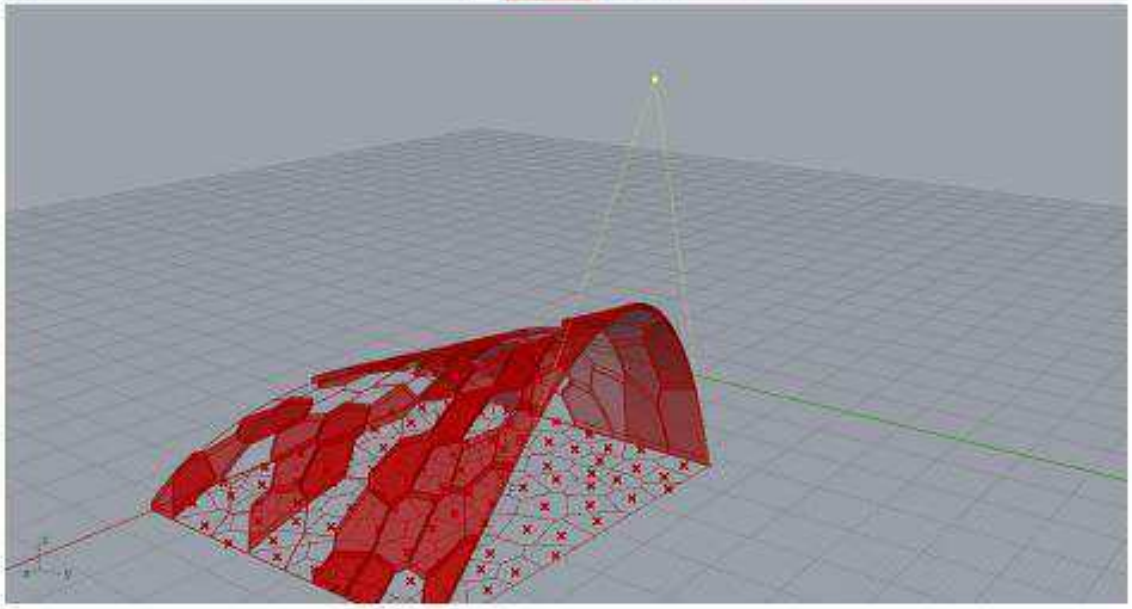
7. Pour marquer les volumes créés à partir des cellules de Voronoï, nous utilisons le composant <*Surface-Freeform-Pipe*> dont l'entrée <*C*> est connectée à la sortie <*E*> du composant <*DeBrep*>.



VI. Varier les paramètres et modifier la forme de base

Nous pouvons modifier les paramètres portant sur le nombre de points structurant le diagramme de Voronoï et les paramètres de sélection aléatoire des surfaces de subdivision. Nous pouvons aussi modifier les propriétés des courbes de base et générer ainsi d'autres maillages.





3 - Énoncés des modèles paramétriques réalisés au cours de MAN en 2015 et 2017 – TP3

Université de Liège
ARCHITECTURE
Modélisation Architecturale Numérique

Pierre LECLERCQ pierre.leclercq@ulg.ac.be
Med-Anis GALLAS ma.gallas@ulg.ac.be
Xavier LANGLOIS xavier.langlois@student.ulg.ac.be

A3 TP

COURS DE MODELISATION ARCHITECTURALE
TP - 3e BACH IR ARCHITECTES - 2014/15

MODELISATION PARAMETRIQUE [TP3]

I. Objectifs

L'objectif de cet exercice est de tester le potentiel de traitement de surfaces proposé par le logiciel de modélisation paramétrique *Grasshopper*. Il s'agit d'imaginer des types de traitements de surfaces de couverture ou d'enveloppes, d'identifier les principaux paramètres contrôlant leur construction géométrique et enfin de proposer un scénario de modélisation mettant en œuvre les opérateurs géométriques intégrés au logiciel. Cet exercice propose d'expérimenter une deuxième facette de la dimension dynamique de la modélisation paramétrique. Une facette qui porte sur l'exploration rapide de plusieurs configurations et la réutilisation du même modèle pour différents cas d'application.

II. Scénario

Pour ce TD, le scénario est le suivant :

- une surface *nurbs* constituée à partir de deux courbes sur laquelle nous testons plusieurs types de maillages.

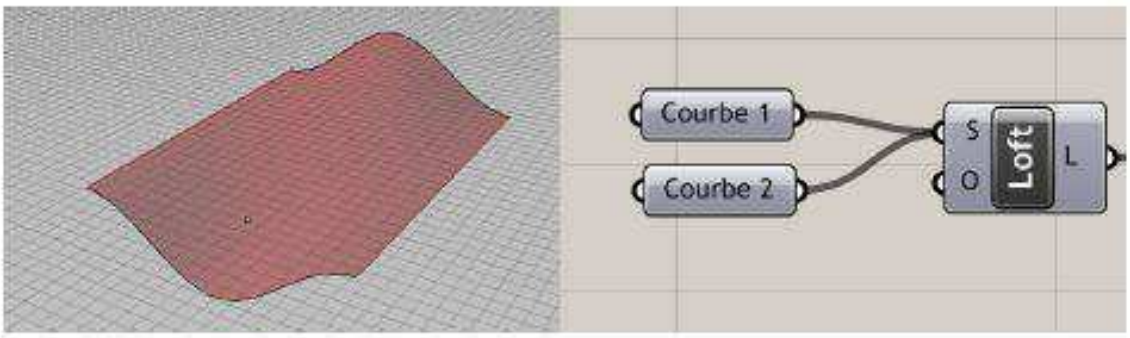
Les objets et paramètres à manipuler :

- des courbes : duplication, révolution, subdivision, extrusion et projection.

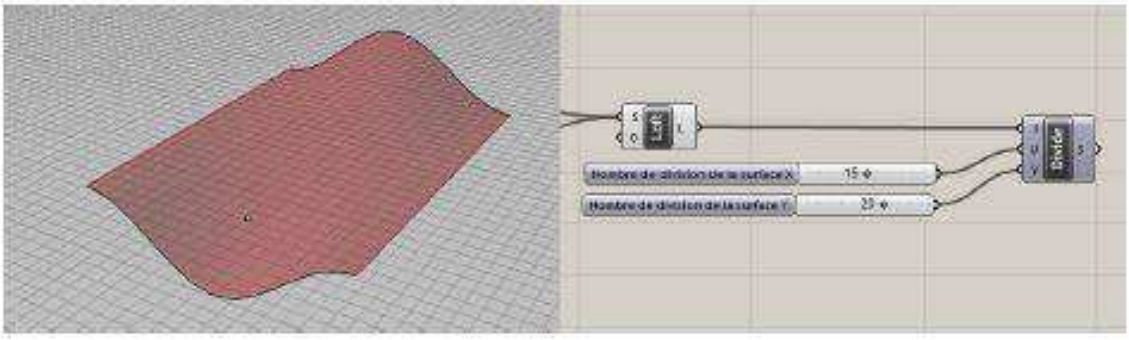
III. Créer des surfaces de subdivisions

Objectif : Tester plusieurs types de maillages sur une surface *nurbs*:

1. Pour commencer, créons deux courbes sous *Rhino* pour générer une surface *nurbs*. Cette surface peut être créée en utilisant le composant *<Surface-Freedom-Loft>*.



2. Nous subdivisons cette surface suivant deux directions u et v pour générer un ensemble de segments dans ces deux directions. Le composant *<Math-Domain-Divide domain2>* est connecté à deux composants de type *slider* afin de définir le nombre de segments dans chaque direction.

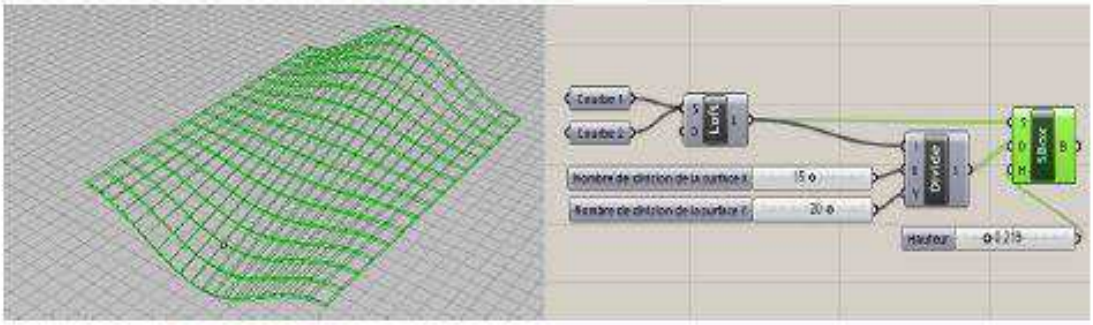


3. Ces segments permettent de créer des surfaces de subdivision utiles pour différents types de traitements de surface. Pour effectuer cette action, nous utilisons le composant *<Surface-Util-Isotrim>* qui est connecté à la première surface créée (sortie *<L>* du composant <Loft>) à l'ensemble des segments générés (sortie *<S>* du composant *<Divide>).*

I. Créer un traitement de surfaces modulaire

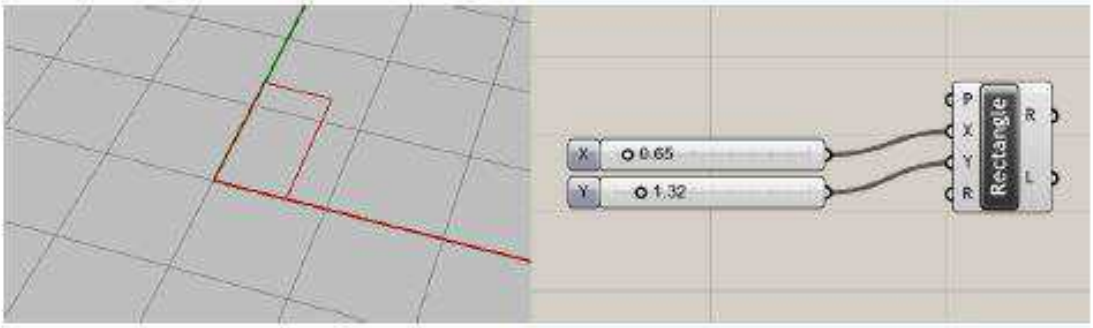
Nous utilisons ce maillage (surfaces de subdivisions) pour construire un traitement modulaire de la surface. Ce traitement utilise deux types de modules répartis sur l'ensemble de la surface selon une condition.

1. Nous commençons par créer des volumes à partir des surfaces de subdivisions créées auparavant. Pour réaliser cette opération, nous utilisons le composant *<Transform-Morph-Surafce Box>* dont l'entrée *<S>* est connectée à la sortie *<L>* du composant *<Loft>*. L'entrée *<D>* est connectée à la sortie *<S>* du composant *<Divide>*. L'épaisseur donnée à ces différents volumes est définie par un composant de type *slider* (valeur de type réel variant de 0 à 1).

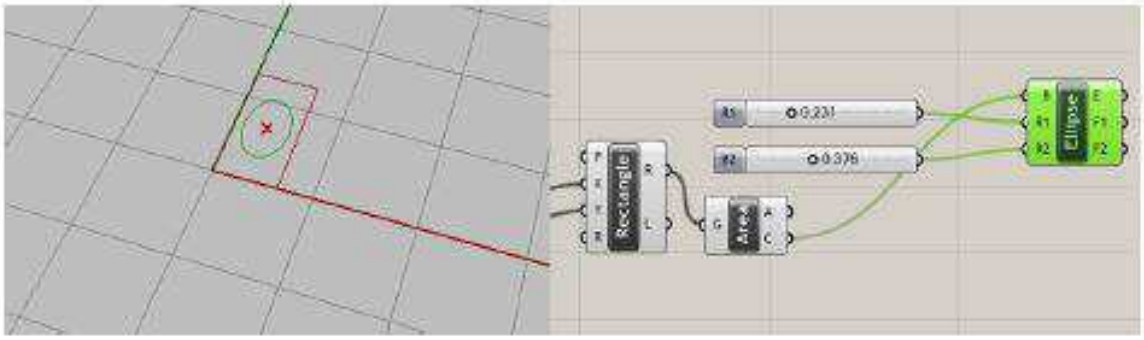


2. L'étape suivante consiste à créer les modules (deux types de modules avec des tailles d'ouvertures différentes) que nous plaquons ensuite sur les différents volumes de subdivisions.

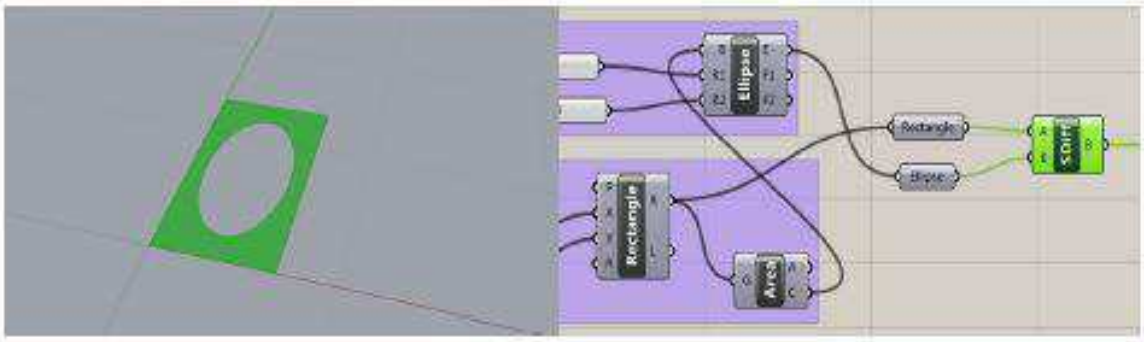
Pour créer le premier type de module, nous utilisons le composant *<Curve-Primitive-Rectangle>* dont les entrées *<X>* et *<Y>* sont connectées à des composants de type *slider* définissant ainsi les dimensions du rectangle de base.



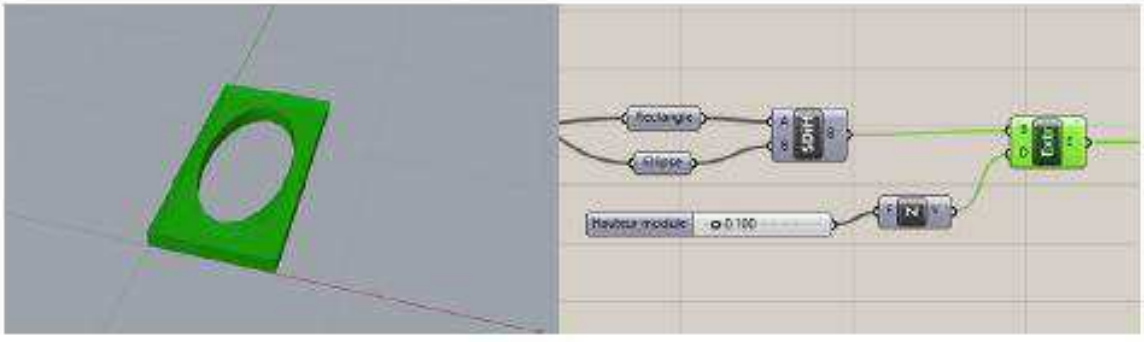
3. Ce module doit intégrer une ouverture de forme ovale. Cette ouverture est créée par le composant *<Curve-Primitive-Ellipse>*. L'entrée *<B>* de ce dernier est connectée au composant *<Surface-Analysis-Aerea>* qui permet de définir le centre du rectangle que nous avons créé. Les entrées *<R1>* et *<R2>* sont connectées à des composants de type *Slider* fixant les valeurs des rayons de l'ellipse.



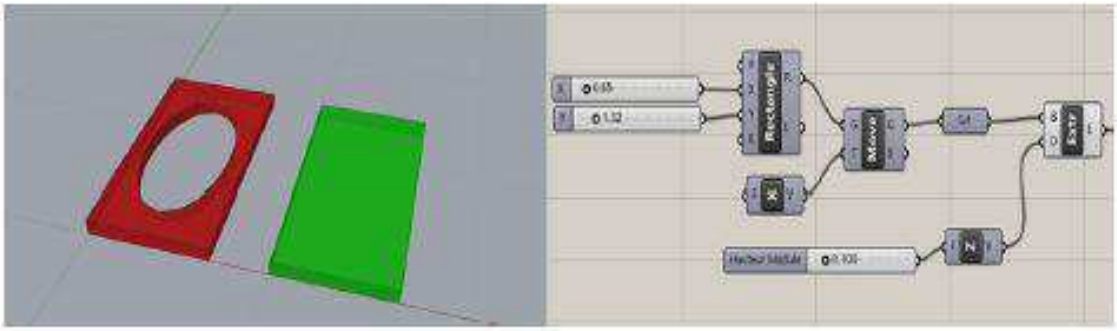
4. Le rectangle et l'ellipse sont connectés à un composant *<Params-Geometry-Surface>* pour créer des surfaces à partir de ces contours. Le composant *<Intersect-Shape-Solid difference>* qui permet de supprimer la zone en forme d'ellipse de la zone rectangulaire.



5. À partir de la surface obtenue, nous créons un volume en utilisant le composant *<Surface-Freeform-Extrude>* suivant une direction définie par un vecteur dans la direction de l'axe Z et une hauteur déterminée par un composant de type *<slider>*.

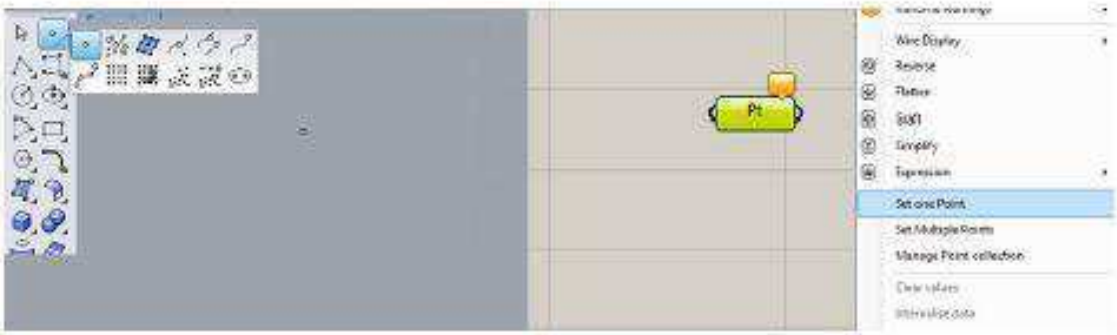


6. Pour le deuxième type de module, nous recopions les composants permettant de créer un module rectangulaire (même valeur d'extrusion) sans ouverture (module opaque).

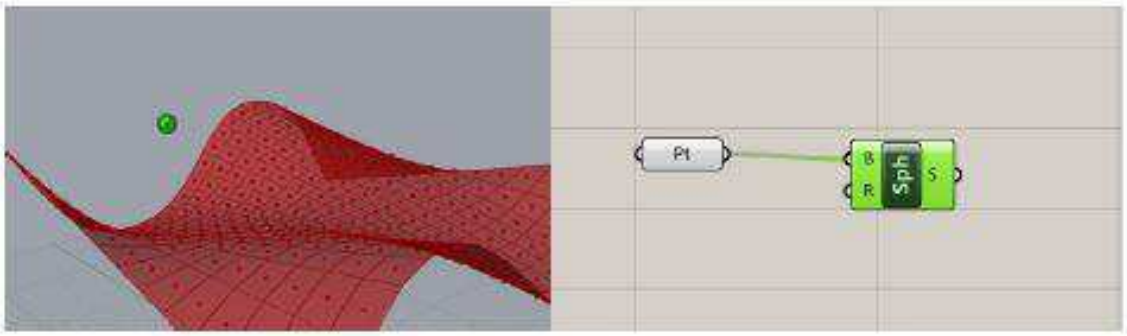


7. La phase suivante consiste à plaquer ces deux types de module sur la surface de base suivant une condition. Cette condition est matérialisée par la position d'un point de référence (notion d'*Attractor*) : si le volume de subdivision est proche du point de référence (distance < 20), ce dernier aura un module avec une ouverture, sinon il est totalement opaque.

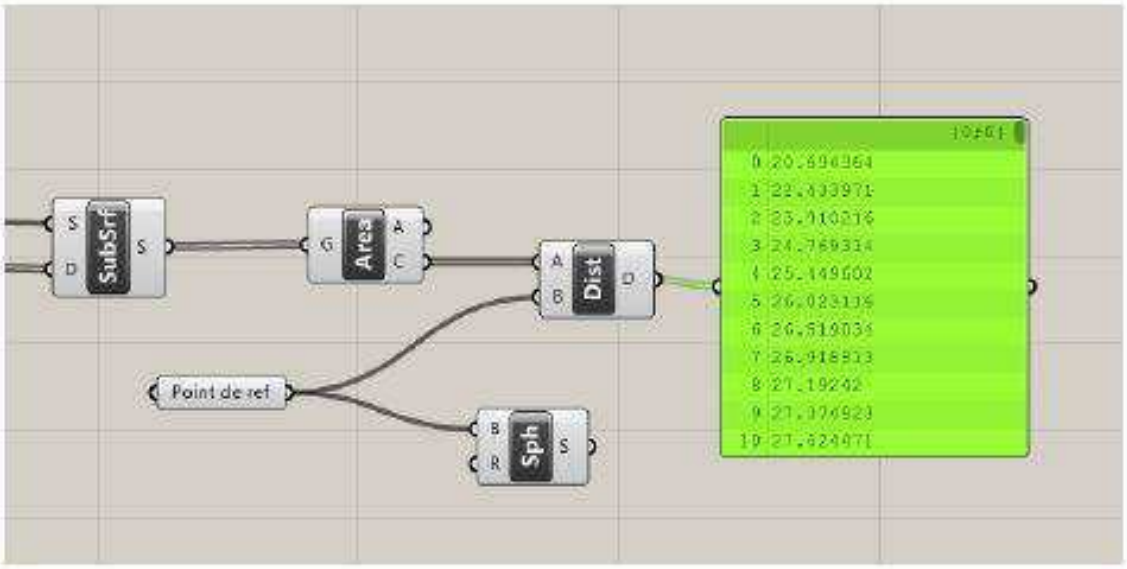
8. La première phase de ce plaquage de module consiste à créer le point de référence. Ce dernier est créé dans l'environnement *Rhinoceros* (plus d'interactivité dans les opérations de déplacement). Ce point est créé avec la fonction <Point Singulier> (voir la capture d'écran) et placé au dessus de la surface de base (surface *Nurbs*). Le point est ensuite chargé dans l'environnement *Grasshopper* pour l'intégrer dans notre module de traitement des surfaces. Cette opération est effectuée grâce au composant <*Params-Geometry-Point*>. Il faut effectuer un clic droit sur ce composant et choisir l'option <*Set One Point*>. Cette option renvoie vers la fenêtre de visualisation de *Rhinoceros* et permet de sélectionner le point de référence. Une fois le point chargé, le composant change de couleur (gris).



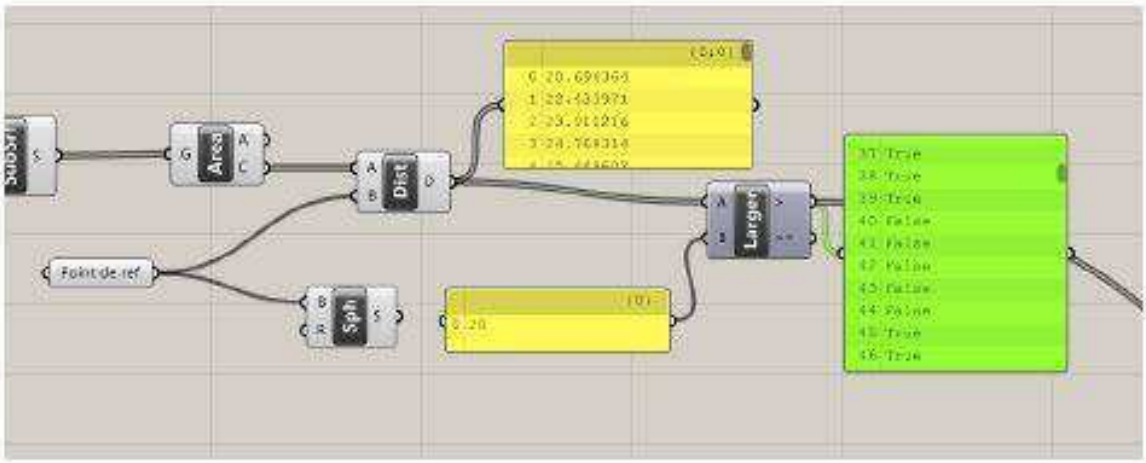
9. Nous créons ensuite une sphère pour mieux visualiser le point de référence. Cette sphère est créée en utilisant le composant <*Surface-Primitive-Sphere*> dont l'entrée <*B*> est connectée à la sortie du composant <*Pt*>.



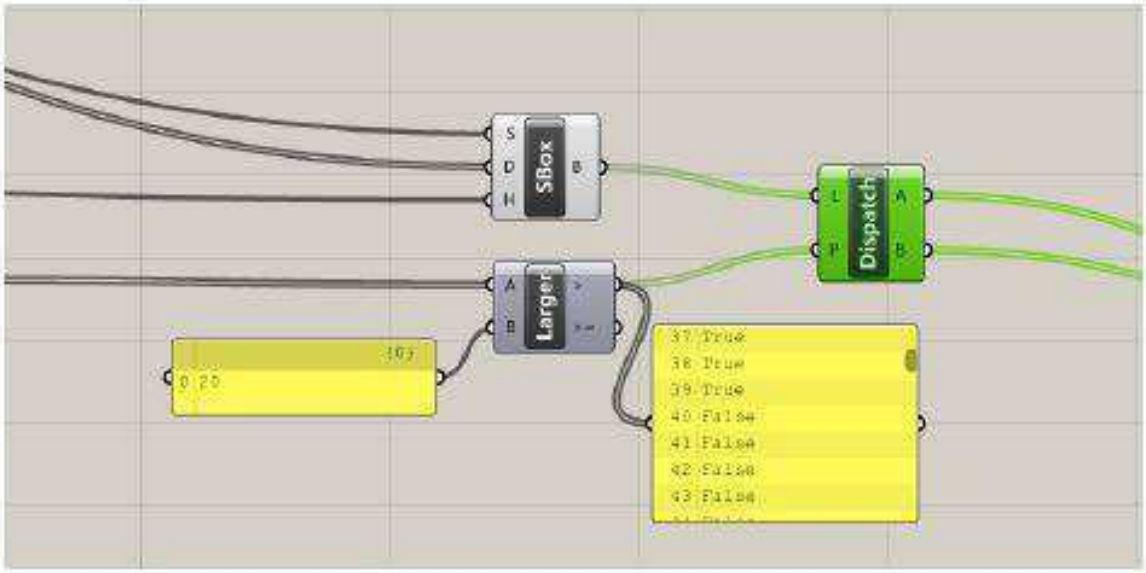
10. Nous calculons ensuite la distance qui sépare ce point de référence des centres des différentes surfaces de subdivisions créées auparavant (Sortie du composant *<Area>*). Cette distance est calculée moyennant le composant *<Vector-Point-Distance>* (entrée *<A>* connectée à la sortie *<C>* du composant *<Area>* et entrée *<B>* connectée au composant *<Point de ref>*)



11. Nous créons ensuite la condition qui permet de répartir les deux types de modules sur la surface de base. Cette condition est créée par le composant *<Maths-Operators-Larger Than>* possédant deux entrées : la première connectée à la sortie du composant *<Dist>* et la deuxième à un composant *<Panel>* intégrant la valeur de référence (Valeur égale à 20). Ce Composant renvoie (par sa sortie *< »> »>* une liste de booléennes (« True » et « False ») indiquant si les surfaces de subdivisions vérifient ou pas la condition (distance supérieure à 20).

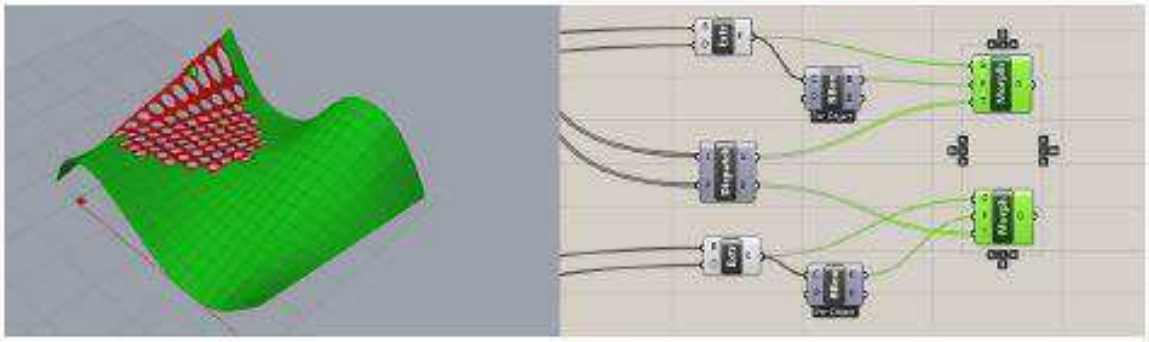


12. Nous essayons de filtrer la liste des volumes de subdivisions (créés par le composant *<SBox>*) en identifiant et en séparant les volumes qui vérifient la condition et celles qui ne la vérifient pas. Ce filtrage (ou aiguillage) est effectué en utilisant le composant *<Sets-List-Dispatch>*. L'entrée *<L>* de ce composant est connectée à la sortie du composant *<SBox>* (intégrant la liste des volumes de subdivisions) alors que son entrée <P> est connectée à la sortie *<« > »>* du composant *<Larger>* (intégrant le résultat de l'évaluation de distance). Ce composant renvoie par sa sortie *<A>* la liste des volumes de subdivision vérifiant la condition et par sa sortie *<B>* le reste des volumes (ne vérifient pas la condition).

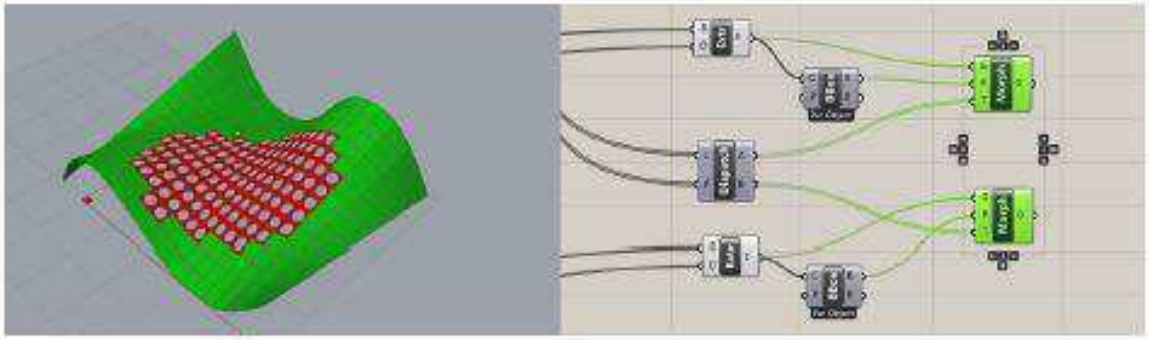


13. L'opération de plaquage des modules sur les différents volumes de subdivisions est effectuée par le composant *<Transform-Morph-Box Morph>*. Ce dernier plaque et adapte des volumes (les modules crées) sur des volumes d'accueil (volumes de subdivisions créés par le composant *<SBox>)*. Le composant <Morph> est utilisé à deux reprises, une première fois pour plaquer les modules avec une

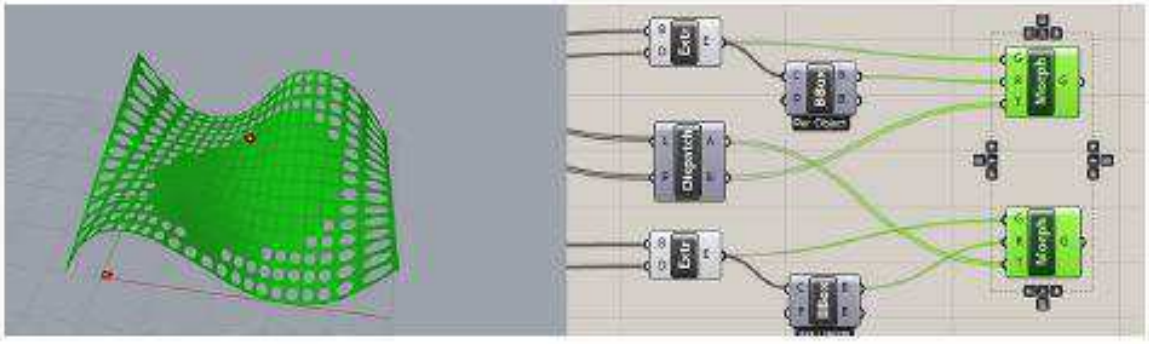
ouverture et une deuxième fois pour les modules opaques (voir les captures d'écrans pour le détail des connexions).



14. Varions la position du point de référence.



15. Inversons les connexions entre le composant *<Dispatch>* et les deux composants *<Morph>*.



4 - Énoncé du modèle paramétrique supplémentaire réalisé au cours de MAN en 2015 – TP4

Université de Liège
ARCHITECTURE
Modélisation Architecturale Numérique

Pierre LECLERCQ pierre.leclercq@ulg.ac.be
Med-Anis GALLAS ma.gallas@ulg.ac.be
Xavier LANGLOIS xavier.langlois@student.ulg.ac.be

A3 TP

COURS DE MODELISATION ARCHITECTURALE
TP - 3e BACH IR ARCHITECTES - 2014/15

MODELISATION PARAMETRIQUE [TP4]

I. Scénario

Modéliser et fabriquer un espace d'exposition intégrant une fonction de projection (espace fermé). Cet espace a une forme de coquille (forme hélico spirale) avec une ouverture à la base reliée par une suture.

La structure est composée d'une série d'arcs liés par des surfaces triangulées formant une série de caissons. Ces derniers créent une structure plissée qui attribue une dimension dynamique à la forme et à l'espace généré.

Le modèle paramétrique générant cette forme est structuré selon une série de courbes crées selon un processus paramétrique. Le découpage de ces courbes nous permettra de construire les différentes faces qui composent notre coquille.

Cet exercice de modélisation porte une attention particulière à la gestion des listes dans *Grasshopper* et les apports qu'ils peuvent apporter au niveau de la modélisation et la préparation du modèle pour la phase de fabrication (réalisation du modèle physique).

Ce modèle est le résultat du travail du RUPP Emilie , SALLOUM Randa et de THIBAULT Pierre, étudiants en Master Architecture Modélisation et Environnement de l'ENSarchitecture Nancy durant la semaine intensive Conception Fabrication Digitale.

II. Phases

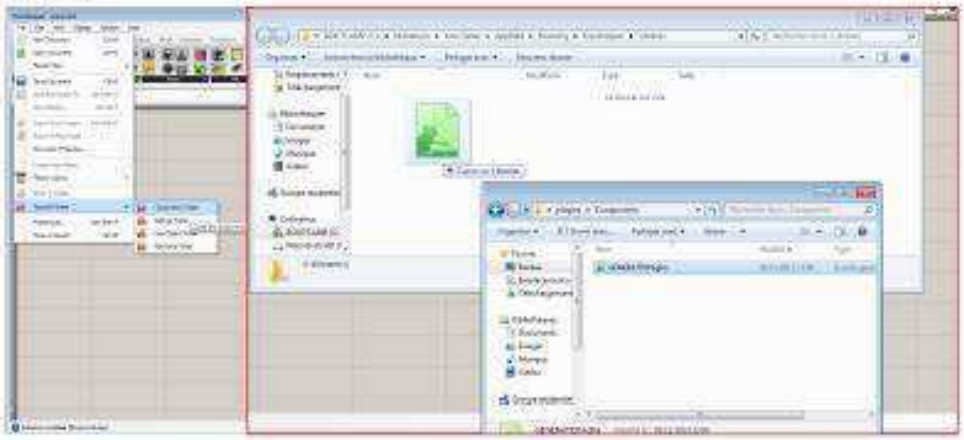
a) Créer les courbes de bases structurant la coquille
b) Découper ces courbes pour créer des groupes de points
c) Utiliser ces points pour générer les surfaces constituant la coquille
d) Préparer les supports graphiques permettant la fabrication du modèle (génération de patrons à découper)

III. Préparation de l'environnement de travail

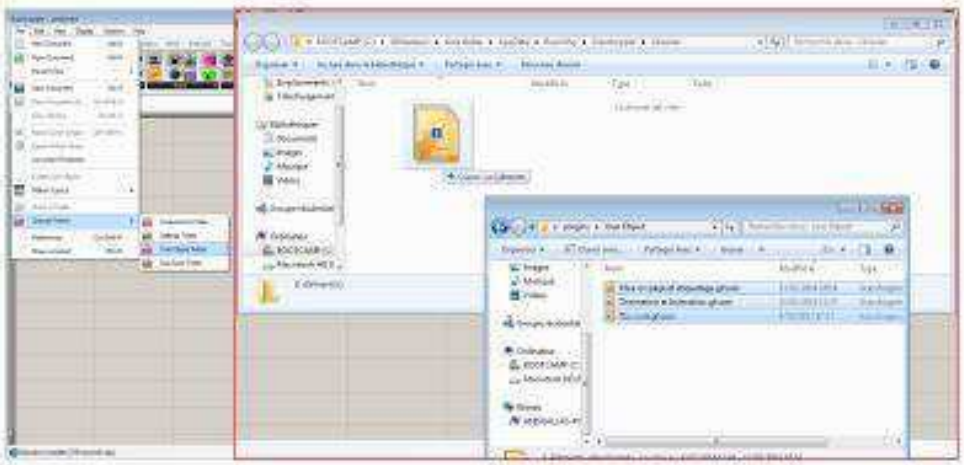
Cet exercice fait appel à des modules extérieurs (Plugins) responsables de l'exécution de certaines tâches complexes. Nous intégrons ces modules en suivant ces étapes :

1. Ouvrir le lien *DropBox*
2. Ouvrir le dossier *plugins*. Ce dernier comporte deux sous-dossiers, *Components* et *User Object*.
3. Ouvrir *Grasshopper*. Aller dans File>SpecialFolders>Components Folder
4. Copier le contenu du dossier *Components* dans le dossier *Components Folder*.
5. Ouvrir *Grasshopper*. Aller dans File>SpecialFolders>User Object Folder
6. Copier le contenu du dossier *User Object* dans le dossier *User Object Folder*.
7. Fermer / relancer *Rhinoceros* et *Grasshopper*
8. Cette procédure permet de charger les modules complémentaires dans Grasshopper.

Components

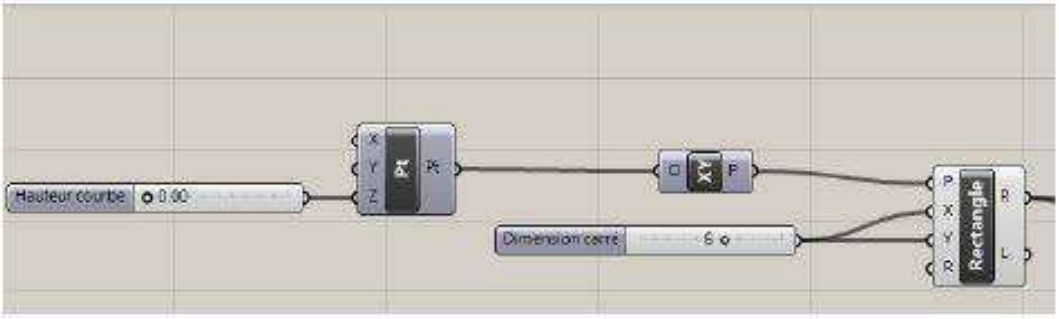


User Object

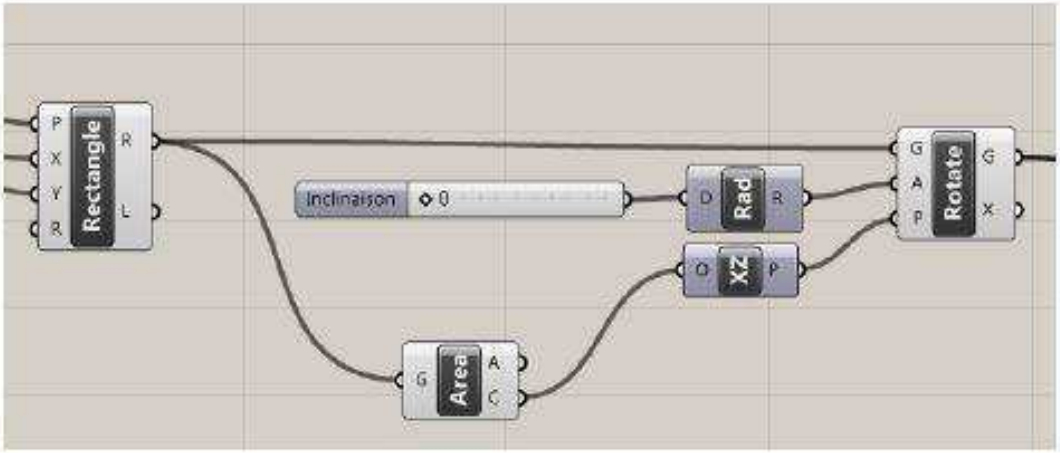


IV. Créer la forme globale du pavillon

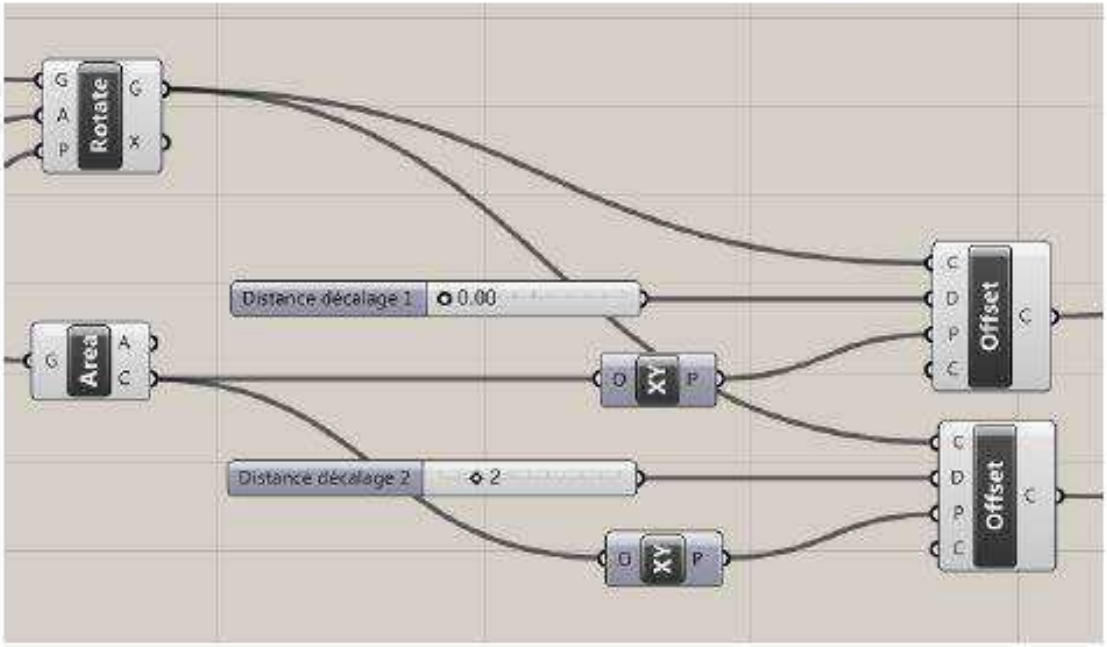
1. Pour commencer, il faut créer un carré (6 mètres de côté). Ce dernier dois être situé sur un plan *XY* avec comme origine un point de cordonnées 0,0,0.



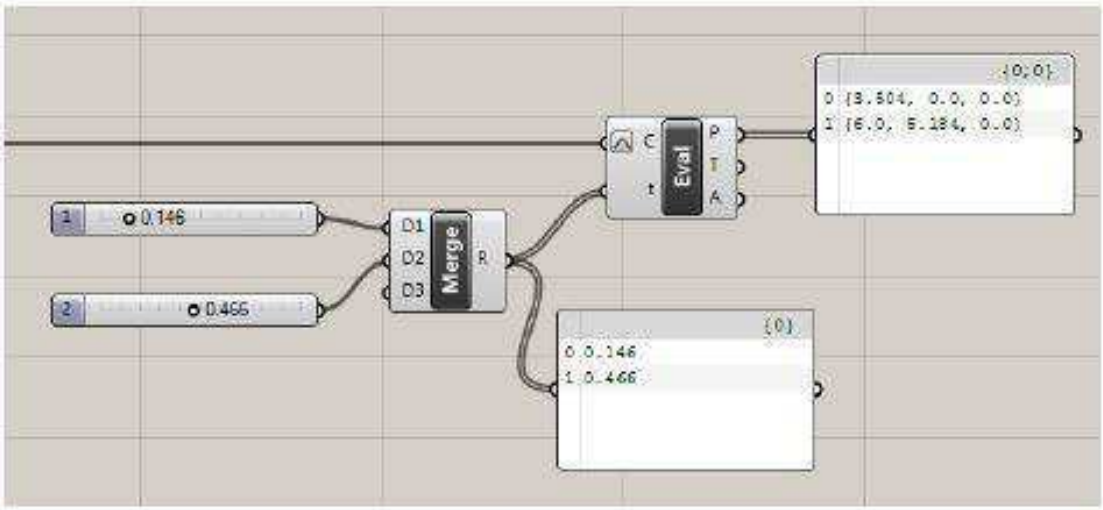
2. Nous allons associer une fonction de rotation à ce rectangle. Le centre de la rotation doit correspondre au centre du rectangle (utiliser *Rotate an object in a plane*). La rotation doit être en degré avec une première valeur égale à 0.



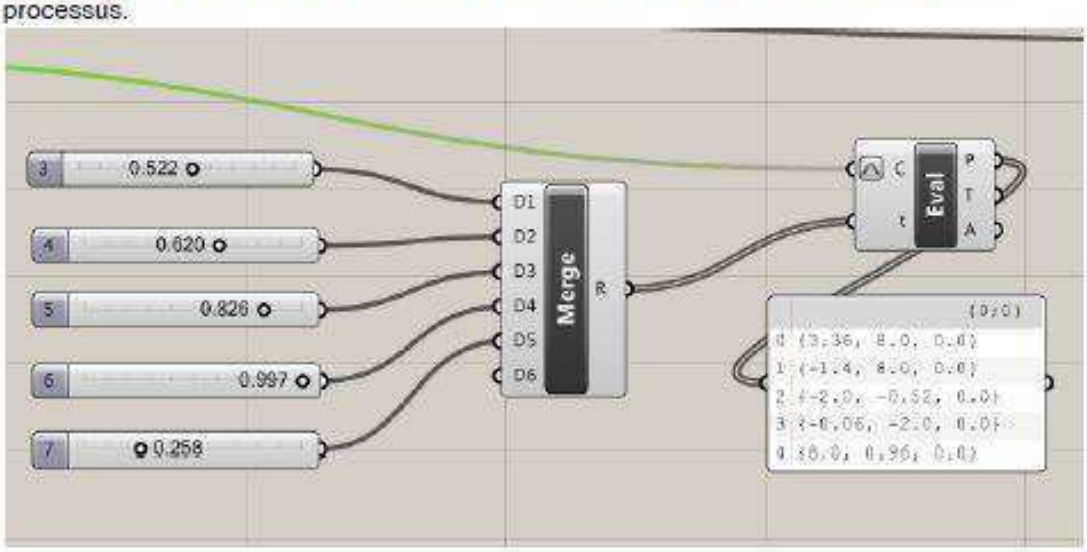
3. Ce rectangle doit être décalé deux fois en utilisant la fonction *Offset*. Ce décalage sera selon le plan *XY* dont l'origine correspond au centre du rectangle. Les valeurs de ces décalages sont : 0 pour le premier et 2 pour le deuxième (Ref capture écran).



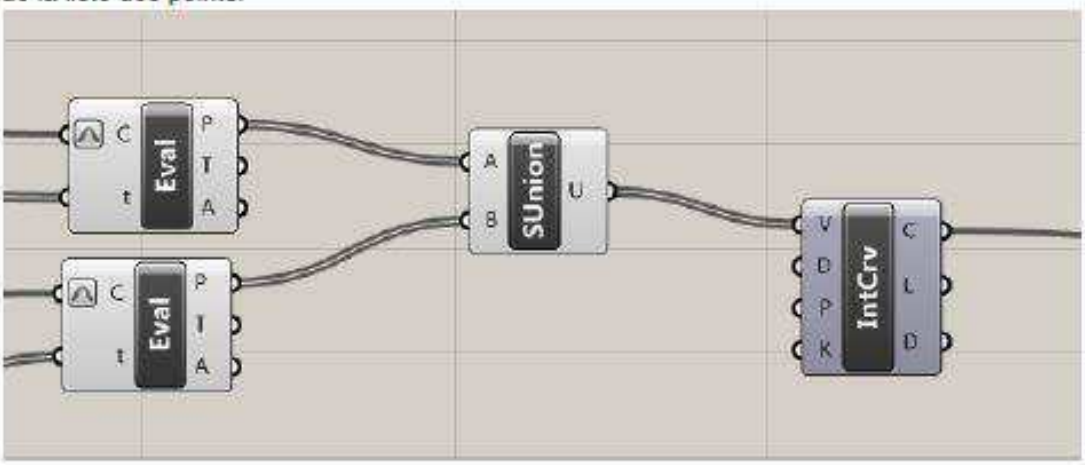
4. Créons ensuite deux points sur le premier carré décalé en utilisant la fonction *EvaluateCurve*. Les postions des deux points sont déterminées en utilisant le composant *Slider* et *Merge*(permet d'associer et de structurer plusieurs valeurs d'entrées). Il faut ajouter l'option « *Reparametrize* » à l'entrée « *C* » ou « *Curve* » du composant « *EvaluateCurve* ».
Vous pouvez choisir vous même les valeurs à attribuer aux sliders ou bien prendre celles indiquées sur la capture écran.



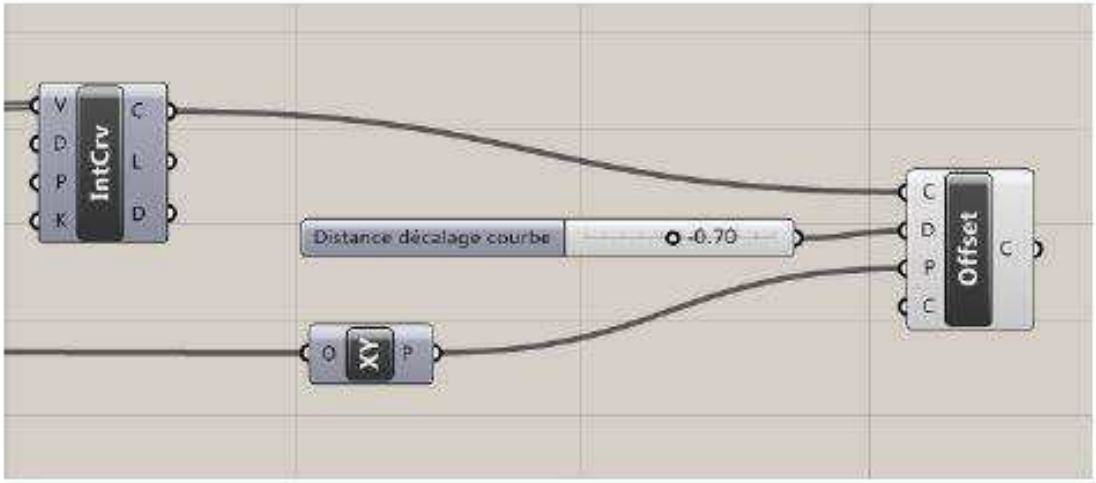
5. Même chose pour le deuxième carré décalé, créons 5 points en utilisant le même processus.



6. L'étape suivante consiste à relier les deux listes de points créés en utilisant le composant « *Set union* ». Utilisons ces points pour créer la première courbe paramétrique. Le composant « *Interpolate* » permet de générer cette courbe à partir de la liste des points.



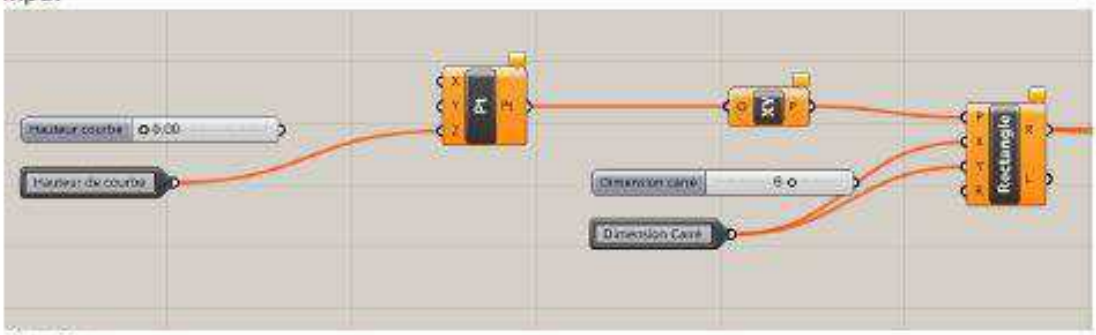
7. Décalons cette première courbe en utilisant le composant « *Offest a curve* ». Utilisons le plan *XY* avec comme origine le centre du premier carré crée.



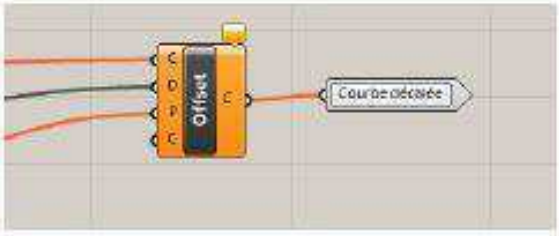
8. Afin d'améliorer la visualisation des points générant les courbes, utilisons le composant « Tag dot » (il est intégré à une nouvelle catégorie «Fac Archi»). Pour l'entrée « Serie », il faut connecter un composant « Panel » et saisir « A ». Ce dernier crée des annotations qui permettent d'identifier les points.

9. Cette opération va être répétée 4 fois. Pour simplifier cette tâche, regroupons les composants crées en un seul « Cluster ». La démarche de création de ce cluster est simple, il faut juste remplacer les entrées et les sorties de certains composants par des « Cluster input » et des « Cluster output « (Ref capture écran).

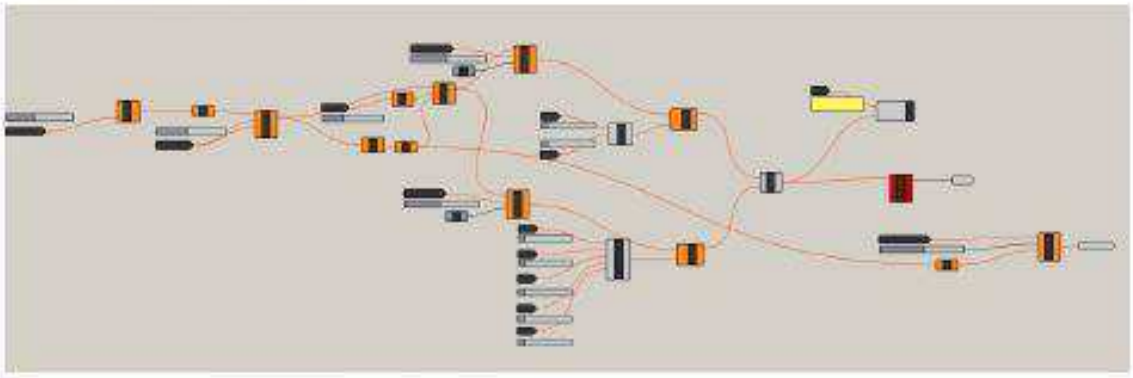
Input



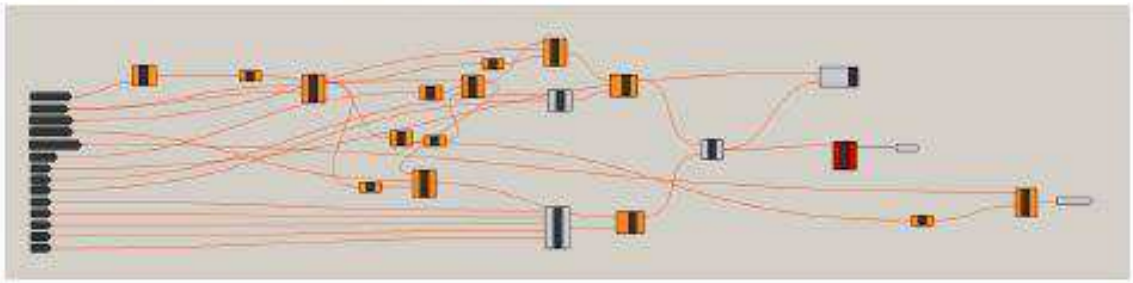
Output



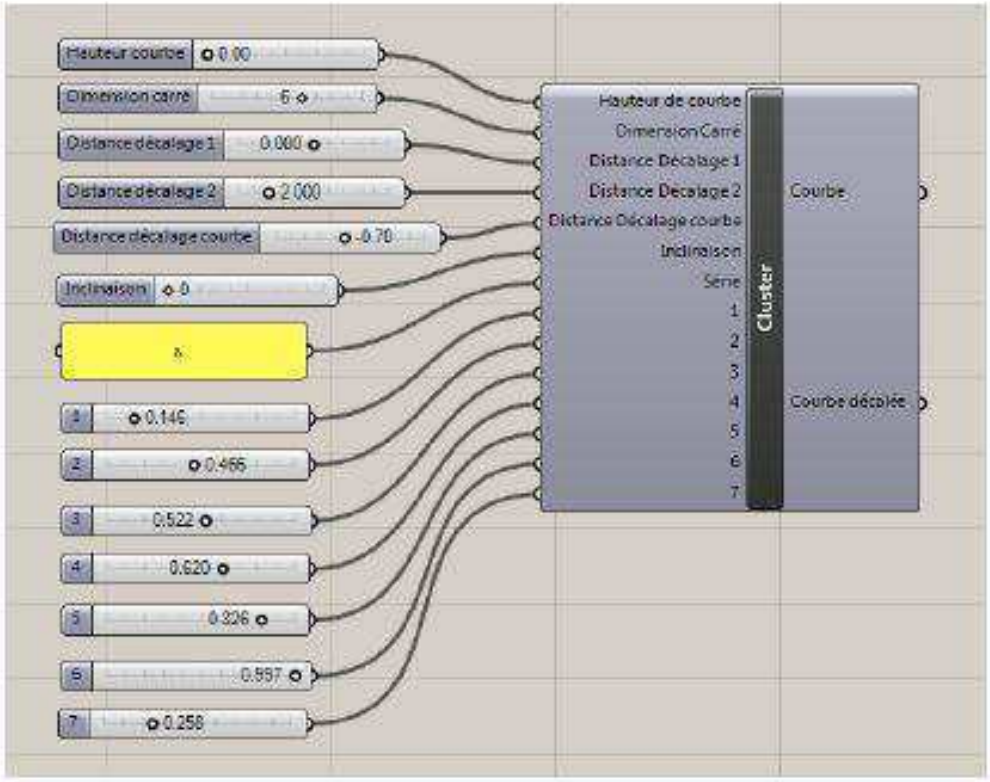
Vue globale du futur cluster



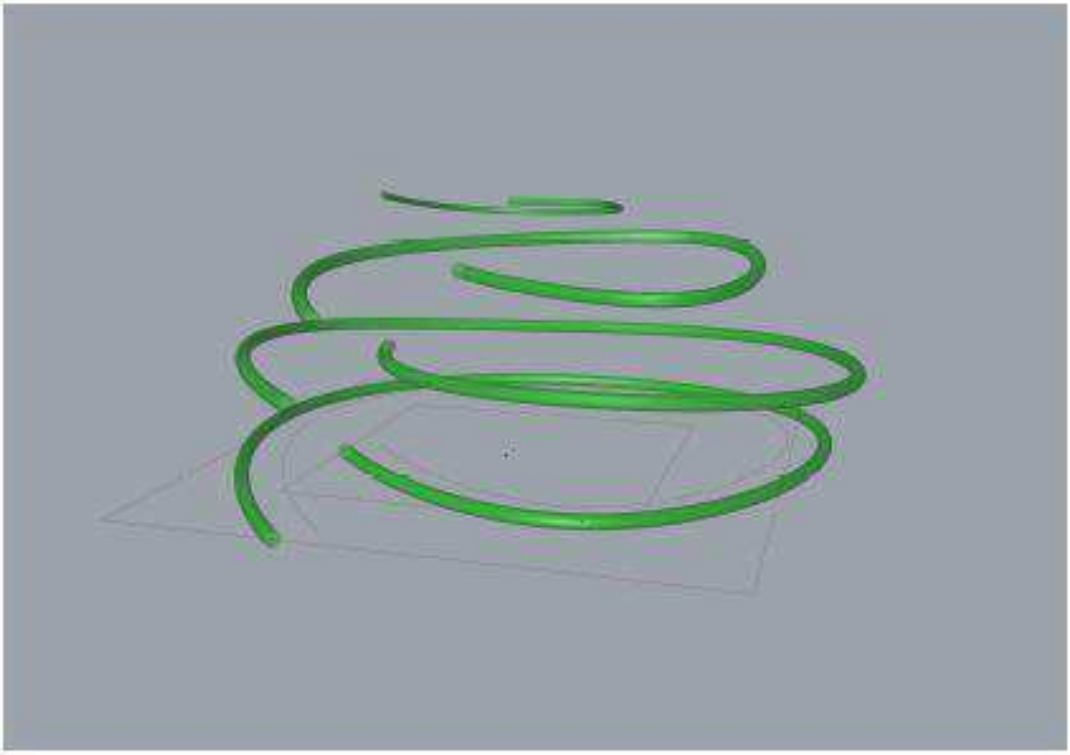
Vue globale avec les inputs alignés à gauche



Cluster



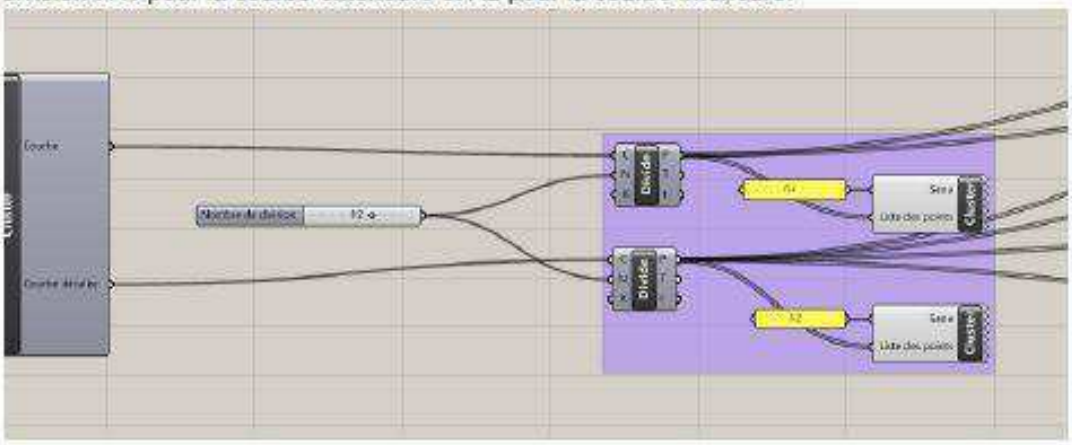
10. Dupliquons ce composant 3 fois et varions les valeurs d'entrées pour créer les courbes structurant notre projet. l'objectif est de déterminer les valeurs permettant de générer courbes superposées (avec une différence d'orientation) afin d'obtenir la forme de la coquille (Ref capture écran).
11.



V. Structurer les courbes génératrices

Diviser les courbes - générer des points - créer des surfaces triangulées constituant l'enveloppe du projet.

1. Diviser les deux courbes (même nombre de division) en utilisant le composant « Divide curve ». Utilisez le composant « Tag dot » pour identifier les points de division : A1 pour la courbe extérieure et A2 pour la courbe intérieure.

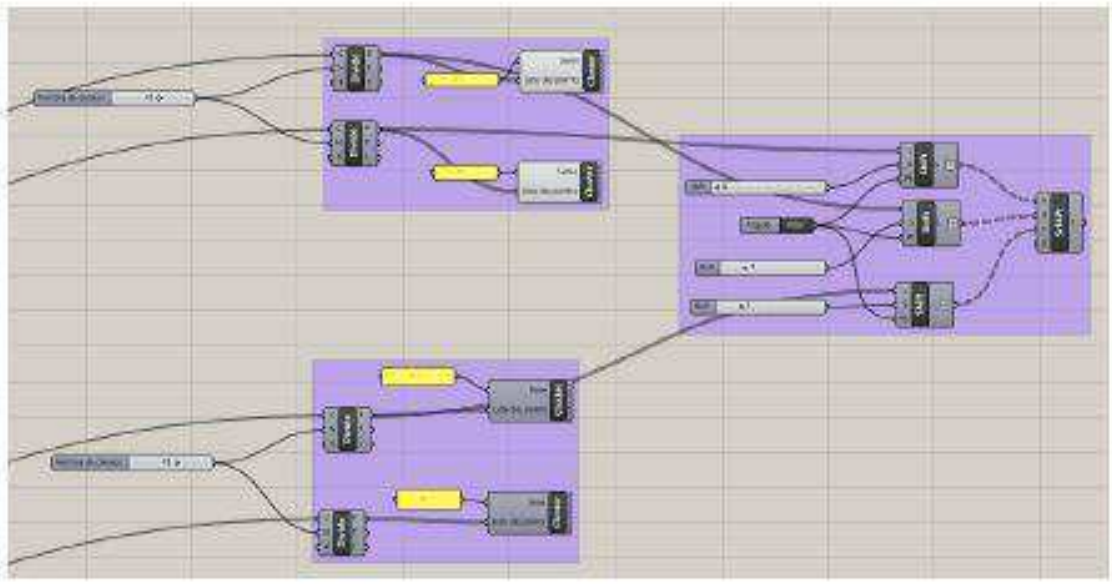


2. Refaire la même opération pour les autres séries de courbes (B1, B2, C1, C2, D1, D2).

3. Pour créer les surfaces, utilisons le composant « 4 point surface ». Ce dernier aura comme entrée trois listes de points avec un décalage. Ce décalage permet de sélectionner les points pertinents pour la création des surfaces.
   Ce décalage est créé en utilisant le composant « Shift ». L'indice de décalage est indiqué sur cette liste :

Niveau	1

	Courbe	Indice
Série de surface 1	A2	0
	A1	1
	B1	1
Série de surface 2	A2	1
	A1	1
	B1	1
Série de surface 3	A2	1
	B1	1
	B2	1
Série de surface 4	A2	1
	B1	2
	B2	1



2 eme niveau de surfaces (courbes B1, B2, C1 et C2) :

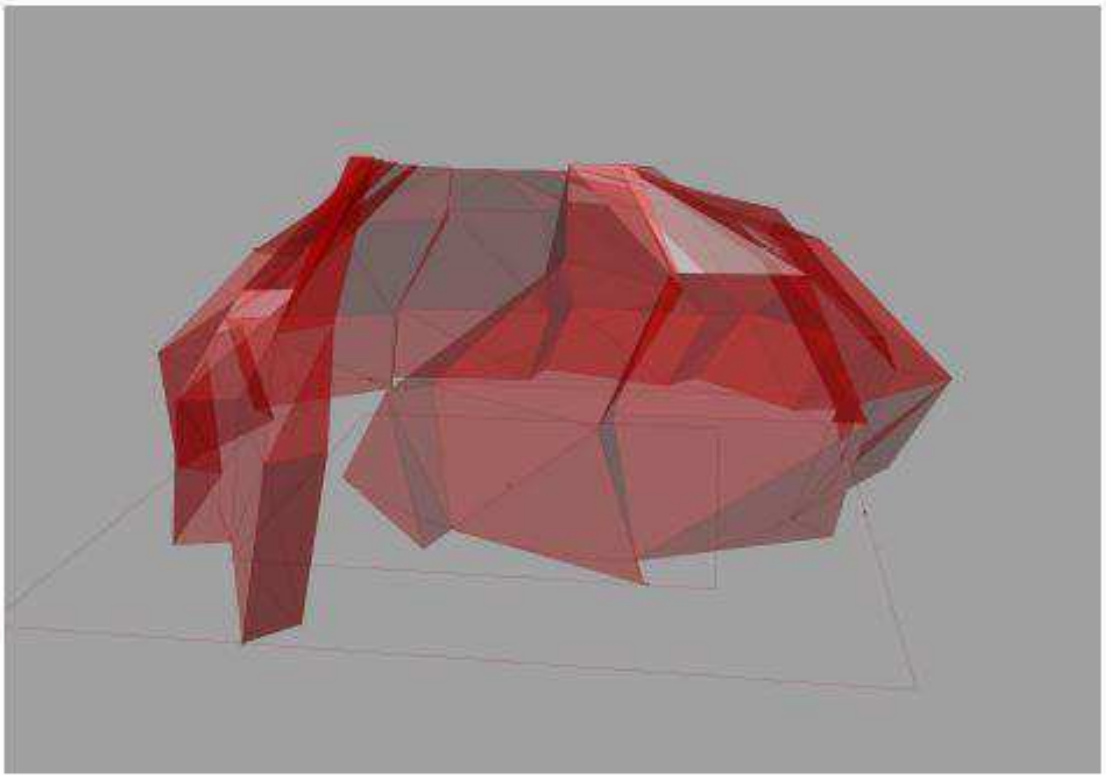
Niveau	2

	Courbe	Indice
Série de surface 1	C2	1
	C1	2
	B2	1
Série de surface 2	C2	1
	C1	1
	B1	1
Série de surface 3	B2	1
	B1	1
	C2	1
Série de surface 4	C1	2
	B1	2
	B2	1

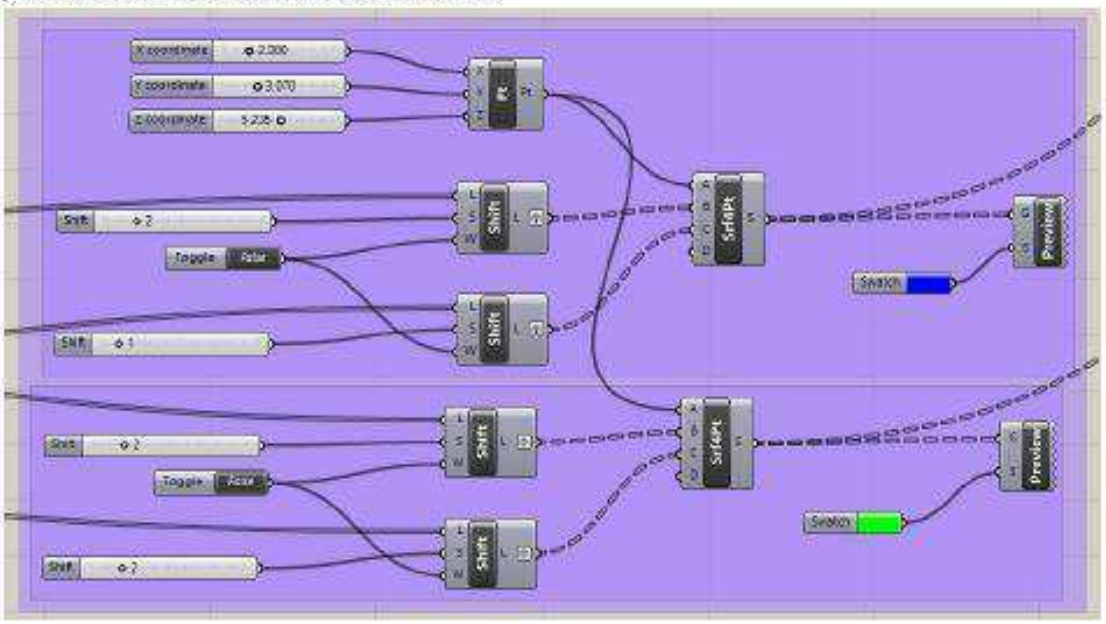
3 eme niveau de surfaces (courbes C1, C2, D1 et D2) :

Niveau	3

	Courbe	Indice
Série de surface 1	C2	1
	C1	2
	D2	1
Série de surface 2	C1	2
	D1	2
	D2	2
Série de surface 3	C2	1
	C1	1
	D2	1
Série de surface 4	C1	2
	D1	2
	D2	1



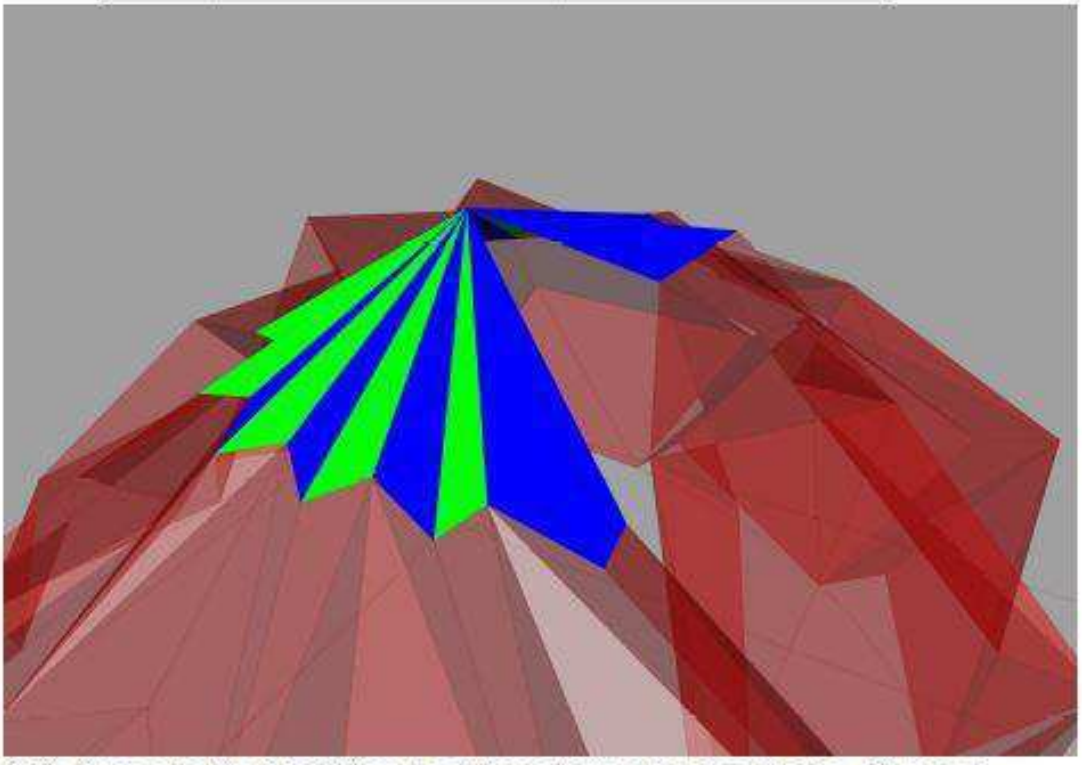
4. Créons, en suivant ce même processus, la toiture de cet espace. Définissons un point de coordonnées 0,0,Z (à définir). Ce point sera connecté à deux listes de points pour constituer les surfaces de couverture.



Niveau	4 (couverture)

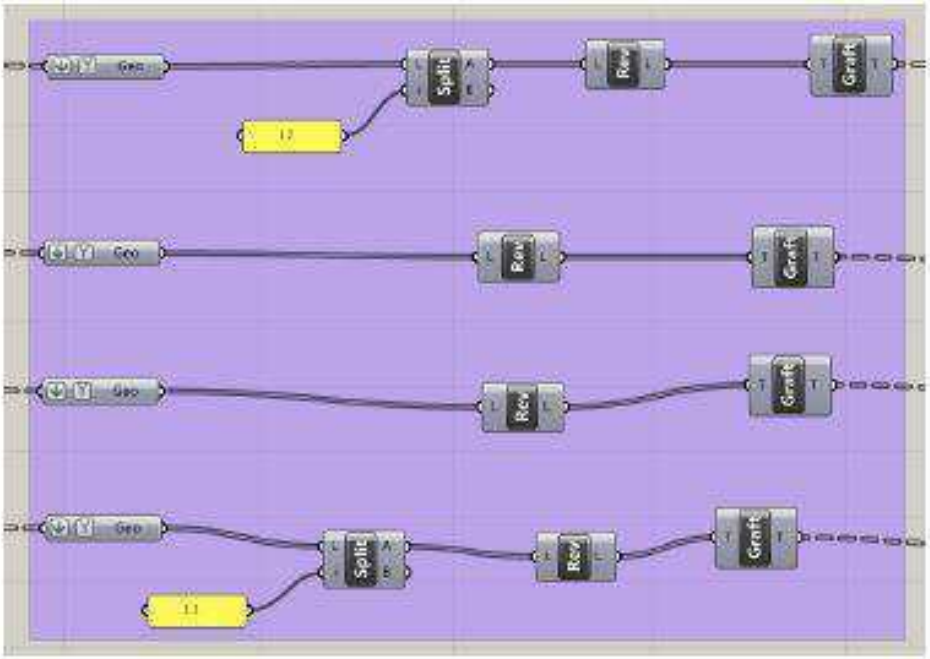
	Courbe	Indice
Série de surface 1	D1	2
	D2	1

Série de surface 2	D1	2
	D2	2

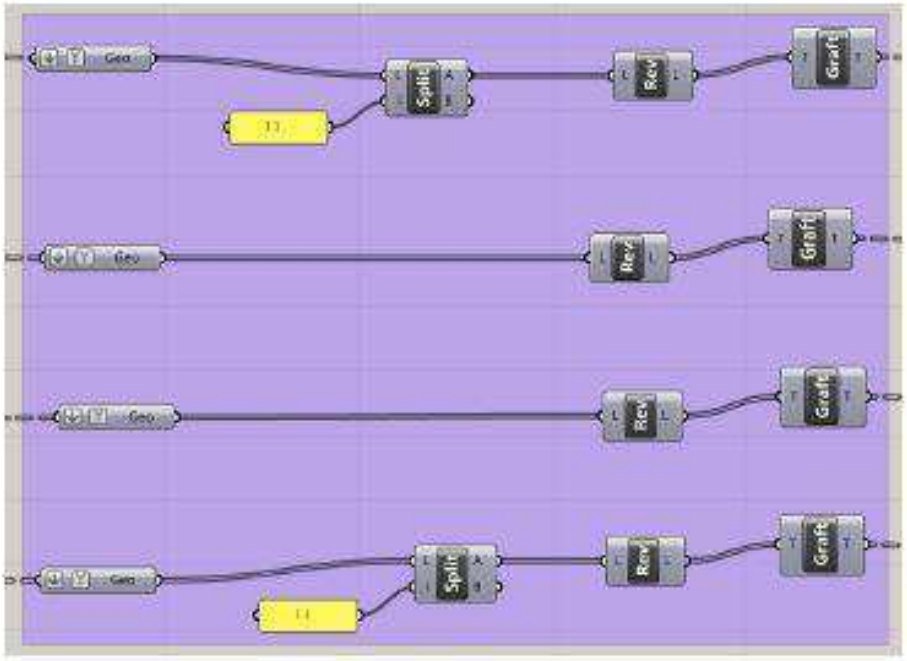


5. Cette étape consiste à regrouper les surfaces pour créer les caissons. Utilisons le composant « Geometry » (avec les options « FItten » et « Simplify ») pour structurer la liste des surfaces. Ce dernier doit être associé au « Split list » pour enlever certaines surfaces. Le résultat sera connecté au composant « Reverse List » et « Graft Tree » pour créer une liste de surface sous forme d'arbre. Le détail des références est précisé sur la capture écran.

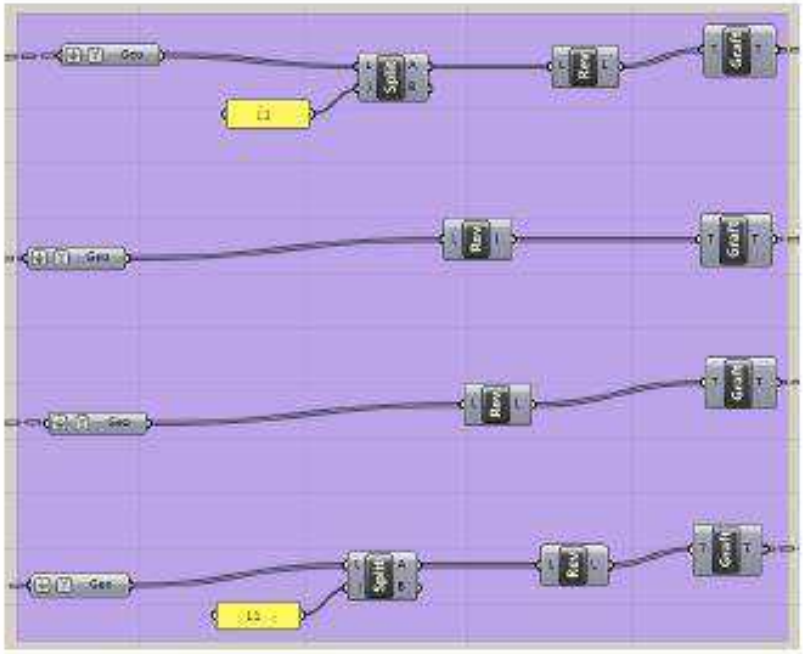
Série 1


Geo
Split
Rev
Graft
17
Geo
Rev
Graft
Geo
Rev
Graft
Geo
Split
Rev
Graft
11

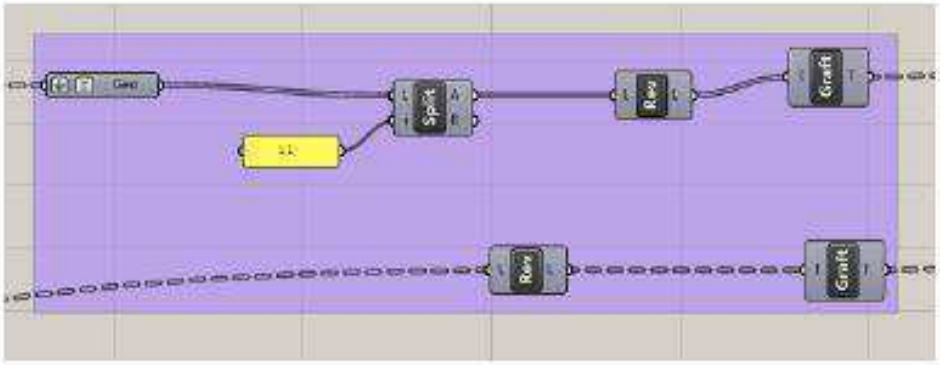
Série 2


Geo
Split
Rev
Graft
11
Geo
Rev
Graft
Geo
Rev
Graft
Geo
Split
Rev
Graft
11

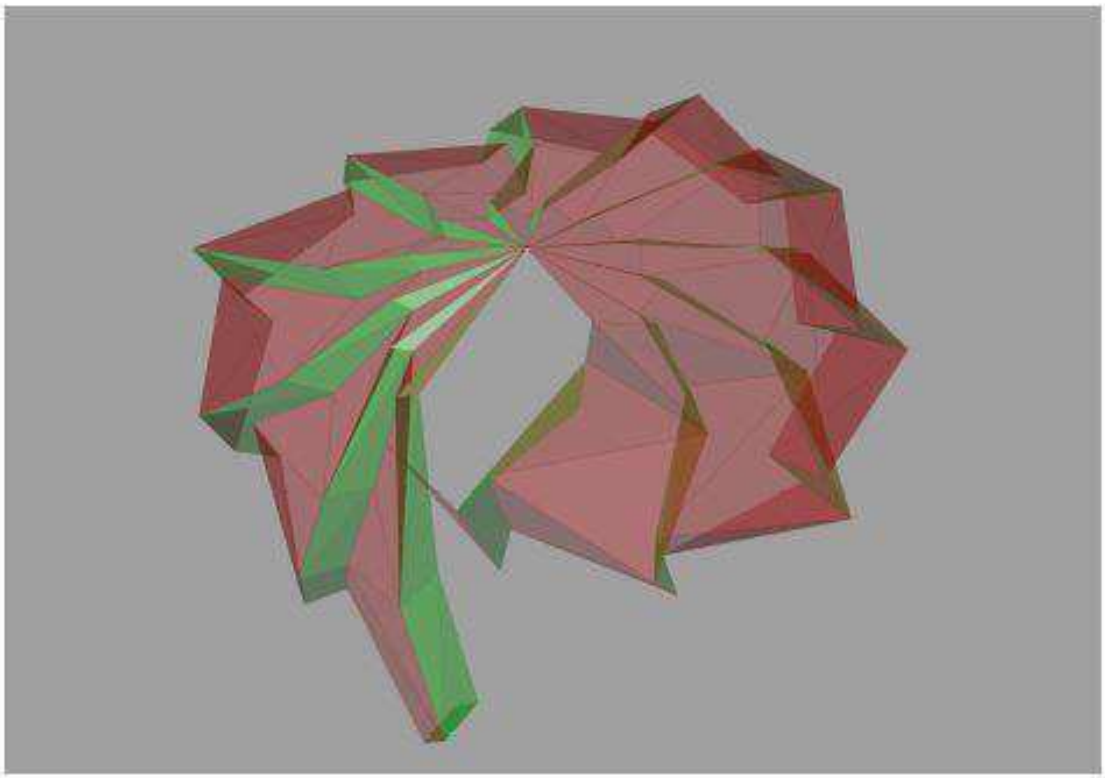
Série 3



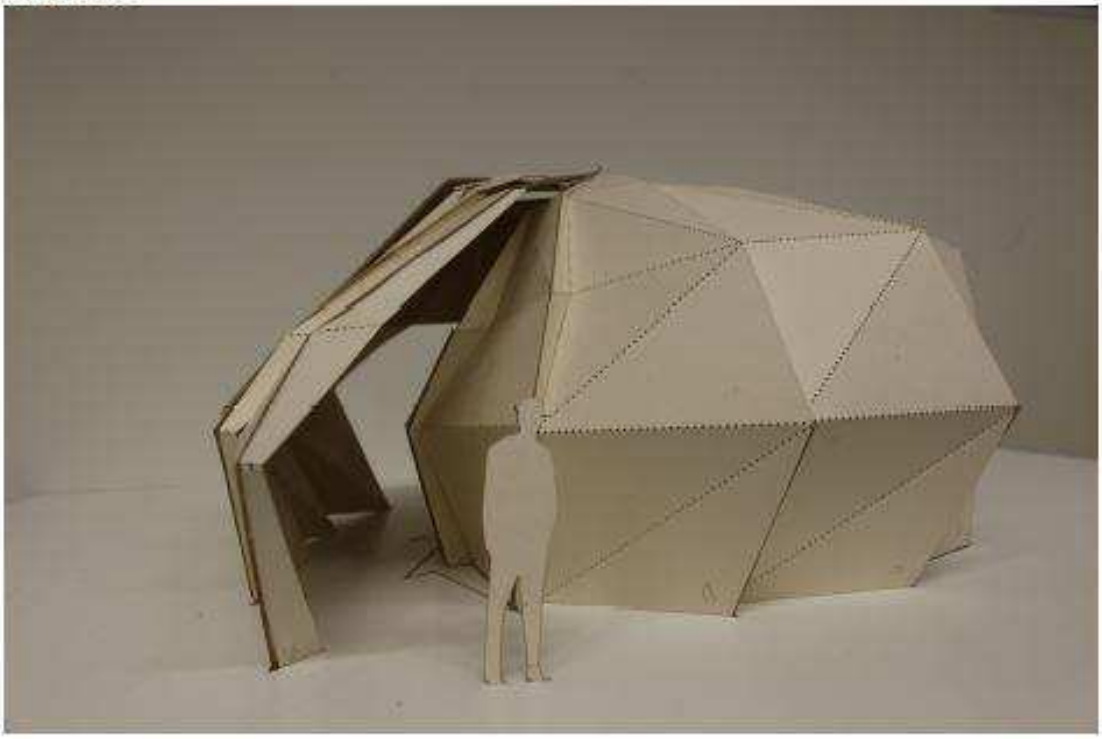
Série 4



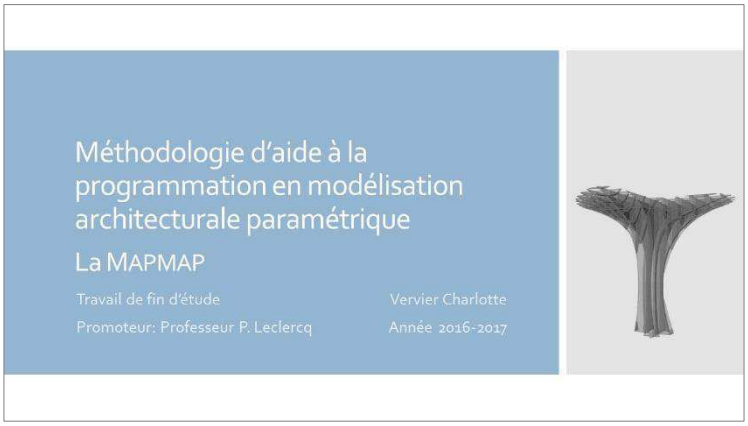
6. La dernière étape consiste à regrouper les listes de surfaces composant les caissons et les listes de surfaces composant les arcs. Utilisons le composant « Merge » pour regrouper les surfaces.

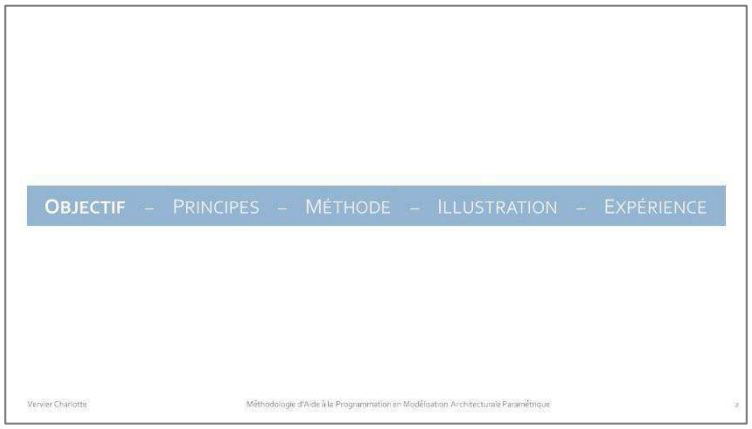


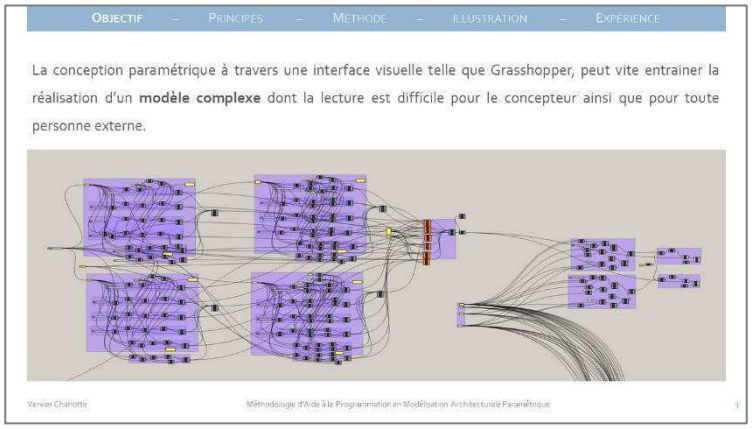
VI. Maquette

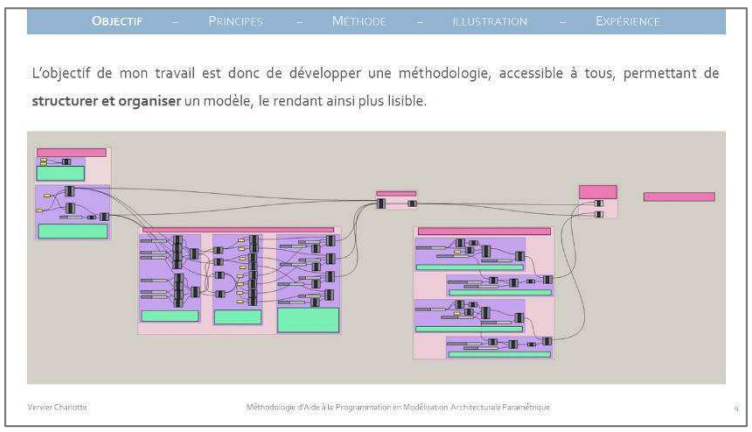


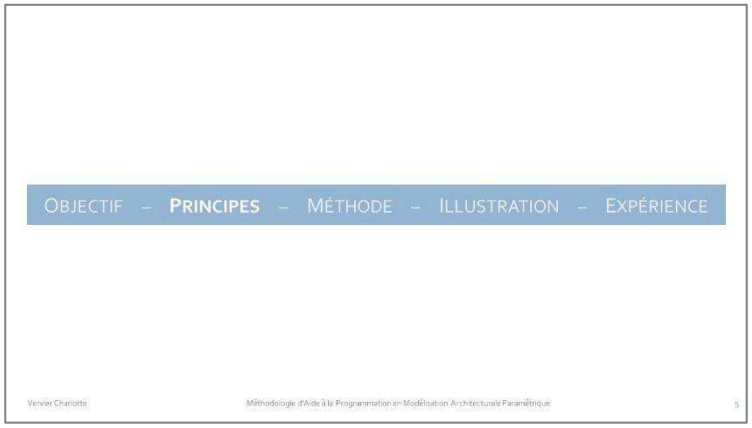
5 - Cours d'introduction à la MAPMAP

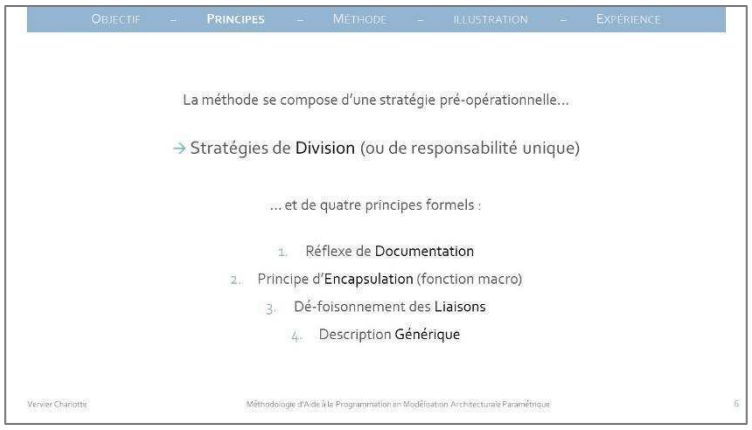


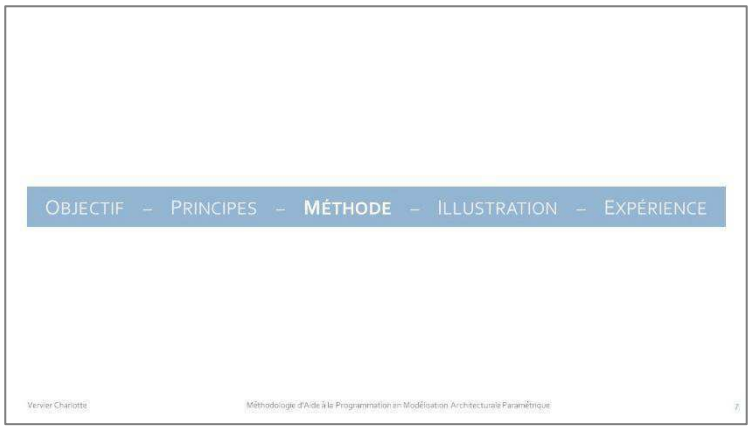


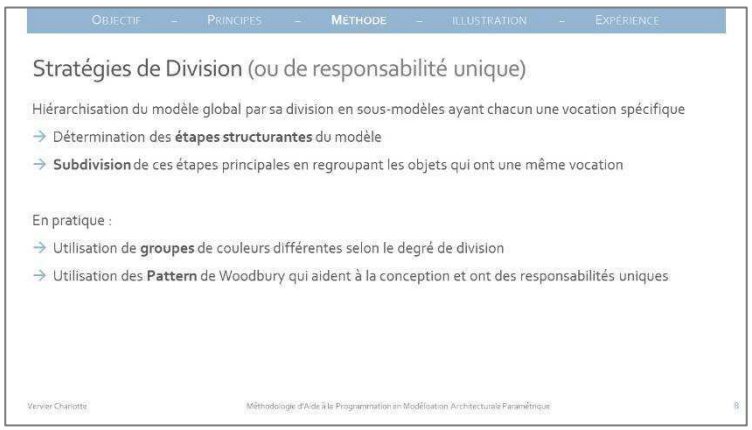





OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE
Vervier Charlotte
Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique
5


OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE
La méthode se compose d'une stratégie pré-opérationnelle...
→ Stratégies de Division (ou de responsabilité unique)
... et de quatre principes formels :
1. Réflexe de Documentation
2. Principe d'Encapsulation (fonction macro)
3. Dé-foisonnement des Liaisons
4. Description Générique
Vervier Charlotte
Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique
6


OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE
Vervier Charlotte
Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique
7


OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE
Stratégies de Division (ou de responsabilité unique)
Hiérarchisation du modèle global par sa division en sous-modèles ayant chacun une vocation spécifique
→ Détermination des étapes structurantes du modèle
→ Subdivision de ces étapes principales en regroupant les objets qui ont une même vocation
En pratique :
→ Utilisation de groupes de couleurs différentes selon le degré de division
→ Utilisation des Pattern de Woodbury qui aident à la conception et ont des responsabilités uniques
Vervier Charlotte
Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique
8

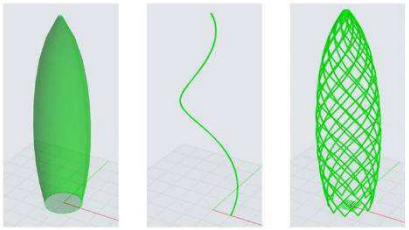
OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division (ou de responsabilité unique)

Illustration par le TP1 : le « Gherkin » de N. Foster

Division en 3 grandes étapes :

1. La formation de la volumétrie
2. La création d'un membre du châssis
3. La réalisation de l'entièreté du châssis



Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 9

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division (ou de responsabilité unique)

Illustration par le TP1 : le « Gherkin » de N. Foster

Formation de la volumétrie | Création d'un membre du châssis | Réalisation de l'entièreté du châssis

Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 10

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Les Pattern

Les pattern sont des échantillons de modèle à vocation unique et des samples, applications concrètes des pattern, sont disponibles sur internet.

P1	Controller	P6	Projection
P2	JIG	P7	Reactor
P3	Increment	P8	Reporter
P4	Point Collection	P9	Selector
P5	Place Holder	P10	Mapping

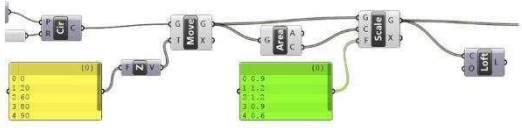
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 11

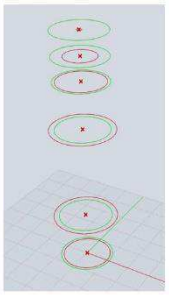
OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 1

Controller

→ Contrôler une partie du modèle à travers une autre partie





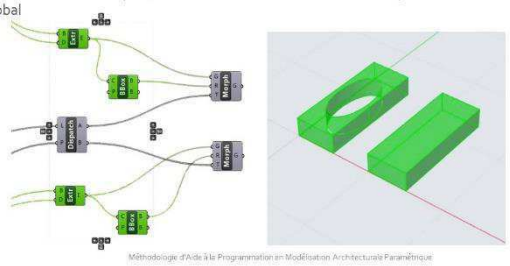
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 12

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 2

JIG

→ Créer une partie du modèle à part, de manière abstraite et non localisée pour l'introduire ensuite dans le modèle global



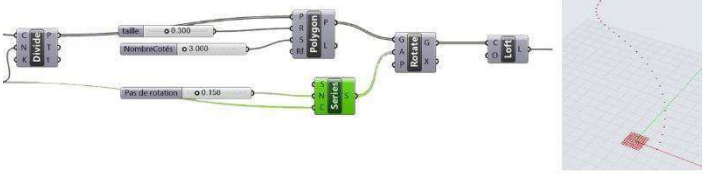
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 13

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 3

Increment

→ Produire un changement à travers une série de valeurs étroitement liées, il en existe de deux sortes : variation croissante d'unités ou variation continue et infinie entre deux limites



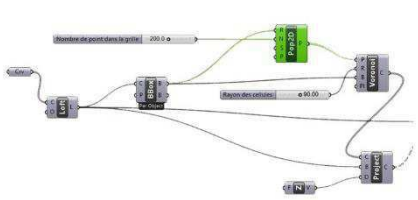
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 14

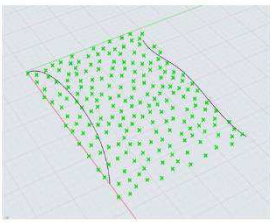
OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 4

Point Collection

→ Regrouper une collection de points localisés





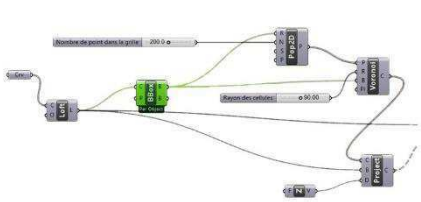
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 15

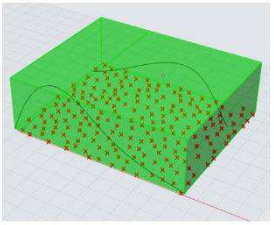
OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 5

Place Holder

→ Déterminer une localisation sur le modèle pour y placer un élément spécifique





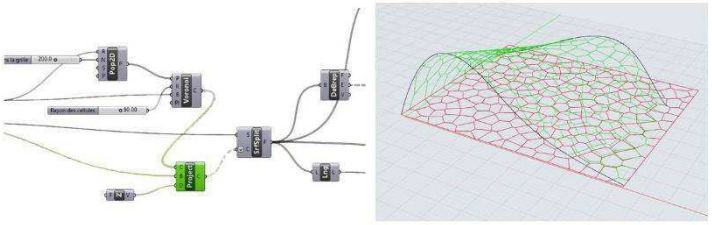
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 16

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 6

Projection

→ Transformer un objet dans un autre contexte géométrique



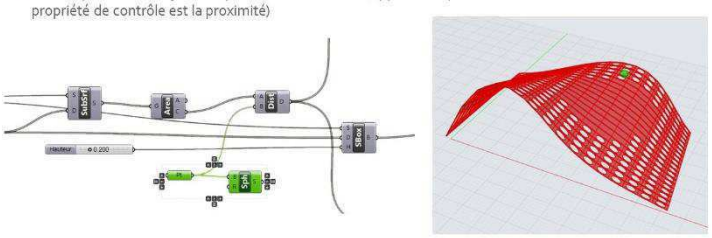
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 17

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 7

Reactor

→ Faire répondre un objet à la proximité d'un autre (application particulière d'un « controller » dont la propriété de contrôle est la proximité)



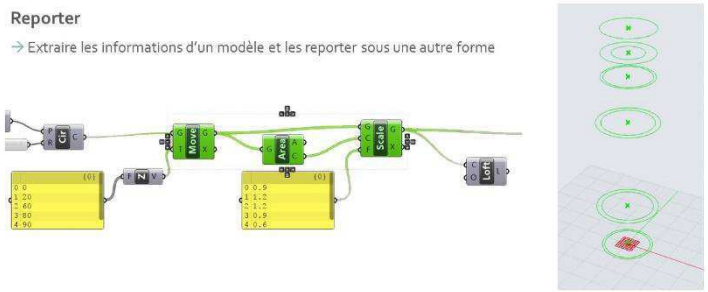
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 18

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 8

Reporter

→ Extraire les informations d'un modèle et les reporter sous une autre forme



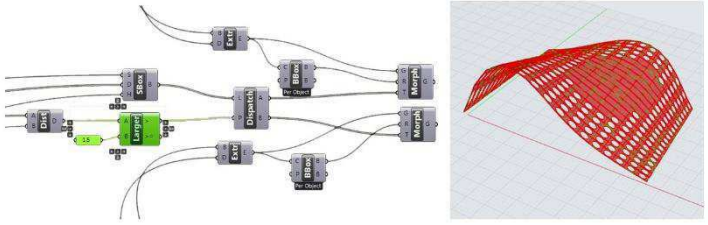
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 19

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 9

Selector

→ Sélectionner les éléments d'une collection qui ont des propriétés spécifiques



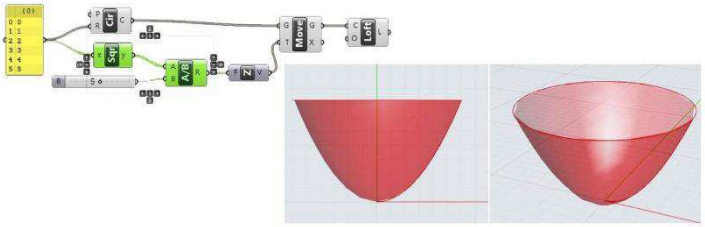
Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — 20

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Stratégies de Division : Pattern 10

Mapping

→ Utiliser une fonction dans un nouveau domaine avec un nouveau rang



OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 1 : Réflexe de Documentation

Donner des noms et/ou descriptions représentatifs aux différentes entités du modèle:

→ **Titrer** les étapes principales

→ **Nommer** les paramètres

→ **Décrire** les opérations de sous-division (référence aux pattern)

En pratique :

→ Utilisation de l'objet *Params-Input-Pannel* pour ajouter des notes (possibilité de modifier la police et la couleur selon la vocation)

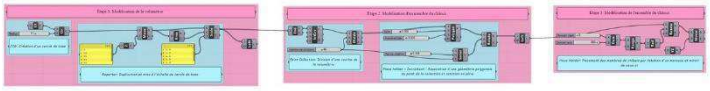
→ Noms non-significatifs à éviter: Point_02 , Courbe_A ,...

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 1 : Réflexe de Documentation

Illustration par le TP1 : le « Gherkin » de N. Foster

→ Utilisation de l'objet *Params-Input-Pannel* pour ajouter des notes (possibilité de modifier la police et la couleur selon la vocation)



→ Noms non-significatifs à éviter: Point_02 , Courbe_A ,...

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 2 : Principe d'Encapsulation (fonction macro)

Simplification de l'interface d'un modèle par la création de « boîtes noires » reprenant une partie du modèle: les *Cluster*

Contexte d'application:

→ **Répétition** d'une partie du modèle comprenant plusieurs fonctions

En pratique :

→ **Nommer** les paramètres *input* et *output* de manière significative, conformément au Réflexe de Documentation

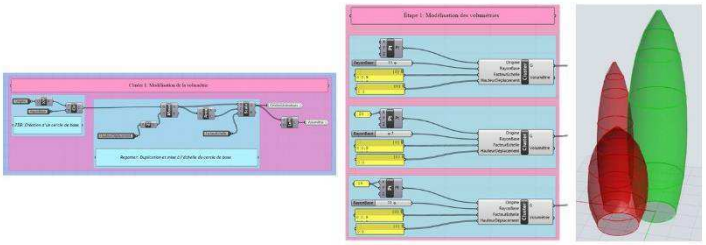
→ Ajouter une note explicative du Cluster à l'extérieur de celui-ci en cas de nécessité

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 2 : Principe d'Encapsulation (fonction macro)

Illustration par le TP1 : le « Gherkin » de N. Foster

→ Création d'un cluster de l'étape 1 pour dupliquer/reporter la volumétrie



OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

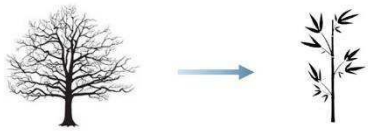
Principe 3 : Dé-foisonnement des Liaisons

Faire dépendre plusieurs fonctions d'un même paramètre réduit les entrées à manipuler pour modifier le modèle MAIS, dans le cas de modèles complexes, cela diminue la lisibilité du modèle.

→ Ne pas utiliser directement un même paramètre d'entrée dans différentes étapes de division du modèle

En pratique:

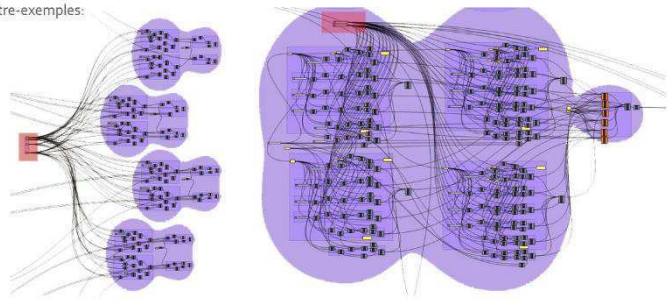
→ **Dupliquer** la variable d'entrée dans les différentes étapes structurantes du modèle



OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 3 : Dé-foisonnement des Liaisons

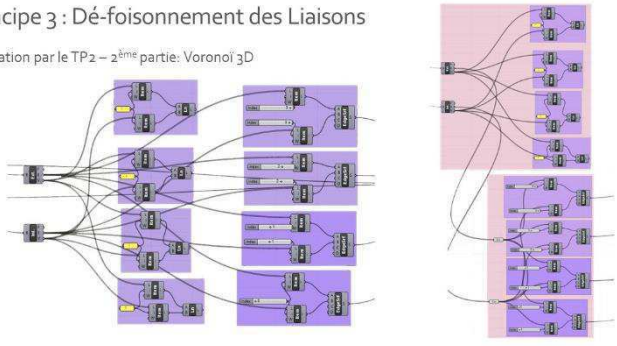
Contre-exemples:



OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 3 : Dé-foisonnement des Liaisons

Illustration par le TP2 – 2ème partie: Voronoï 3D



OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 4 : Description Générique

Fondamentalement, la modélisation paramétrique permet une plus grande **flexibilité** du modèle que la modélisation directe.

Cependant, ce degré de flexibilité dépend également du modèle créé par le concepteur.

En pratique:

→ Prévoir l'**évolution** d'un modèle par la **substitution** de fonctions précises par leurs **fonctions génériques** (qui requièrent souvent un plus grand nombre de paramètres d'entrée)

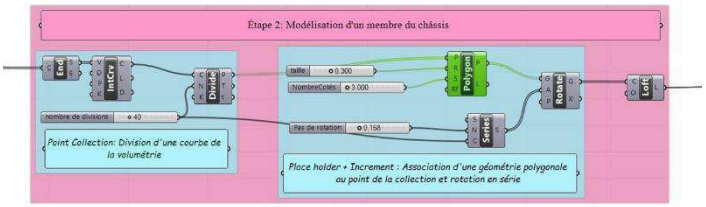
Remarque: Le niveau d'application de ce principe dépend de l'expérience du concepteur.

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Principe 4 : Description Générique

Illustration par le TP1 : le « Gherkin » de N. Foster

→ Utilisation d'un polygone dont le nombre de côtés peut varier pour la création du châssis



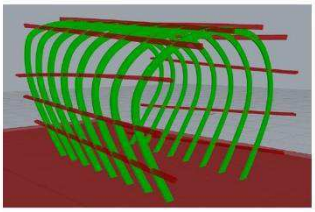
OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – EXPÉRIENCE

Avez-vous des questions sur la MAPMAP ?

OBJECTIF – PRINCIPES – MÉTHODE – **ILLUSTRATION** – EXPÉRIENCE

OBJECTIF – PRINCIPES – MÉTHODE – **ILLUSTRATION** – EXPÉRIENCE

Pavillon sur un chemin piéton



Vervier Charlotte Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique 33

OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – **EXPÉRIENCE**

Vervier Charlotte Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique 34

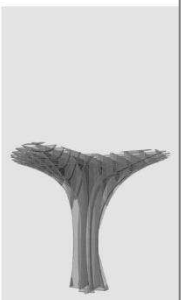
OBJECTIF – PRINCIPES – MÉTHODE – ILLUSTRATION – **EXPÉRIENCE**

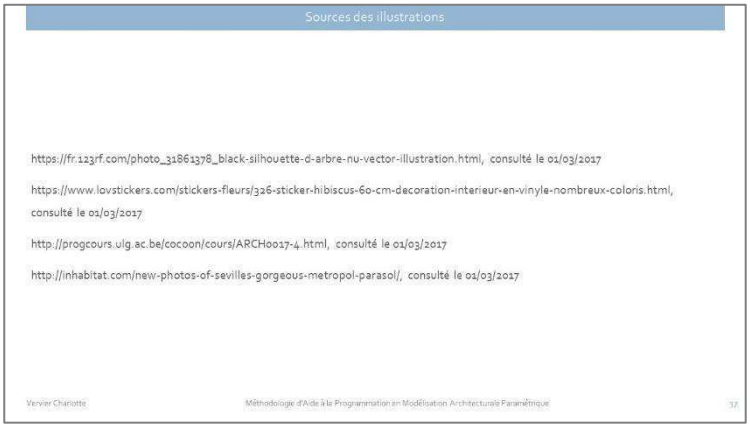
L'expérience consiste à réaliser, par groupe de deux, un modèle paramétrique (inspiré du Metropol Parasol de Séville) en appliquant la méthode MAPMAP.

Sont à votre disposition : l'énoncé, le récapitulatif de la méthode, internet, un encadrant, tout document en votre possession.

Vervier Charlotte Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique 35

Merci pour votre écoute




Sources des illustrations
https://fr.123rf.com/photo_31861378_black-silhouette-d-arbre-nu-vector-illustration.html, consulté le 01/03/2017
https://www.lovstickers.com/stickers-fleurs/326-sticker-hibiscus-60-cm-decoration-interieur-en-vinyle-nombreux-coloris.html, consulté le 01/03/2017
http://progcours.ulg.ac.be/cocoon/cours/ARCH0017-4.html, consulté le 01/03/2017
http://inhabitat.com/new-photos-of-sevilles-gorgeous-metropol-parasol/, consulté le 01/03/2017
Vervier Charlotte
Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique
17

6 - Énoncé de l'exercice de modélisation de la première expérience

Université de Liège
ARCHITECTURE
Modélisation Architecturale Numérique

Pierre LECLERCQ pierre.leclercq @ ulg.ac.be
John SCHRAYEN john.schrayen @ ulg.ac.be
Charlotte VERVIER charlotte.vervier @student.ulg.ac.be

A3 TPpa4

COURS DE MODÉLISATION ARCHITECTURALE
TP 1 - 3e BACH IR ARCHITECTES - 2016/17

MODELISATION PARAMETRIQUE [TP 4]

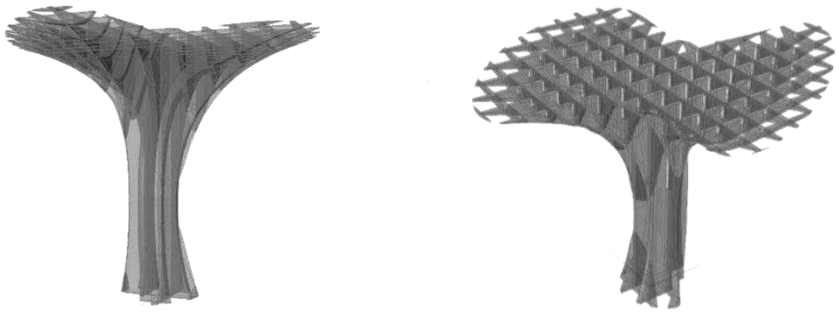
TP préparé par J. Schrayen et C. Vervier

I. Objectifs

Concevoir, modéliser et fabriquer un parasol sur un espace public en s'inspirant du cas réel du METROPOL PARASOL de Séville réalisé par l'architecte Jürgen Hermann Mayer entre 2005 et 2011.

Photo du Metropol Parasol (http://inhabitat.com) (http://www.publicspace.org)

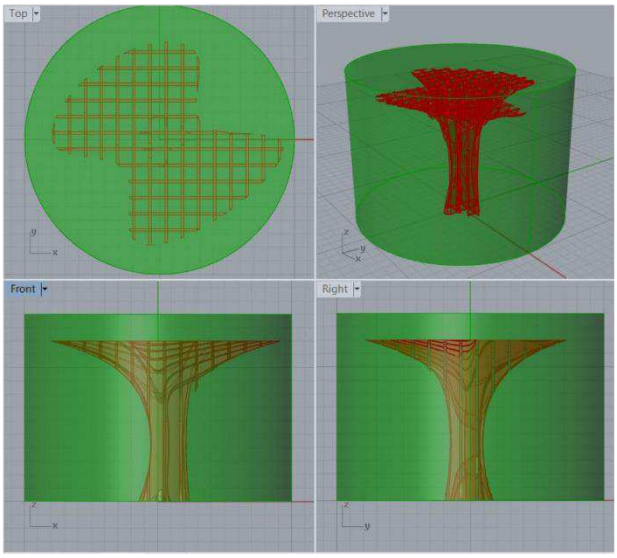
Par la suite, cet exercice fera l'objet d'un prototypage. Cette étape de construction permet au concepteur de développer la réflexion d'un système constructif adapté à sa matérialisation.



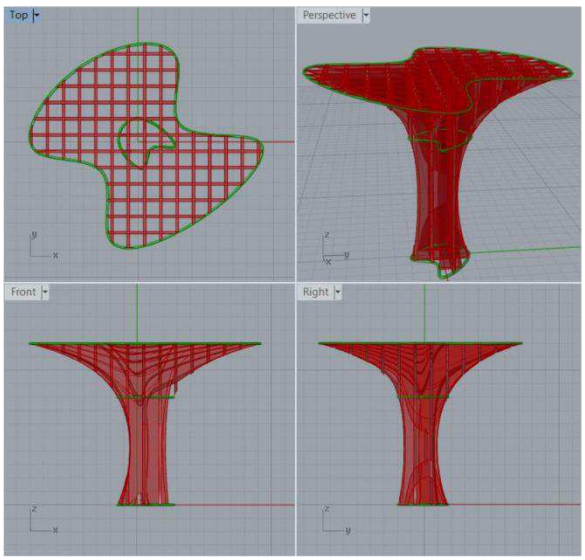
TPpa 4 – 22 mars 2017 1/4

II. Scénario

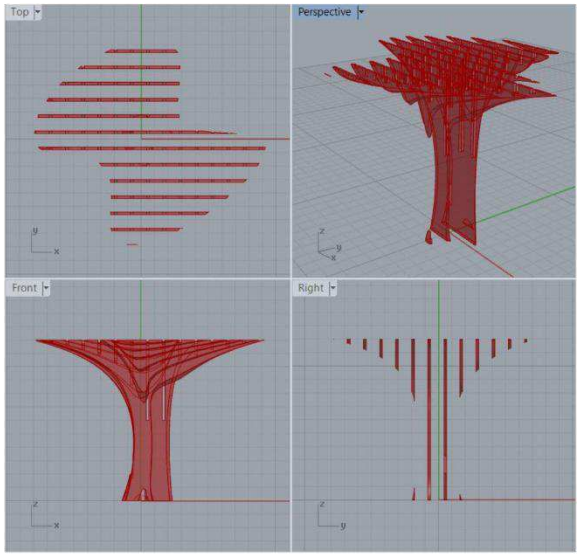
→ La géométrie est composée d'un seul parasol. Le volume de celui-ci doit être compris dans un cylindre de 15unités de haut et 10unités de rayon.

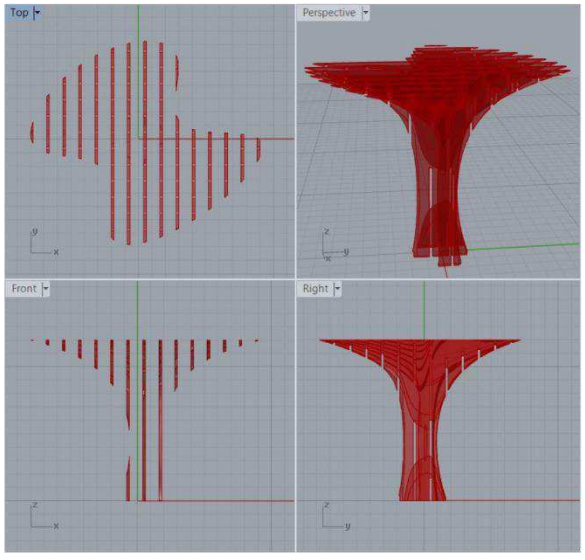


→ Trois courbes situées dans des plans XY composent la volumétrie de base :
La courbe inférieure est comprise dans un cercle de 3 unités de rayon.
La courbe intermédiaire est une projection de la courbe inférieure aux 2/3 de la hauteur du parasol.
La courbe supérieure est comprise dans le volume cylindrique en termes de hauteur et diamètre.



- → Deux trames régulières et orthogonales composent les parois du modèle, ces trames sont distinctes.
- → L'épaisseur des parois est de 0,25 et leur entraxe est inférieur à 3unités.
- → Les intersections des trames sont travaillées de sorte à former des emboitements : Les parois de la première trame sont creusées sur la moitié supérieure des intersections tandis que les parois de la seconde trame sont creusées sur la moitié inférieure des intersections.





III. Paramètres à manipuler :

Paramètres

- → params-geometry : permet d'associer une géométrie de Rhinocéros à une fonction de Grasshopper
- → params-input : permet d'introduire des valeurs

Mathématiques

- → maths-operators : permet d'effectuer des opérations mathématiques

Ensembles

- → sets-list : permet de traiter les listes de données
- → sets-tree : permet de gérer les arborescences au sein des données

Vecteurs

- → vector-plane : permet de créer un plan
- → vector-point : permet de créer un point
- → vector-vector : permet de créer un vecteur

Courbes

- → curve-primitive : permet de créer une courbe à partir de différents types d'entrées

Surfaces

- → surface-analysis : permet d'analyser les propriétés d'une surface
- → surface-freeform : permet de créer une surface à partir de différents types d'entrées
- → surface-primitive : permet de créer différents types de volumes à partir de surfaces
- → surface-util : permet d'effectuer des transformations à partir de surfaces

Intersections

- → intersect-shape : permet de créer des opérations d'intersections entre des volumes

Transformations

- → transform-affine : permet d'effectuer des transformations à partir de volumes
- → transform-euclidean : permet d'effectuer des transformations dans l'espace euclidien

7 - Récapitulatif de la mise en œuvre de la MAPMAP

Mise en œuvre de la MAPMAP

Étape pré-opérationnelle

Application de la **Stratégie de Division** :

➔ Réflexion sur les différentes étapes de modélisation.

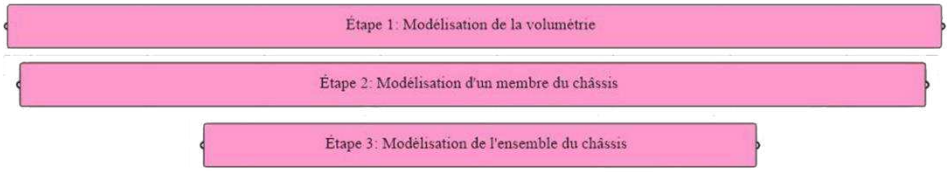
Prémices du **Réflexe de Documentation** :

➔ Réflexion sur les codes couleurs des groupes/sous-groupes et des notes nominatives/descriptives qui s'y rapportent.

Durant la modélisation

Réflexe de Documentation :

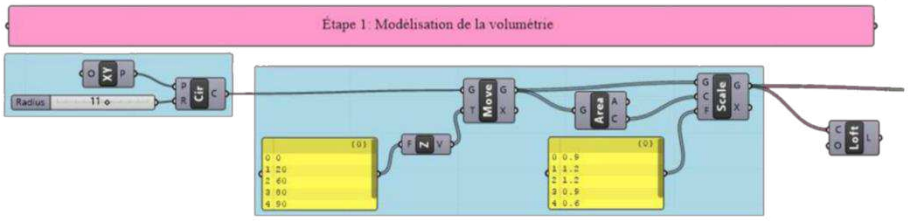
➔ Titrage des étapes principales dans des « Params-Input-Pannel ».



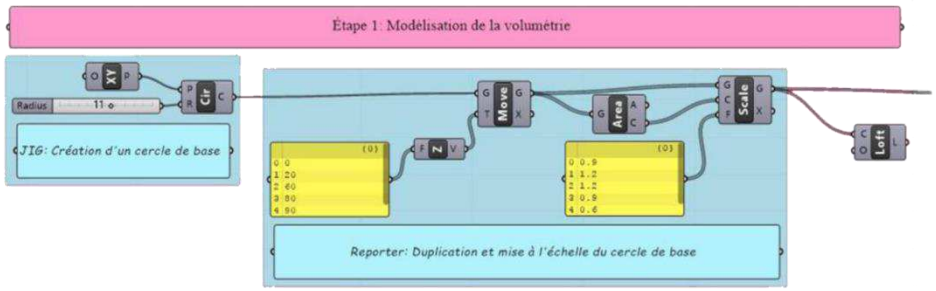
Construction des morceaux de modèle constituant chaque étape successivement à l'aide des Pattern.

Au sein de chaque étape :

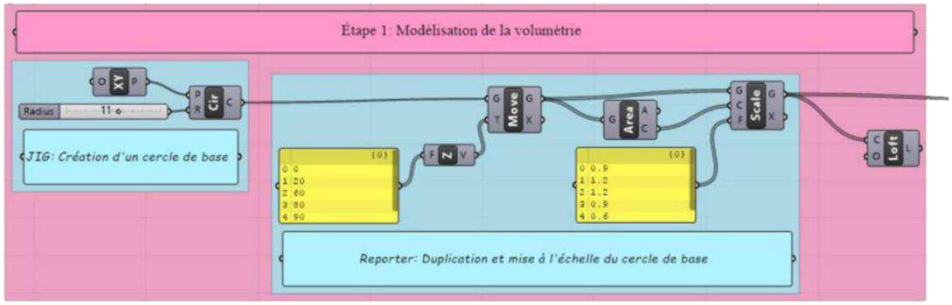
➔ Application de la **Stratégie de Division** en rassemblant, dans un sous-groupe, les objets qui ont une même vocation.
Les Pattern servent à identifier ses différents rôles.



➔ **Principe d'Encapsulation** : création d'un cluster en cas de répétition d'une partie complexe du modèle dont les variables d'entrée changent entre les différentes répétitions.
➔ **Description Générique** : substitution d'objet trop précis par leur générique en prévoyance de la flexibilité du modèle
➔ **Réflexe de Documentation** : ajout d'une note descriptive pour chaque morceau de modèle, dans le (sous-)groupe qui y correspond.



➔ Création d'un groupe avec tous les objets d'une étape, dont son titre.



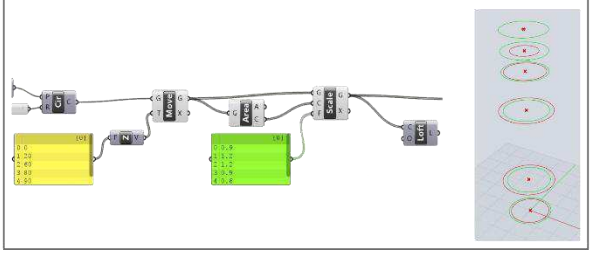
Entre deux étapes :

➔ **Dé-foisonnement des Liaisons** : Si une variable comprise dans une étape est également utilisée dans une autre, celle-ci est dupliquée dans le second groupe et reliée à la variable initiale.

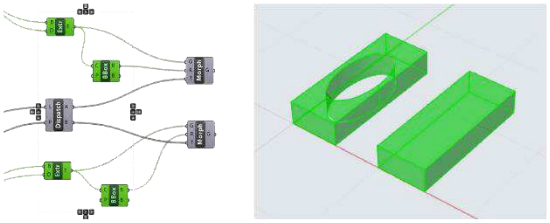
8 - Récapitulatif des dix patterns complémentaires à la MAPMAP

Récapitulatif des Pattern

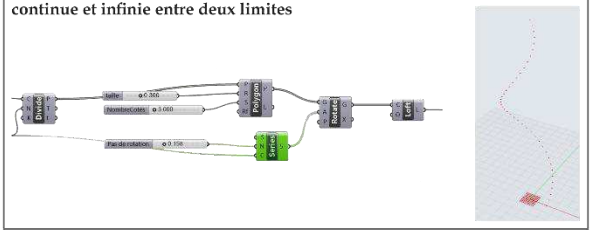
Pattern 1 : Controller
► Contrôler une partie du modèle à travers une autre partie



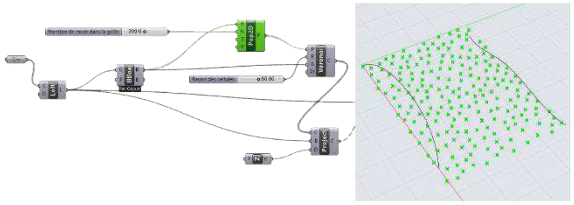
Pattern 2 : JIG
► Créer une partie du modèle à part, de manière abstraite et non localisée pour l'introduire ensuite dans le modèle global



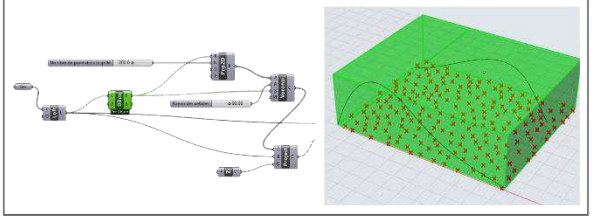
Pattern 3 : Increment
► Produire un changement à travers une série de valeurs étroitement liées, il en existe de deux sortes : variation croissante d'unités ou variation continue et infinie entre deux limites



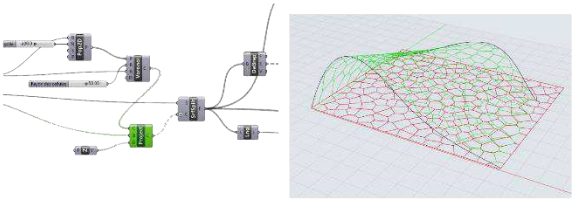
Pattern 4 : Point Collection
► Regrouper une collection de points localisés



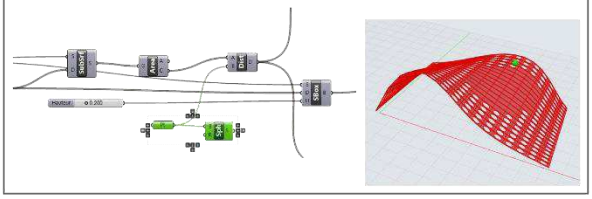
Pattern 5 : Place Holder
► Déterminer une localisation sur le modèle pour y placer un élément spécifique



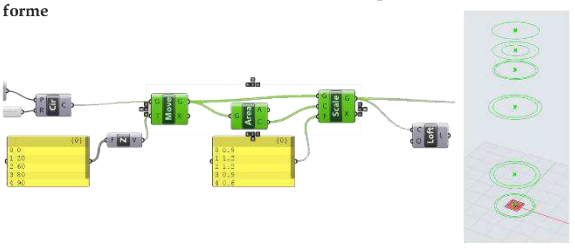
Pattern 6 : Projection
► Transformer un objet dans un autre contexte géométrique



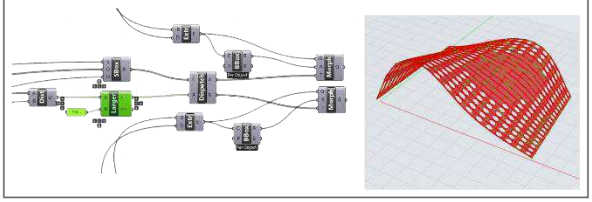
Pattern 7 : Reactor
► Faire répondre un objet à la proximité d'un autre (application particulière d'un « controller » dont la propriété de contrôle est la proximité)



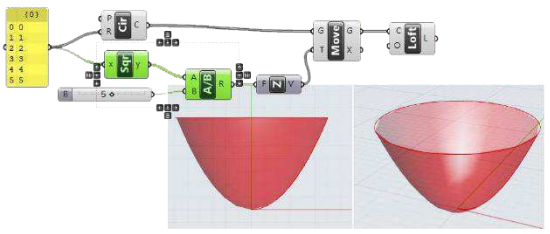
Pattern 8 : Reporter
► Extraire les informations d'un modèle et les reporter sous une autre forme



Pattern 9 : Selector
►Sélectionner les éléments d'une collection qui ont des propriétés spécifiques



Pattern 10 : Mapping
► Utiliser une fonction dans un nouveau domaine avec un nouveau rang



Vervier Charlotte — Méthodologie d'Aide à la Programmation en Modélisation Architecturale Paramétrique — Mars 2017

9 - Questionnaire du groupe 1 lors de la première expérience

Nom : Prénom :

Formation(s) :

Questionnaire d'évaluation complémentaire :

Votre participation à ce questionnaire permettra d'améliorer les résultats de l'expérience à laquelle vous venez de participer.

Pour les trois premières questions, veuillez entourer le chiffre de 1 à 5 correspondant à votre réponse.

Q1 - Suite à la formation dispensée durant les séances de TP de MAN, comment considérez-vous votre niveau de compréhension de la modélisation paramétrique et du logiciel Grasshopper ?

1. Je ne comprends rien et je fais souvent des erreurs
2. Je pose beaucoup de question mais j'arrive à suivre
3. Je construis le modèle au fur et à mesure des explications de l'assistant sans trop de soucis
4. J'avance plus vite que les explications de l'assistant mais je dois parfois poser des questions
5. Je fais tout le TP tout seul et pourrait me passer des encadrants

Q2 - Quelle est votre compréhension de la MAPMAP, la méthodologie qui vous a été présentée en début d'expérience ?

1. Je n'ai rien compris
2. J'ai compris quelques principes mais je ne vois pas comment les mettre en œuvre
3. J'ai compris tous les principes mais j'ai encore du mal à savoir comment les appliquer
4. J'ai compris comment mettre la méthode en œuvre au fur et à mesure de la création du modèle
5. J'ai tout de suite compris la méthode et comment l'appliquer

Q3 - D'après vous, quelle est la contribution de la MAPMAP ?

1. La méthode est tout à fait inutile
2. Cela pourrait être utile mais, personnellement, je n'en ai pas besoin
3. Quelques idées me semblent bonnes à retenir mais pas l'entièreté de la méthode
4. C'est une méthode intéressante mais je ne suis pas sûr de l'appliquer à chaque fois
5. Cette méthode est parfaite et je suis sûr de m'en resservir

Q4 - Quelles difficultés avez-vous rencontré pour réaliser l'exercice ?

.....
.....
.....
.....

Q5 - Quelles difficultés avez-vous rencontré lors de la mise en œuvre la MAPMAP ? Pourquoi ?

.....
.....
.....
.....

Q6 - Quels avantages identifiez-vous dans l'application de la MAPMAP ?

.....
.....
.....
.....

Q7 - Quels inconvénients identifiez-vous dans l'application de la MAPMAP ?

.....
.....
.....
.....

Q8 - Certains principes de la MAPMAP vous semblent-ils moins pertinents ? Ou plus difficiles à comprendre/à mettre en œuvre ? Lesquels et pourquoi ?

.....
.....
.....
.....

10 - Questionnaire du groupe 2 lors de la première expérience

Nom : Prénom :

Formation(s) :

Questionnaire d'évaluation complémentaire :

Votre participation à ce questionnaire permettra d'améliorer les résultats de l'expérience à laquelle vous venez de participer.

Pour la première question, veuillez entourer le chiffre de 1 à 5 correspondant à votre réponse.

Q1 - Suite à la formation dispensée durant les séances de TP de MAN, comment considérez-vous votre niveau de compréhension de la modélisation paramétrique et du logiciel Grasshopper ?

1. Je ne comprends rien et je fais souvent des erreurs
2. Je pose beaucoup de question mais j'arrive à suivre
3. Je construis le modèle au fur et à mesure des explications de l'assistant sans trop de soucis
4. J'avance plus vite que les explications de l'assistant mais je dois parfois poser des questions
5. Je fais tout le TP tout seul et pourrait me passer des encadrants

Q2 - Quelles difficultés avez-vous éventuellement rencontré pour réaliser l'exercice ?

.....
.....
.....
.....
.....
.....

11 - Énoncé de la seconde expérience

Université de Liège
2ème Master Ingénieur Civil Architecte
Travail de Fin d'étude

Charlotte VERVIER charlotte.vervier @student.ulg.ac.be
Promoteur : Pierre LERCLERCQ pierre.leclercq @ ulg.ac.be

SECONDE PHASE D'EXPERIMENTATION : COMPREHENSION D'UN MODELE PARAMETRIQUE PAR UN UTILISATEUR EXTERNE

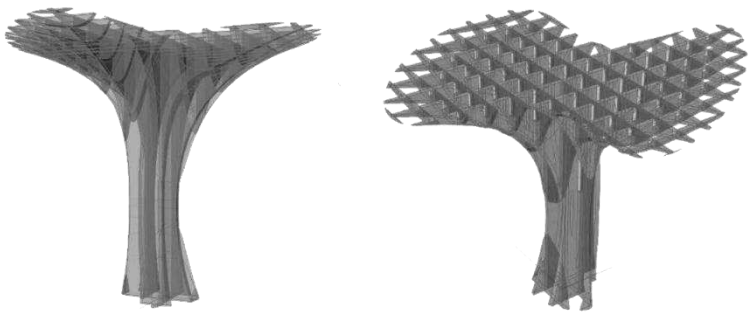
Énoncé préparé par C. Vervier

Présentation du modèle

Les étudiants de 3ème bachelier ingénieur civil architecte inscrits au cours de MAN à l'université de Liège, ont réalisé, par binôme, un exercice de modélisation paramétrique grâce au logiciel Rhinocéros et son extension Grasshopper, le 22 mars 2017.

Cet exercice consistait en la modélisation d'une structure inspirée du METROPOL PARASOL de Séville réalisé par l'architecte Jürgen Hermann Mayer entre 2005 et 2011 :

Type de géométrie paramétrique obtenue :



Scénario

Afin d'évaluer la qualité de communication des modèles réalisés par les étudiants de 3èmes bacheliers. Il vous est demandé de prendre connaissance de l'un des modèles et d'effectuer deux opérations sur celui-ci :

- Dans un premier temps, il vous est demandé de modifier l'épaisseur des montants de la structure de sorte que celle-ci valle 0,195 unités pour tous les montants.
- Dans un second temps, il vous est demandé de modifier l'entraxe entre les montants de la structure de sorte que celui-ci valle 1 unité.

Durant cet exercice, vous avez pour seules sources d'information :

- Ce document contenant une brève explication des modèles
- La maquette physique fournissant des informations sur la géométrie modélisée
- Le modèle qui vous a été assigné

La réalisation de cet exercice est chronométrée, cependant l'exercice prendra fin au bout de 20minutes.

12 - Questionnaire de la seconde expérience

Nom : Prénom :

Formation(s) :

Questionnaire d'évaluation complémentaire :

Votre participation à ce questionnaire permettra d'améliorer les résultats de l'expérience à laquelle vous venez de participer.

Pour la première question, veuillez entourer le chiffre de 1 à 5 correspondant à votre réponse.

Q1 - Suite à la formation dispensée durant les séances de TP de MAN, comment considérez-vous votre niveau de compréhension de la modélisation paramétrique et du logiciel Grasshopper ?

1. Je n'ai jamais rien compris
2. J'arrivais à suivre les TP mais je posais beaucoup de question
3. Je construisais le modèle au fur et à mesure des explications de l'assistant sans soucis mais je n'ai jamais utilisé ce type de logiciel en dehors des TP
4. Depuis les TP j'ai réutilisé quelque fois le logiciel mais je n'ai pas beaucoup approfondi mes connaissances
5. Depuis les TP je me suis intéressé à ce type de logiciel et j'ai développé mes connaissances dans ce domaine

Q2 - Quelles difficultés avez-vous rencontré pour réaliser l'exercice ?

.....
.....
.....
.....

Q3 - Trouvez-vous le modèle bien construit ? Pourquoi ?

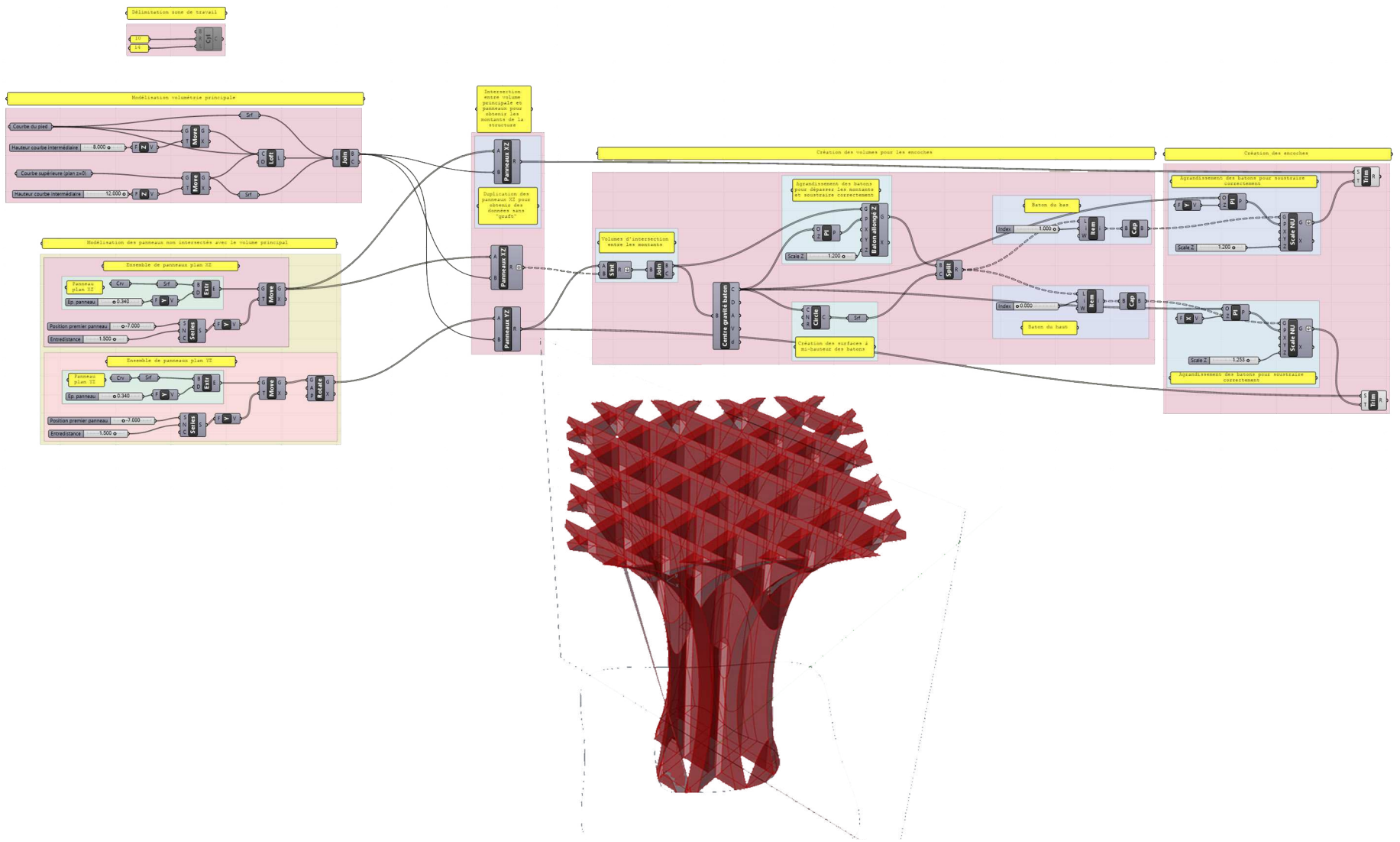
.....
.....
.....
.....

Q4 - Auriez-vous des suggestions pour améliorer la compréhension du modèle ?

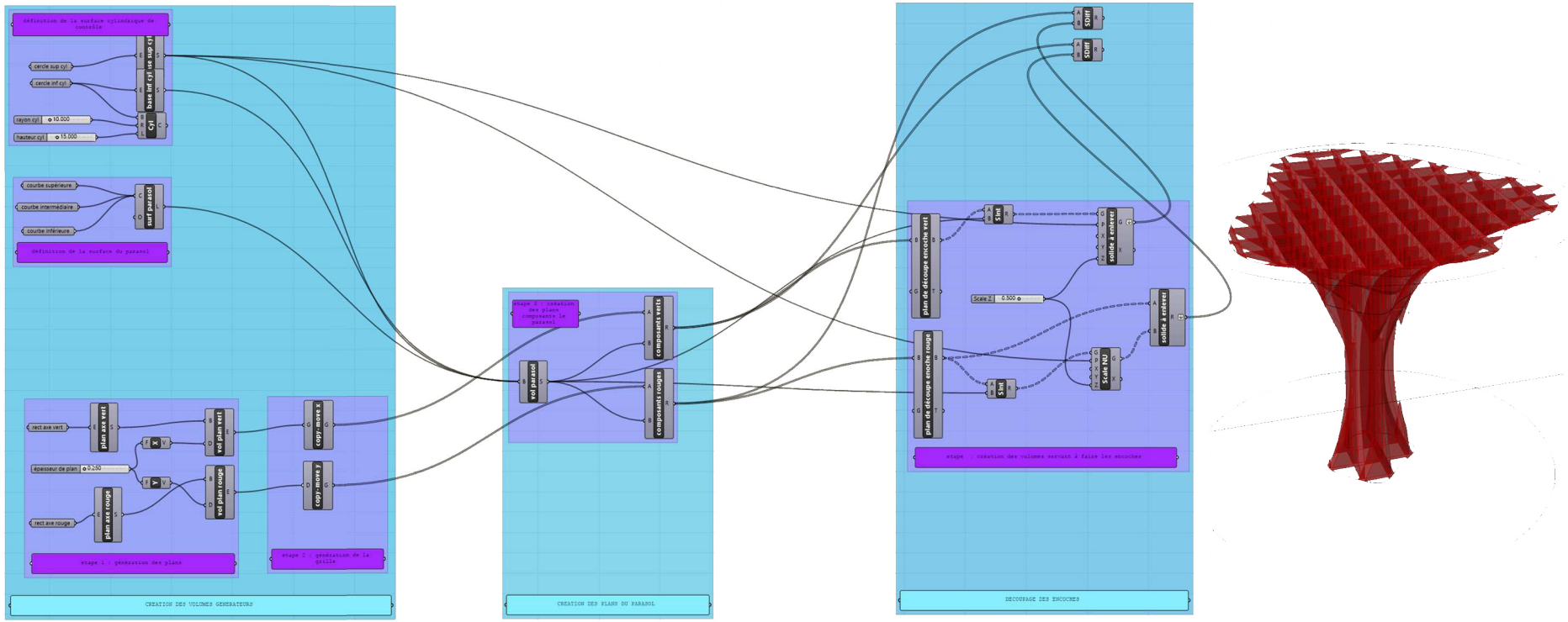
.....
.....
.....
.....

Vervier Charlotte Travail de Fin d'Étude Avril 2017

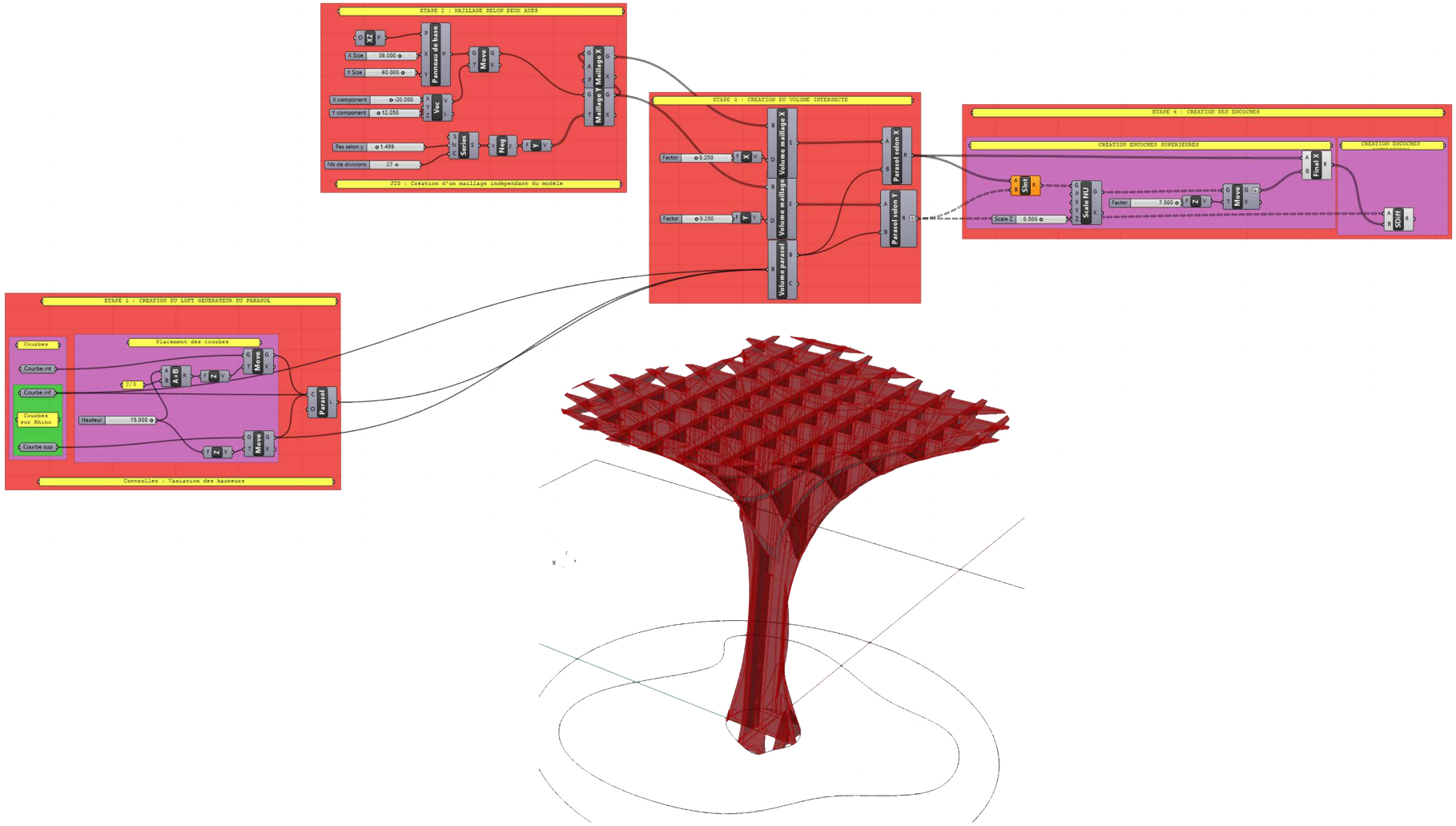
13 - Modèles réalisés lors de la première expérience – Modèle 0



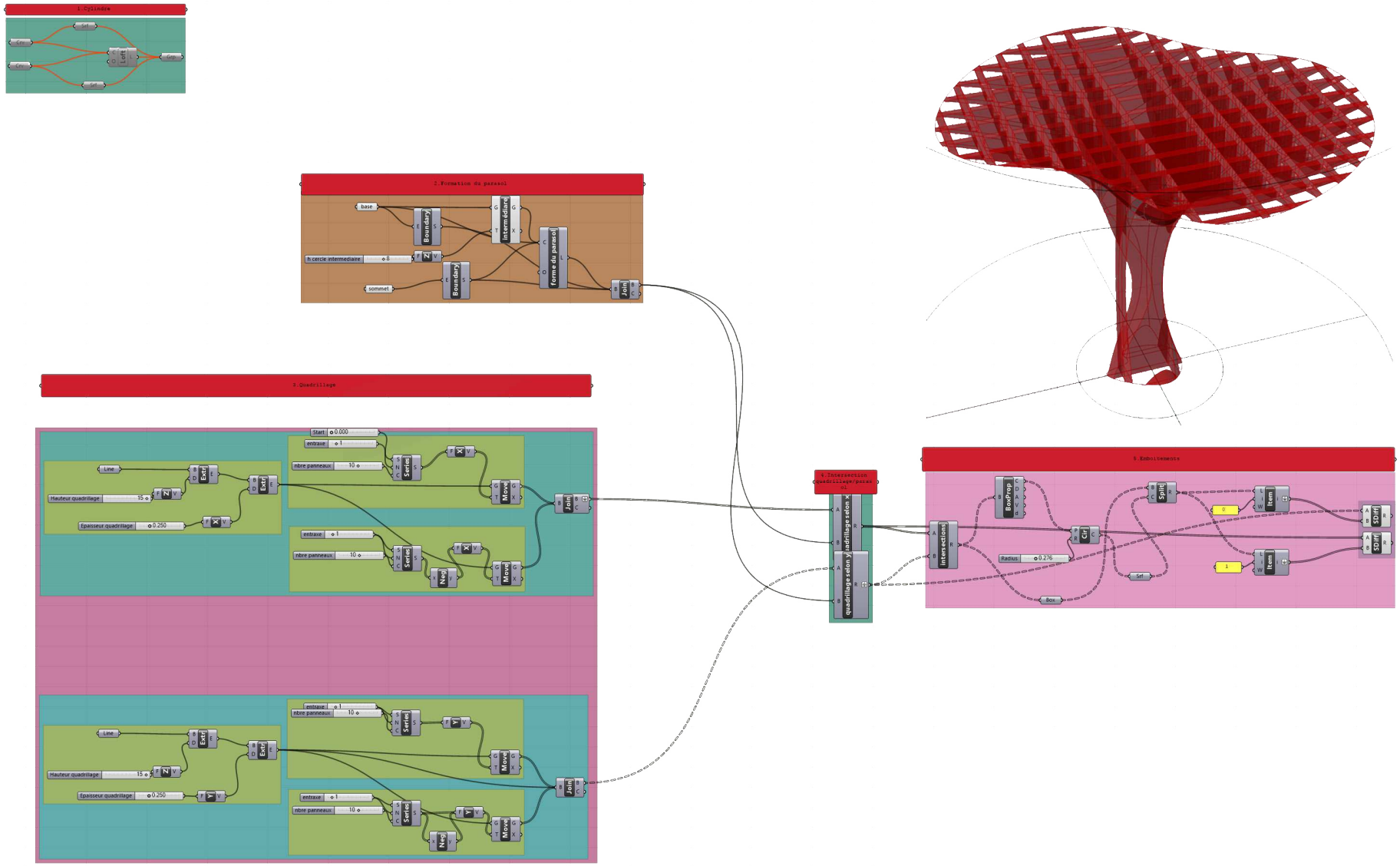
13 - Modèles réalisés lors de la première expérience – Modèle 1



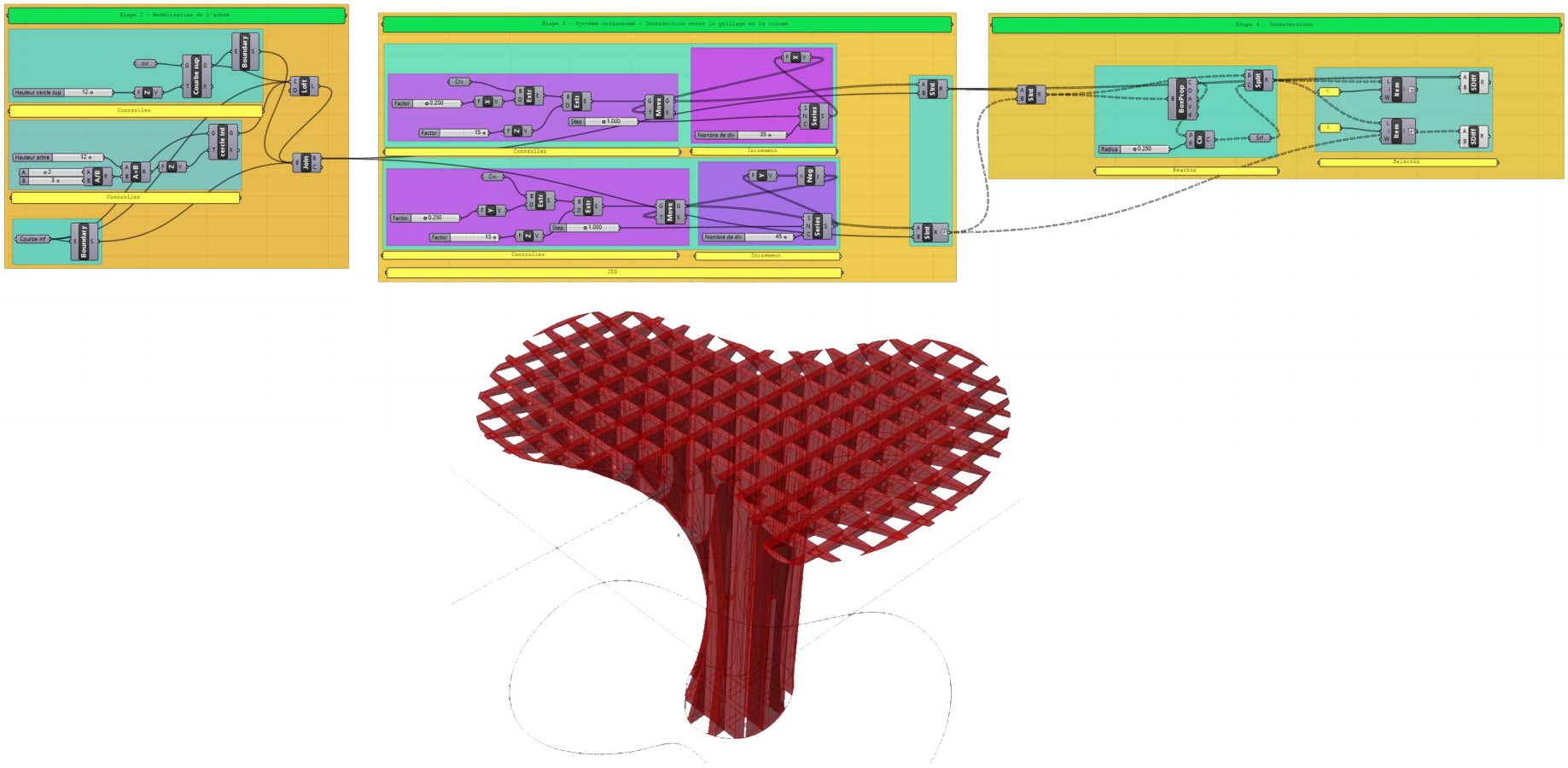
13 - Modèles réalisés lors de la première expérience – Modèle 2



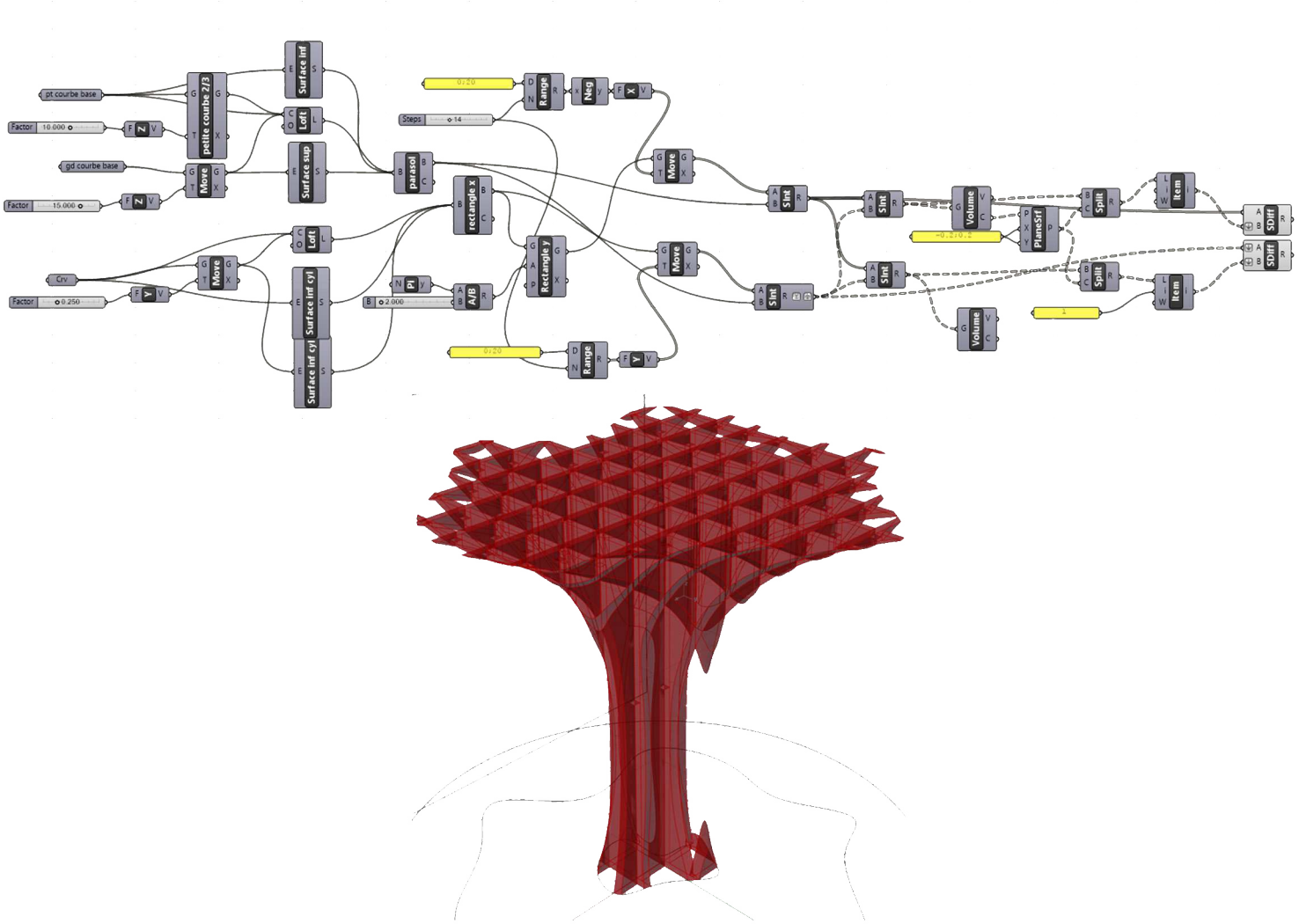
13 - Modèles réalisés lors de la première expérience – Modèle 3



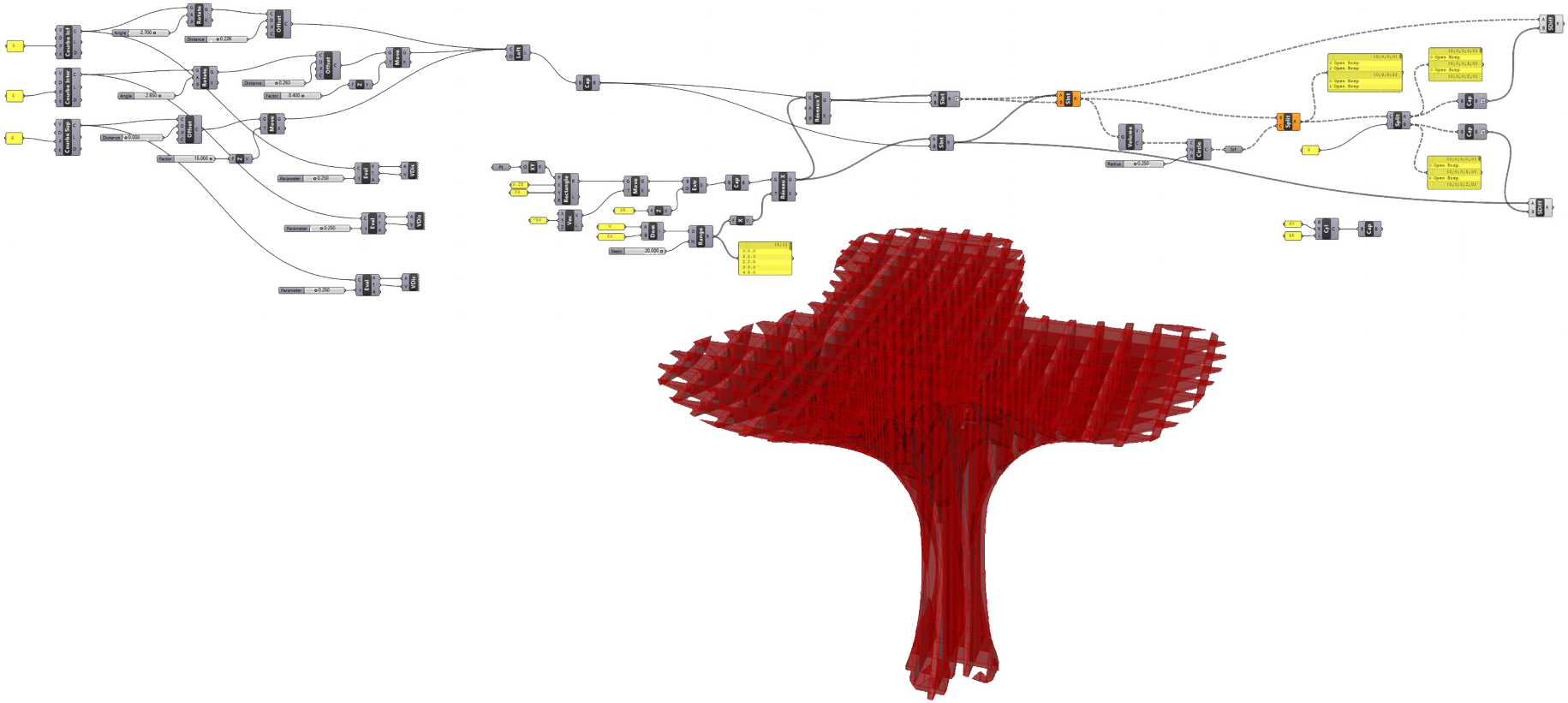
13 - Modèles réalisés lors de la première expérience – Modèle 4



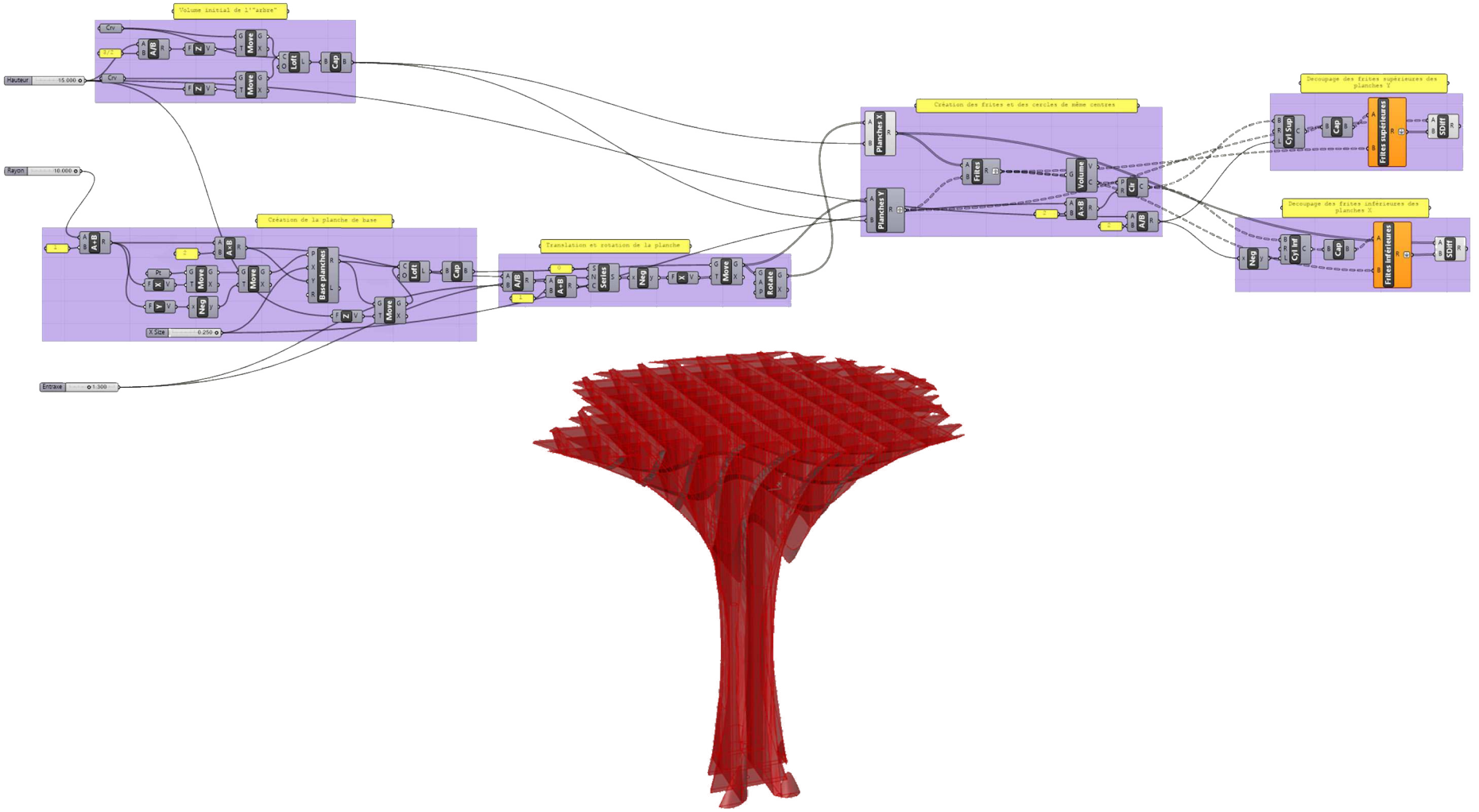
13 - Modèles réalisés lors de la première expérience – Modèle 5



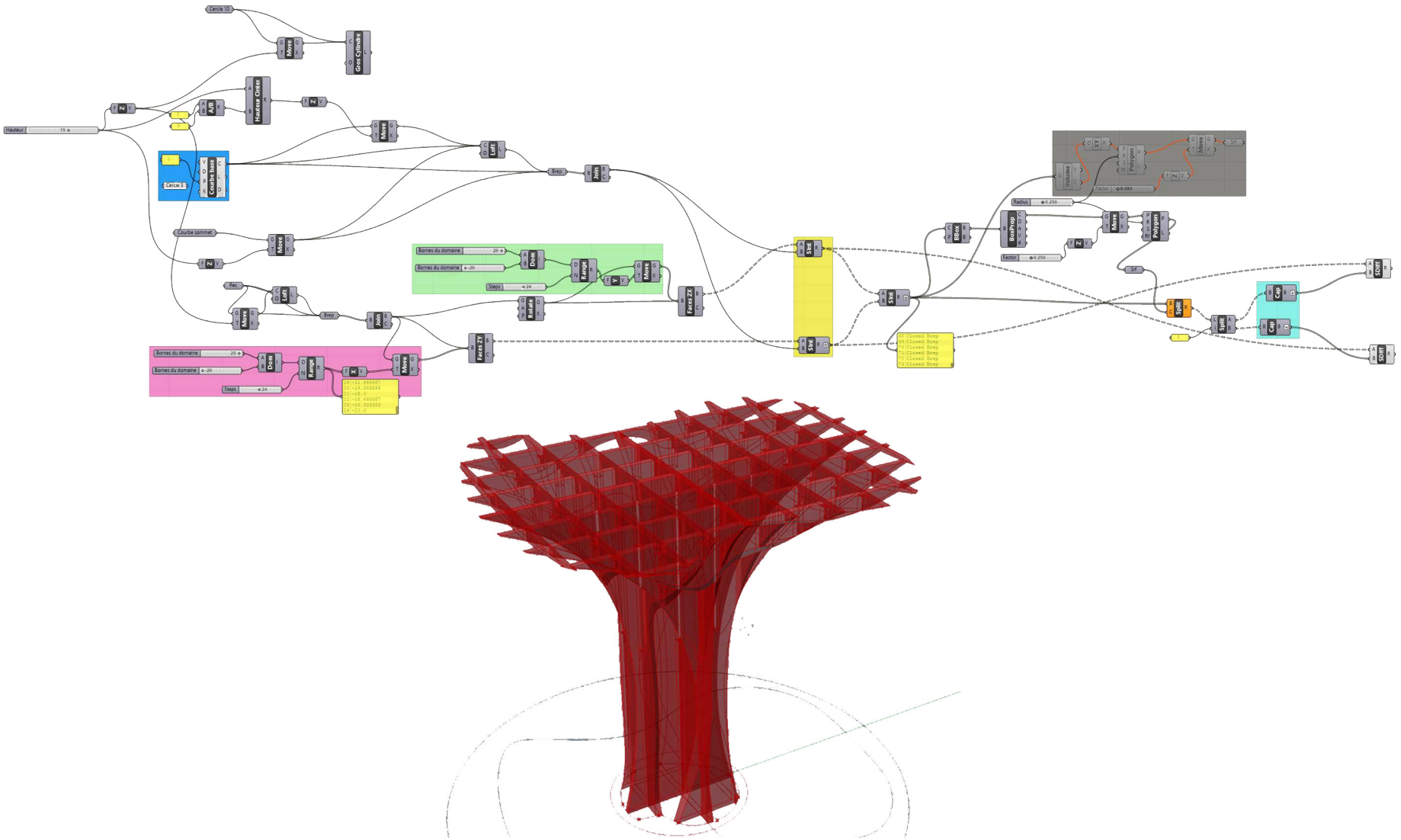
13 - Modèles réalisés lors de la première expérience – Modèle 6



13 - Modèles réalisés lors de la première expérience – Modèle 7



13 - Modèles réalisés lors de la première expérience – Modèle 8



13 - Modèles réalisés lors de la première expérience – Modèle 9