

# MODELAÇÃO E ESTRUTURAÇÃO DE SUPERFÍCIES



A superfície escolhida para ser desenvolvida neste trabalho será o exercício 3.2 realizado em aula, a Planorbis. A sua abordagem no Grasshopper será muito semelhante ao exercício do caramujo do mar desenvolvido anteriormente. No entanto, neste caso, a superfície mantém-se no plano ortogonal, não acompanhando uma espira como acontecia no caramujo.

A *Planorbis* tem um crescimento contínuo em torno de um eixo central e a sua forma geométrica é definida por meio de duas diretrizes planas (espirais concêntricas) e por geratrizes definidas por círculos existentes em planos perpendiculares ao plano que contém as directrizes



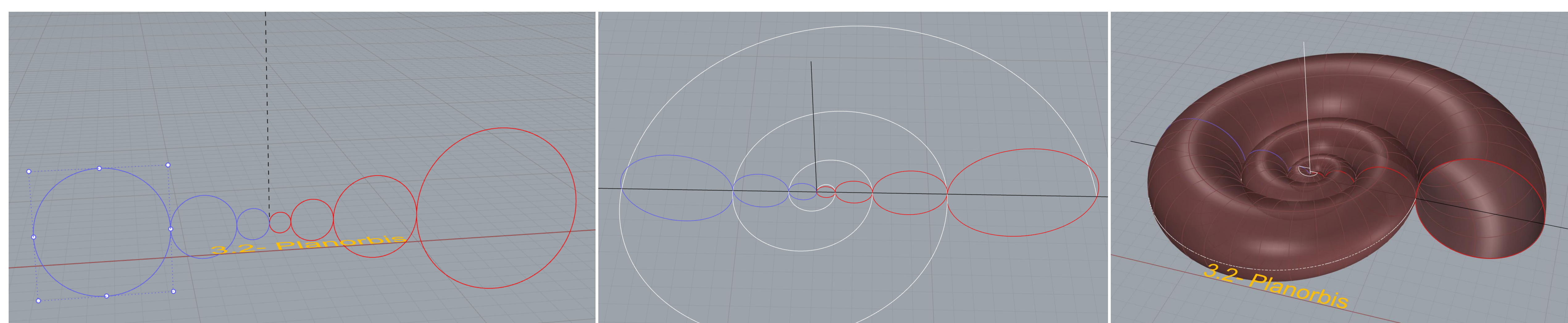
**Grasshopper**

**Trabalho 3- Planorbis**

A Planorbis foi feita no Rhino a partir de uma espiral plana construída com várias circunferências de diferentes dimensões organizadas lateralmente. Começou com a criação de uma circunferência base vertical e, a partir dela, foram definidas novas circunferências com diferentes diâmetros para ambos os lados, criando a estrutura principal da forma.

Foram criadas as diretrizes interiores através de uma meia esfera e do comando *Spiral*.

Por fim, a superfície foi construída recorrendo várias vezes ao comando *Sweep 2*, utilizando as duas diretrizes (*rails*) e duas geratrizes (*shapes*) até completar a geometria final da *Planorbis*.

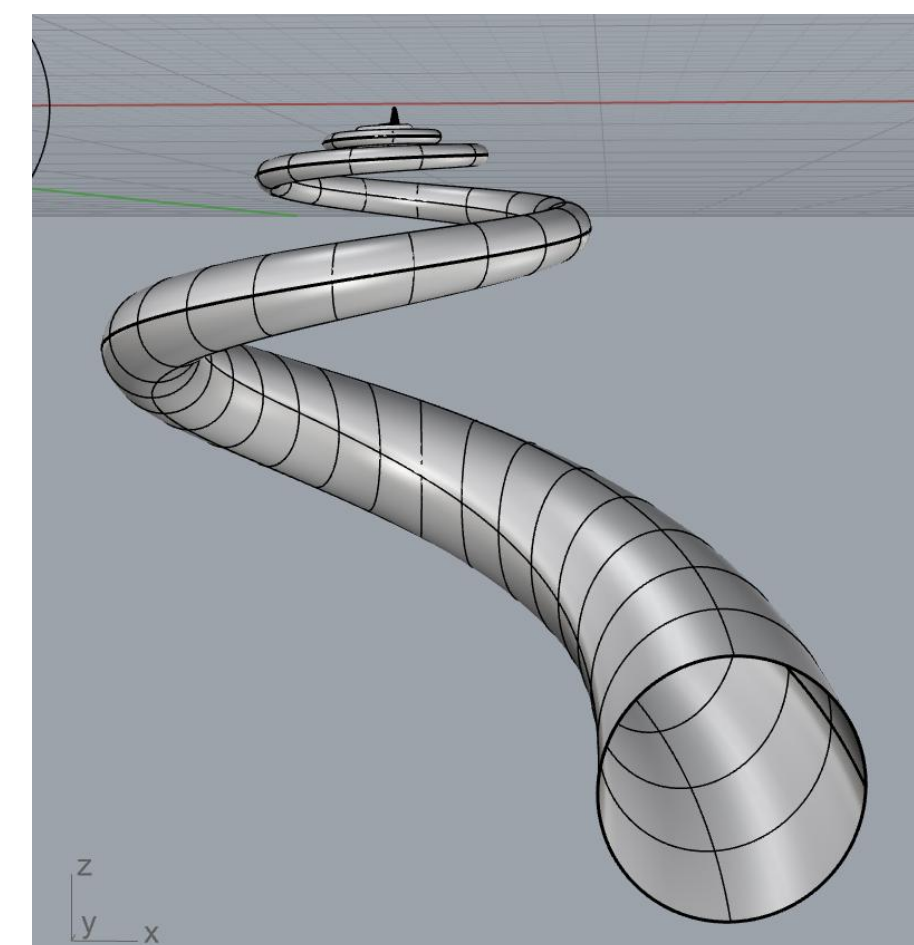
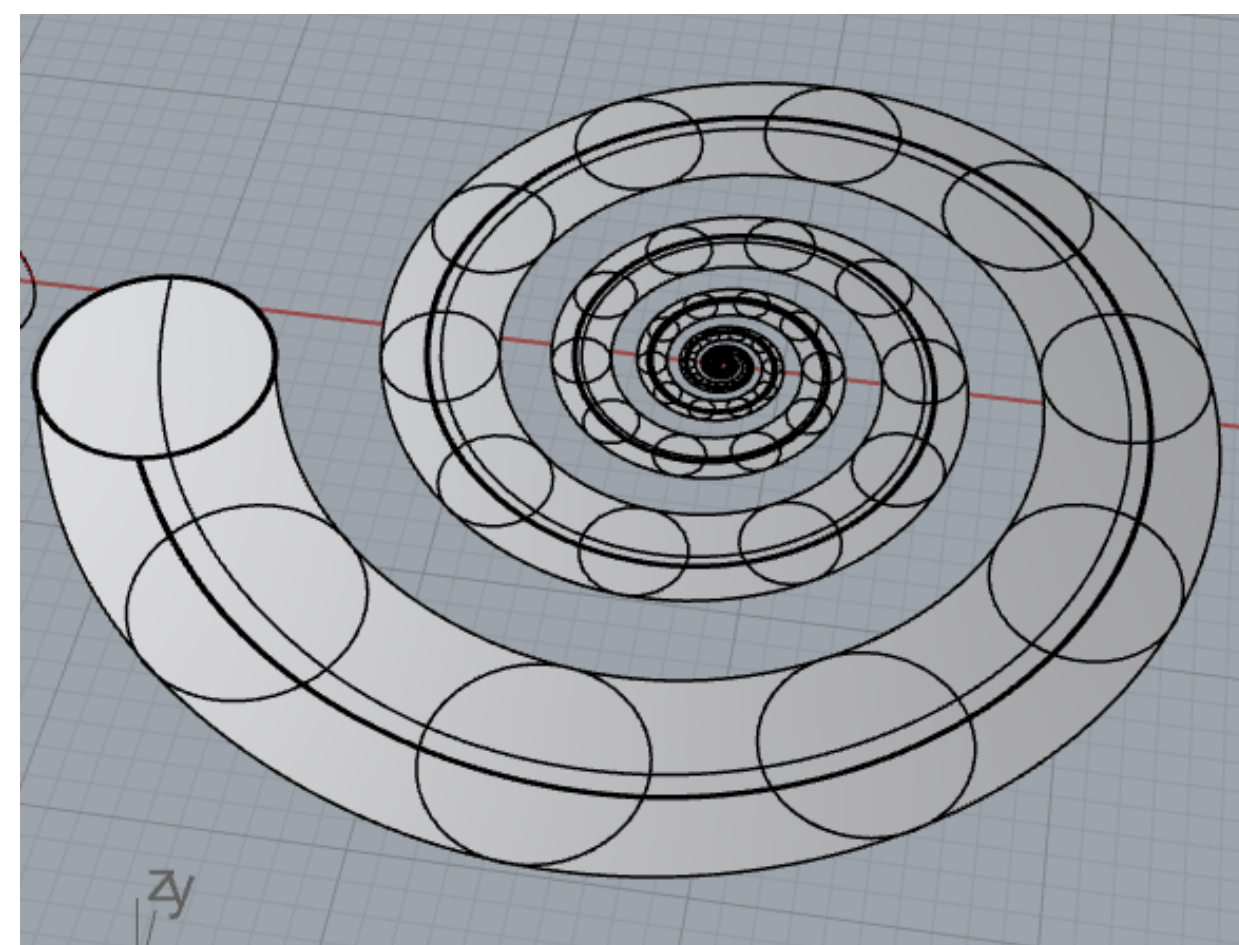
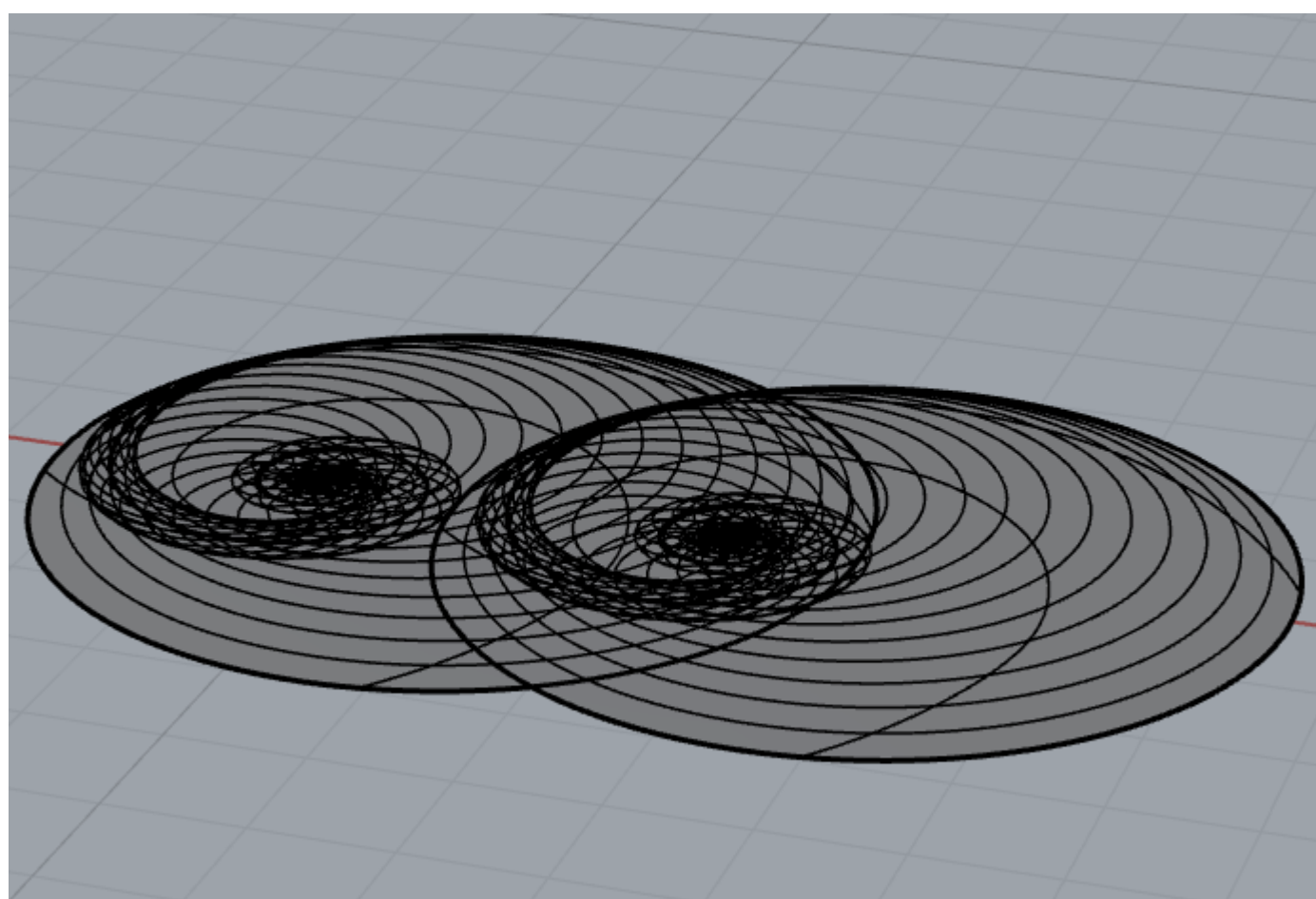
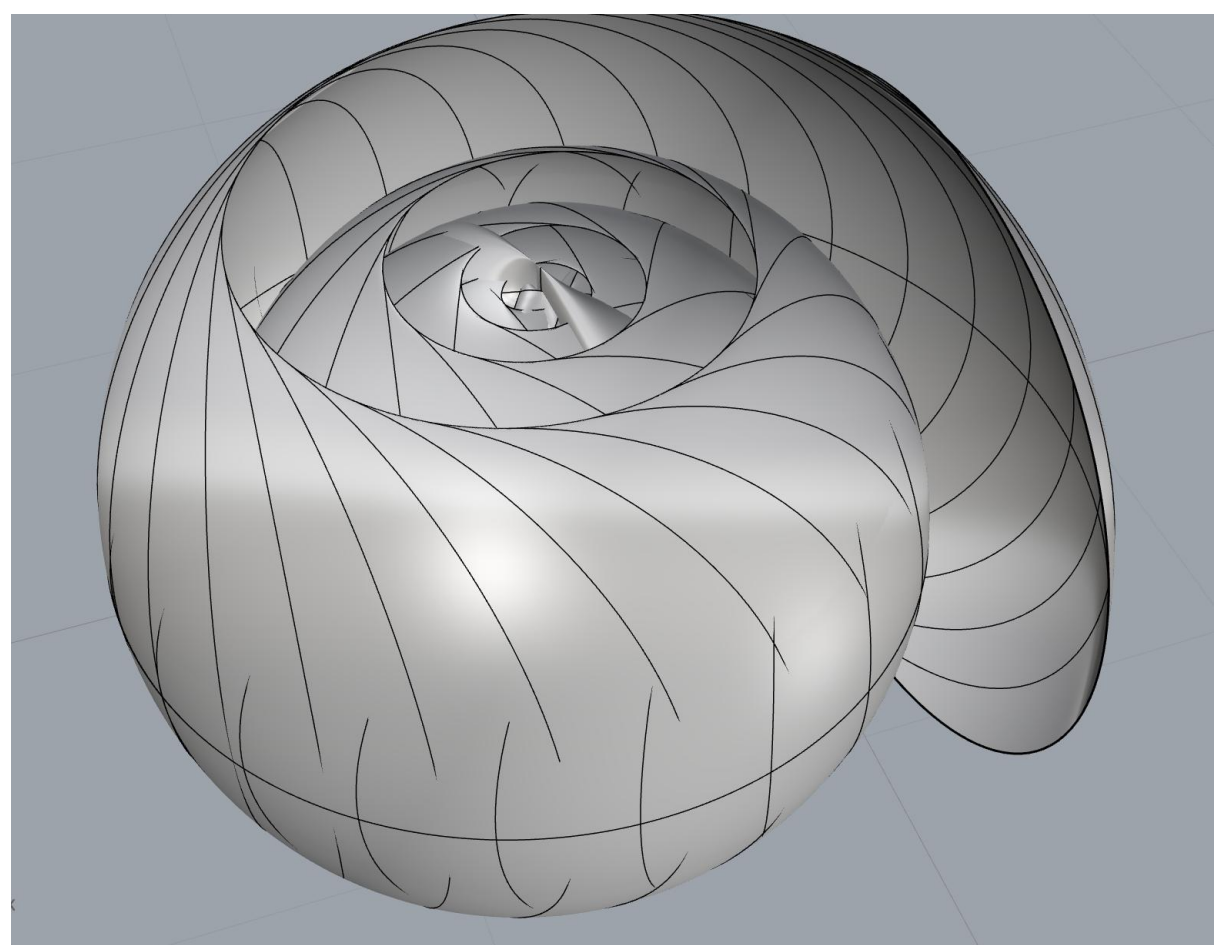
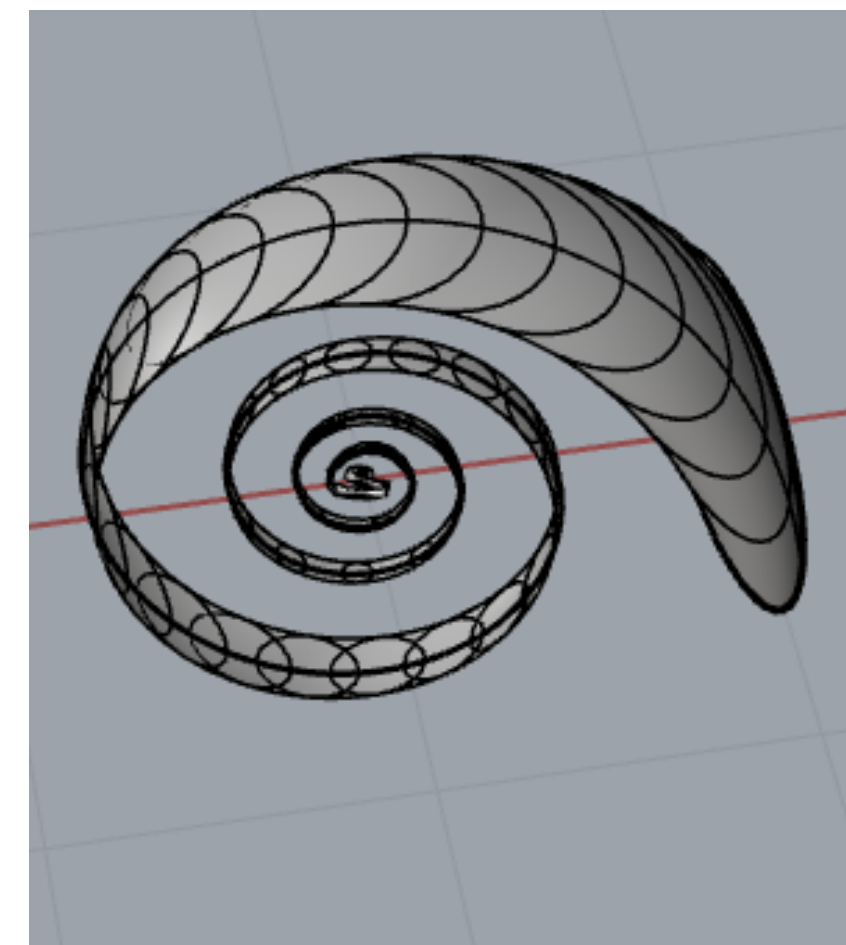
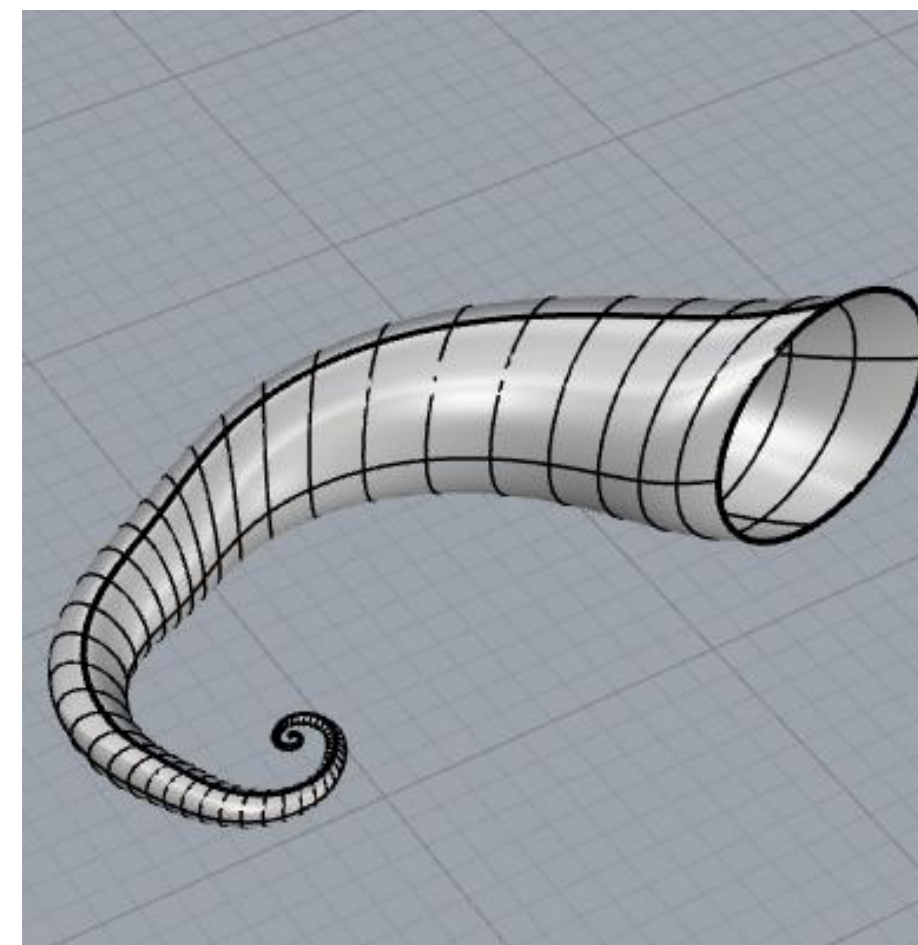
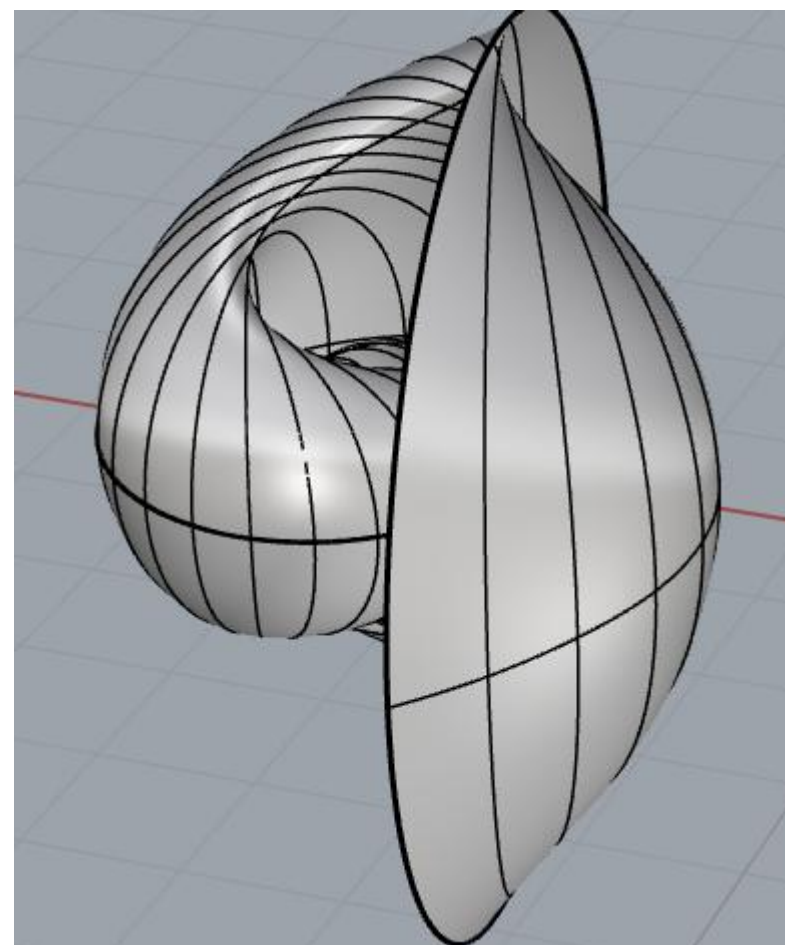
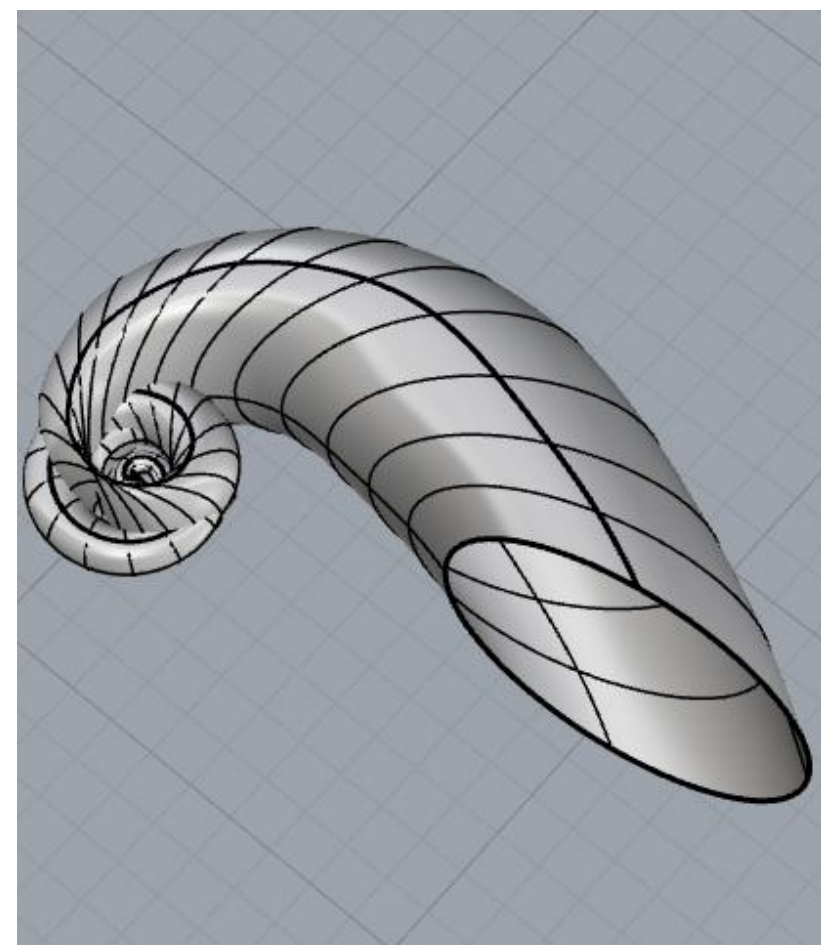
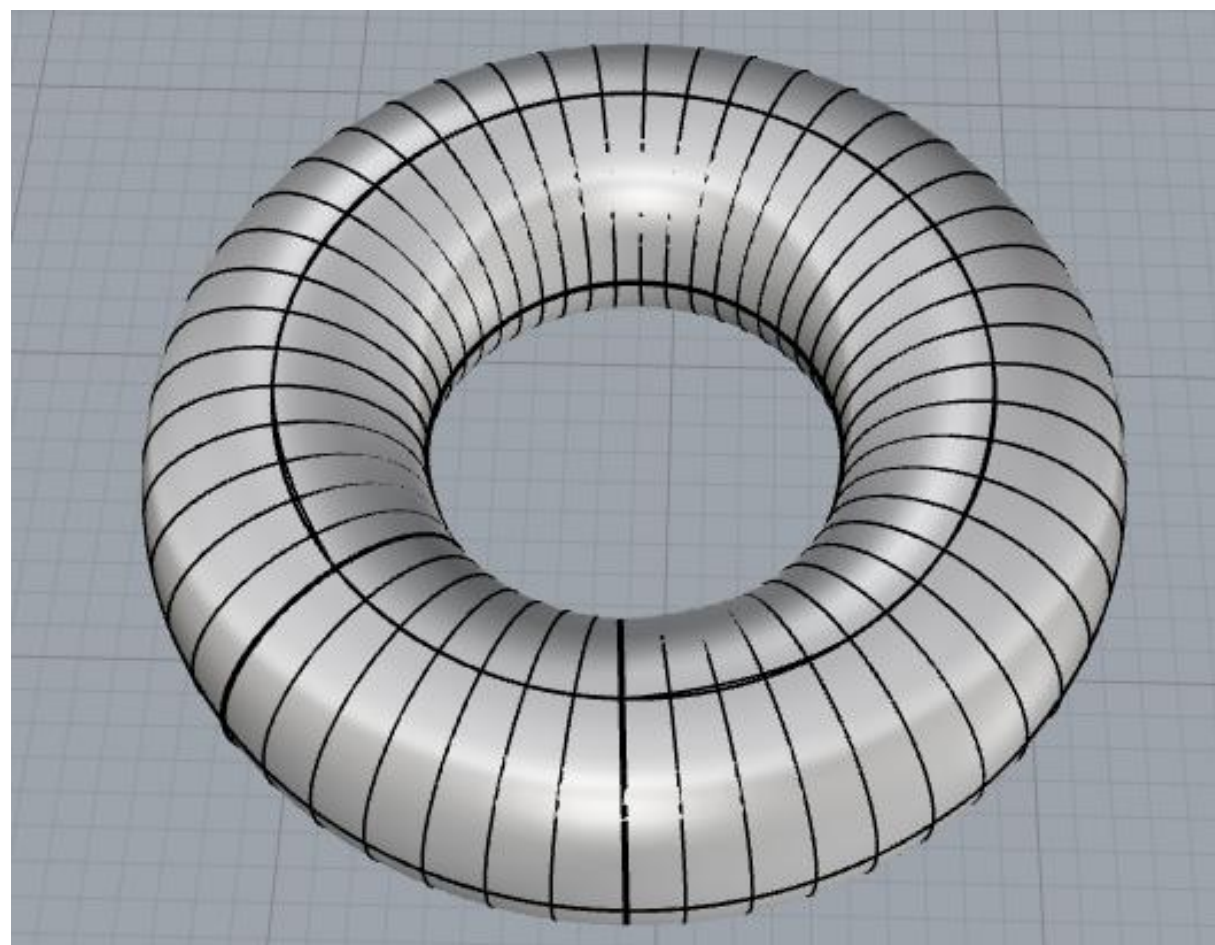


# Rhino

# Trabalho 3- Planorbis



**Processo / Tentativas anteriores até chegar à solução final:**



**Grasshopper**

**Trabalho 3- Planorbis**

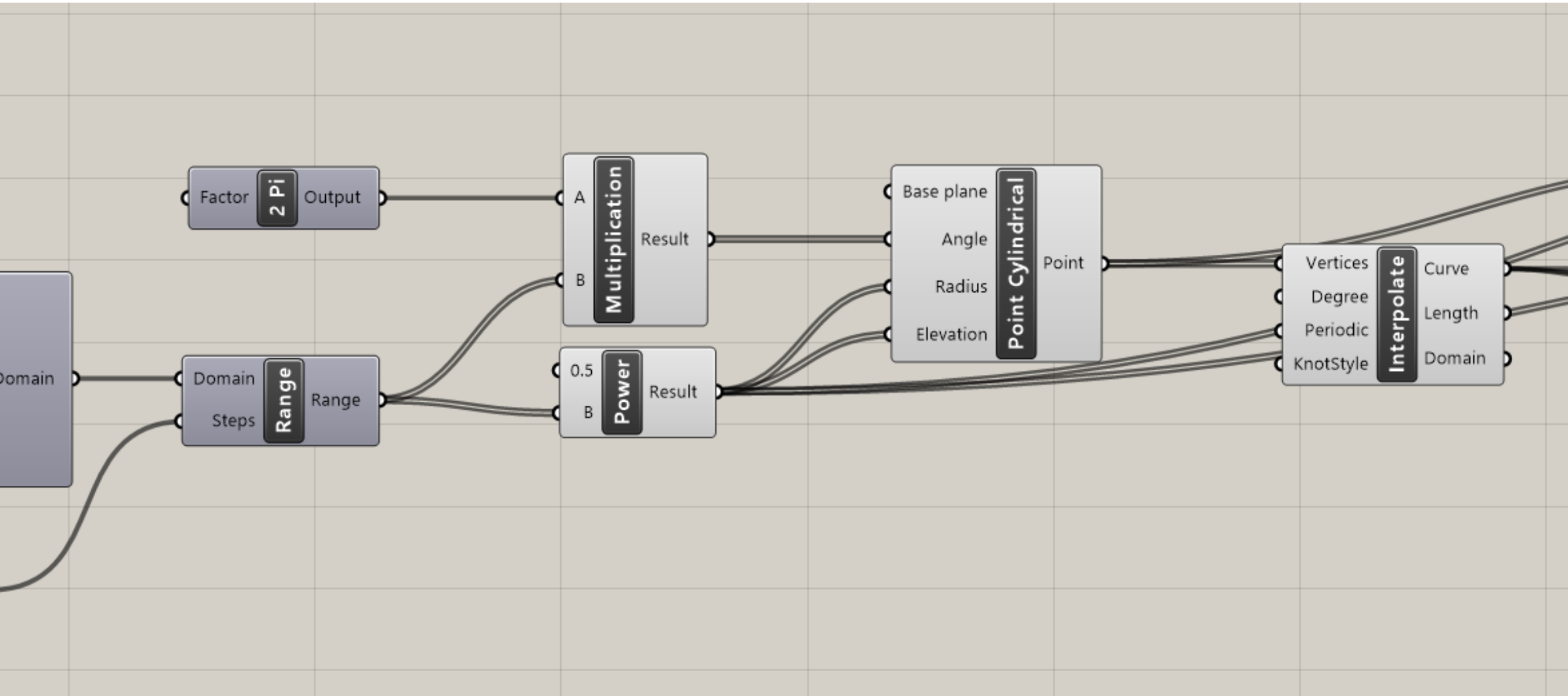
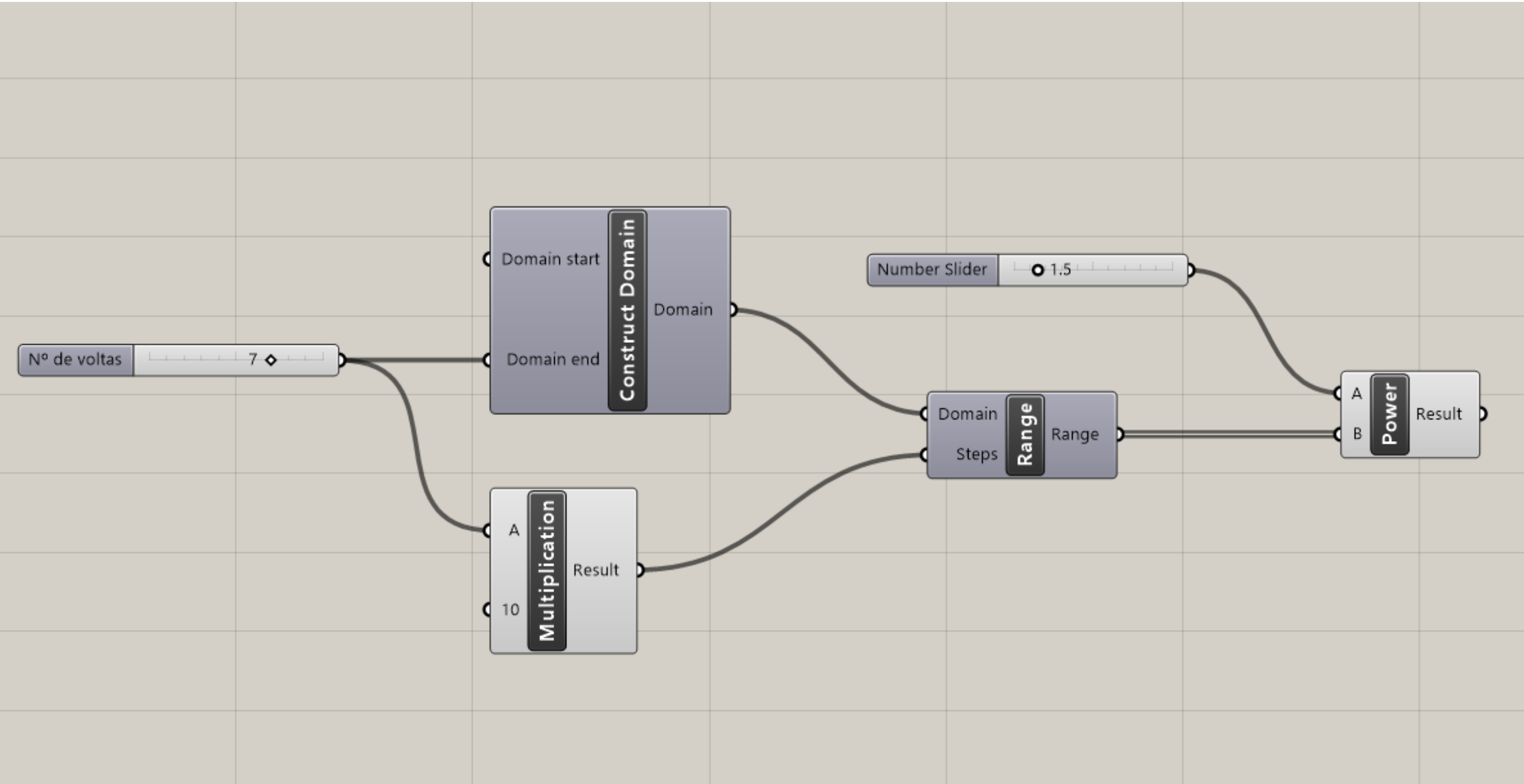
Para começar o Planorbis, primeiro tentamos criar a reta de orientação que será plana. Utilizou-se o *Construct Domain* para criar um intervalo de valores que representa o número total de voltas da curva. O valor do *Slider*, que inicialmente era 7 (mais tarde alterado para 9), foi ligado ao *Construct Domain*, criando assim um intervalo entre 0 e 7.

Depois, esse valor foi multiplicado por 10 para aumentar o número de divisões e dar mais detalhe à curva. Os dois resultados foram ligados ao *Range*, que divide o intervalo em várias partes iguais e cria uma sequência. A sequência entrou num *Multiplication*. O valor foi multiplicado por  $2\pi$ , que representa uma volta completa (360 graus). Isto permitiu transformar os valores em ângulos para a rotação.

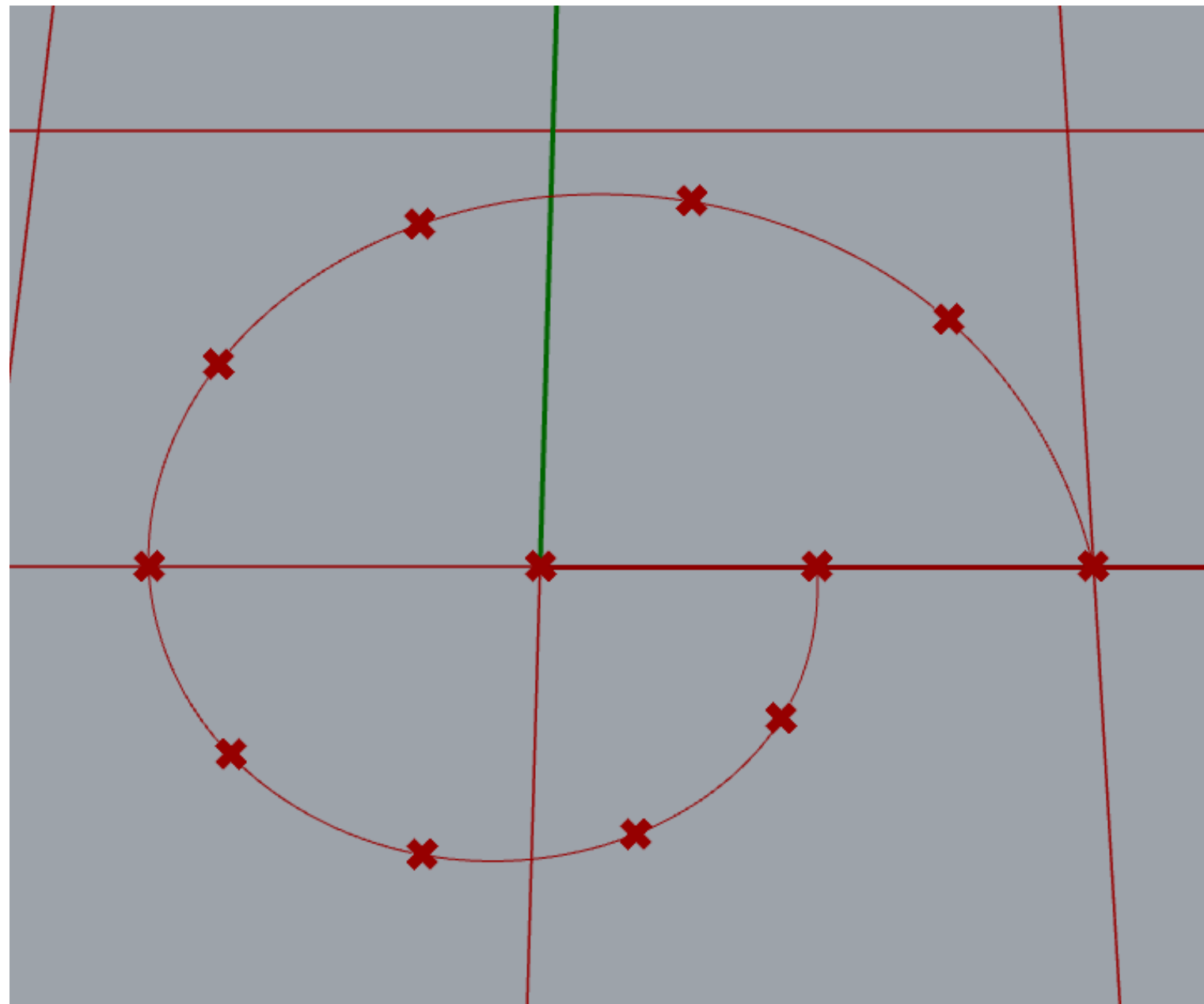
Ao mesmo tempo, a sequência também entrou num componente *Power*, onde foi utilizado o valor 0.5, criando uma progressão não linear e uma forma mais orgânica.

Os resultados foram depois ligados ao *Point Cylindrical*. Os ângulos entraram no input *Angle* e os valores do *Power* foram utilizados no *Radius*. Nesta vez, a entrada *Elevation* não pode ficar com ligações, para que a forma fique no plano horizontal.

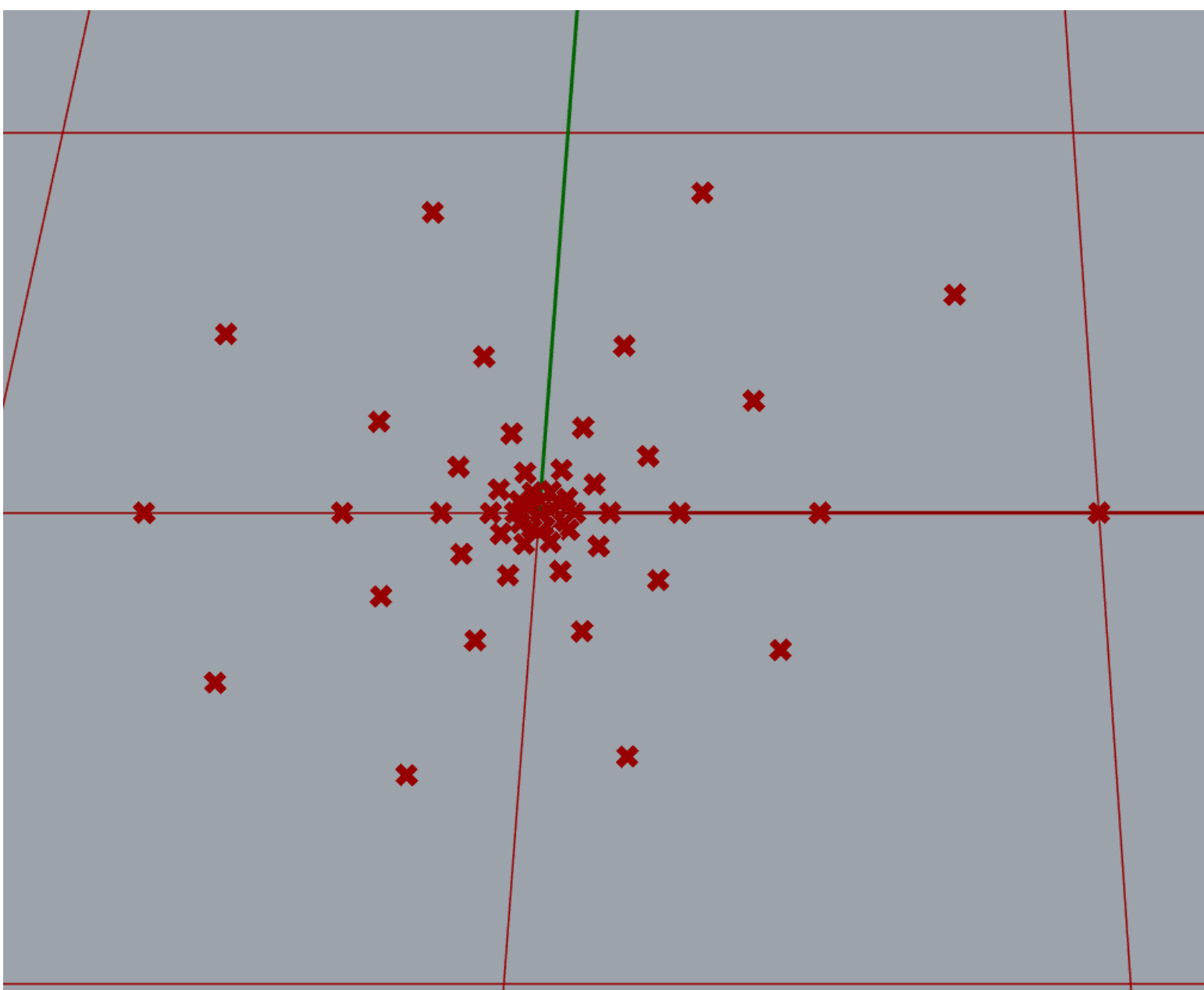
Este componente criou os pontos da espiral no espaço. Por fim, os pontos foram ligados ao *Interpolate*, que uniu todos os pontos através de uma curva contínua.



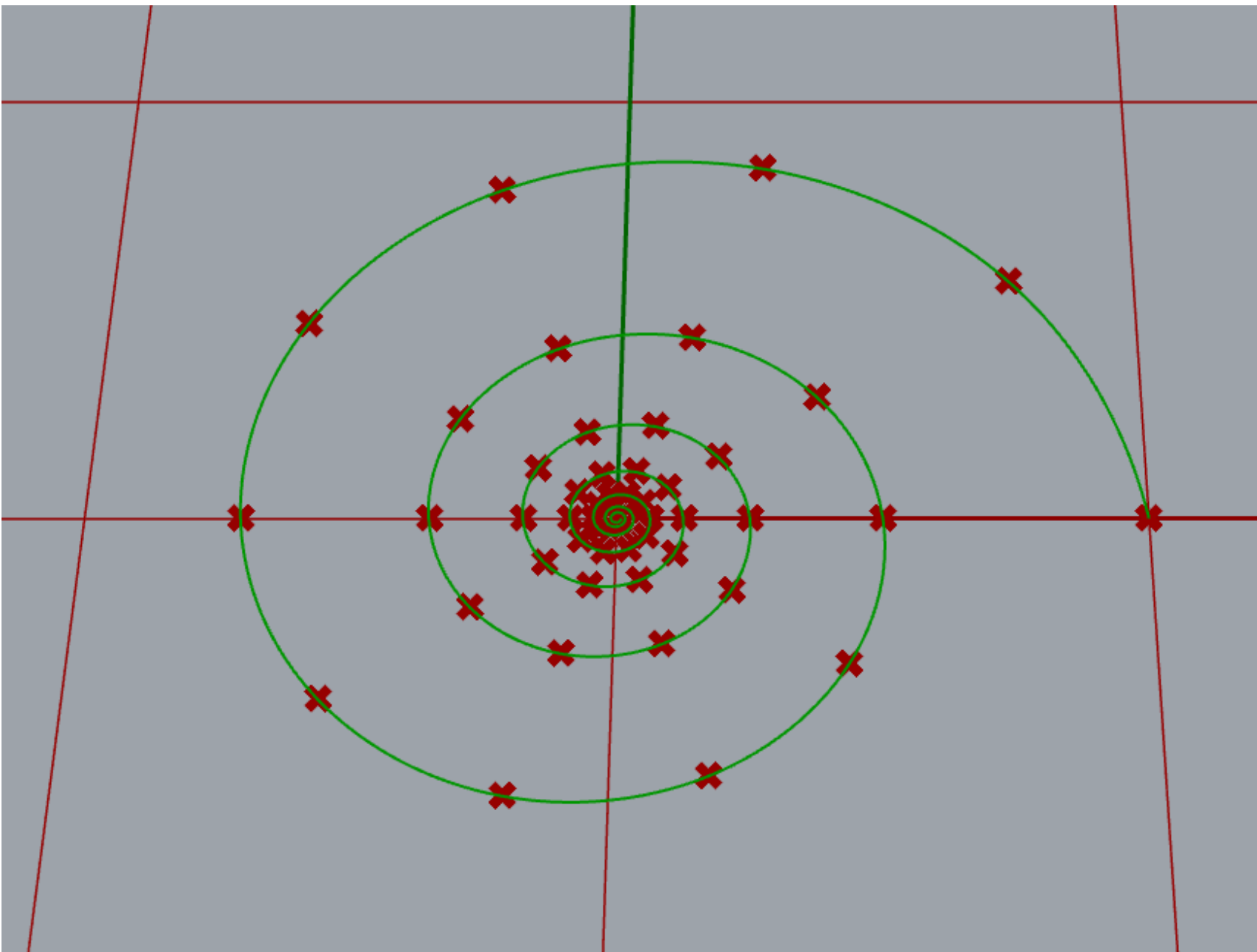
Comandos:



Number Slider  
Controla quantas voltas a curva faz



Point Cylindrical  
Distribui os pontos



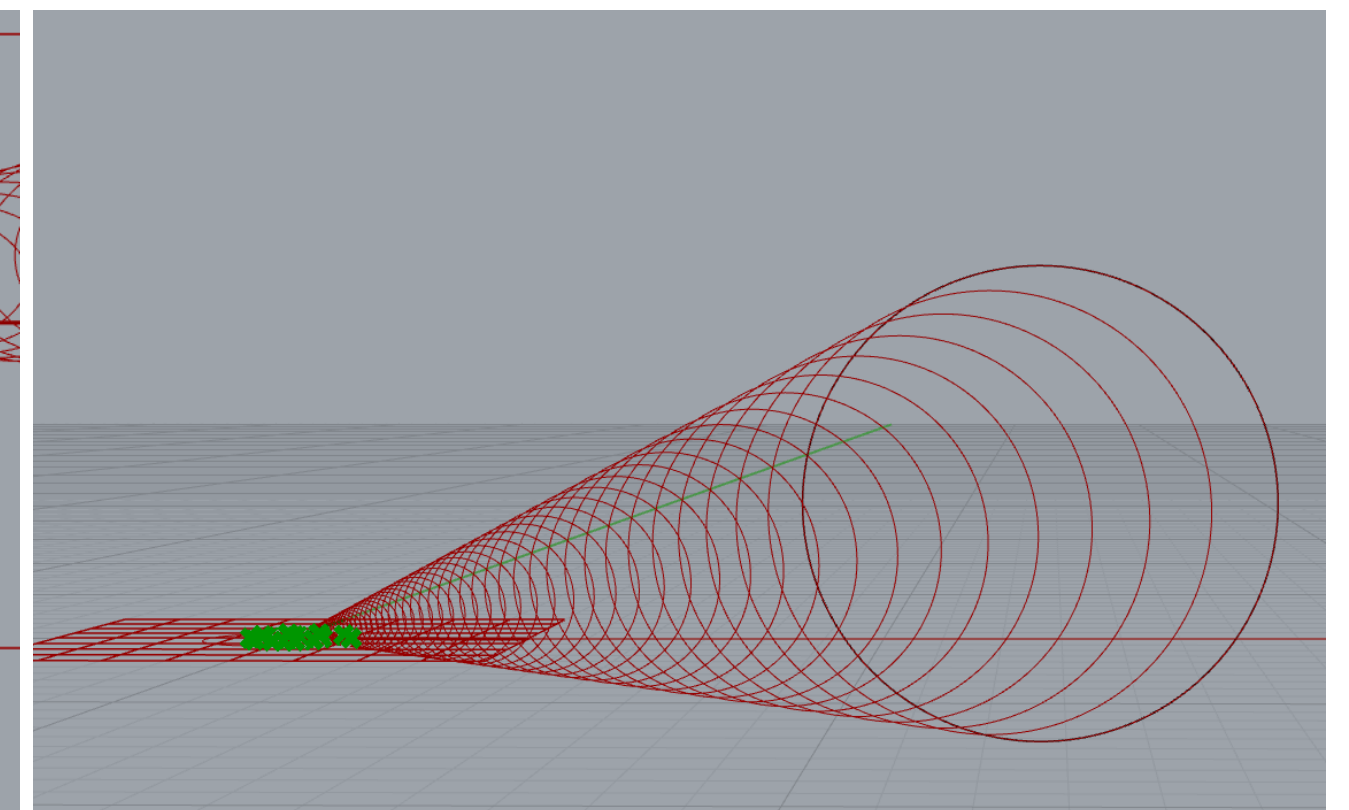
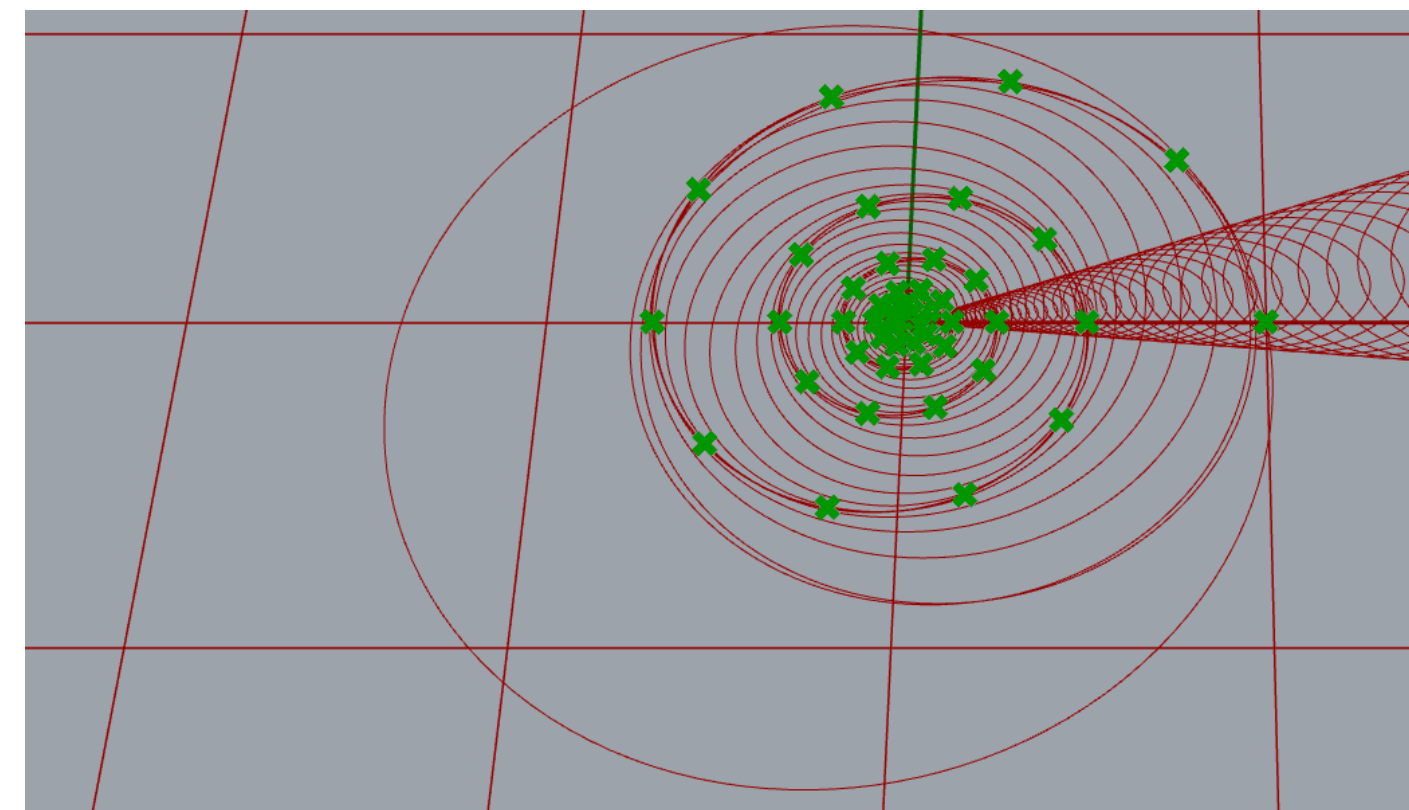
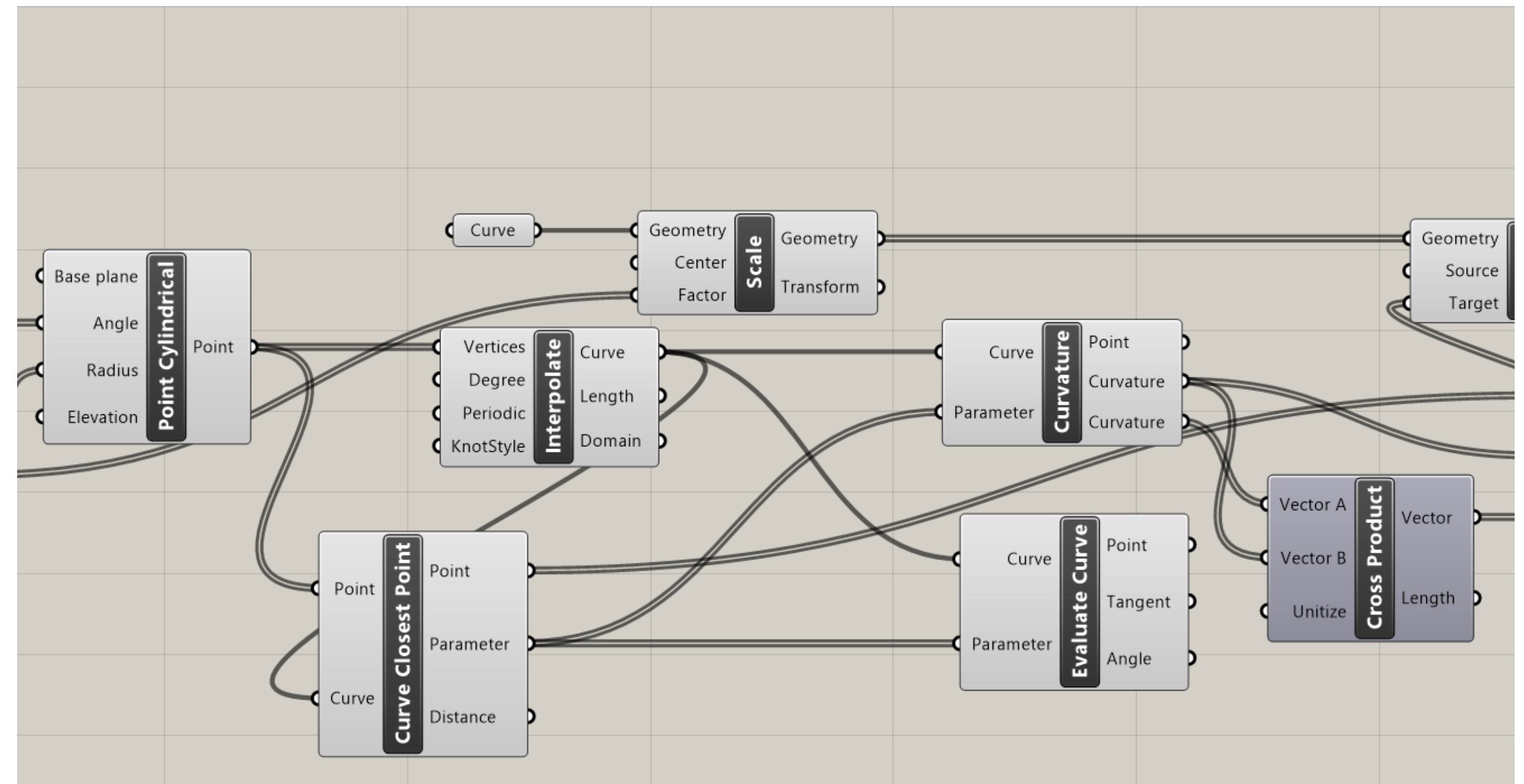
Interpolate  
Cria a curva que passa em todos os pontos

Os pontos criados no Point Cylindrical também vão para o Curve Closest Point, que também recebe a curva criada no interpolate. Isto calcular a posição de cada ponto ao longo da linha.

Depois este parametro é ligado a duas coisas: o Curvature, para descobrir a direção da curvatura da linha em cada ponto. E o segundo é o Evaluate Curve, para encontrar a a orientação para onde a linha segue.

A seguir, a direção da curvatura e a direção tangente ligam-se às entradas do Cross Product para criar um vetor perpendicular.

Adiciona-se uma Curve que entra no Scale. Esta curva se´ra definida como uma circunferência que já tinha sido feita no modle para definir a abertura da concha.



The image displays a visual programming network with the following nodes and connections:

- Curvature Node:**
  - Inputs: Curve, Parameter
  - Output: Point
  - Output: Curvature
  - Output: Curvature
- Evaluate Curve Node:**
  - Inputs: Curve, Parameter
  - Output: Point
  - Output: Tangent
  - Output: Angle
- Cross Product Node:**
  - Inputs: Vector A, Vector B, Unitize
  - Output: Vector
  - Output: Vector
  - Output: Length

Connections in the network include:

- A 'Curve' node connected to the 'Curve' input of both 'Curvature' and 'Evaluate Curve'.
- A 'Parameter' node connected to the 'Parameter' input of both 'Curvature' and 'Evaluate Curve'.
- The 'Point' output of 'Curvature' connected to the 'Point' input of 'Evaluate Curve'.
- The 'Curvature' output of 'Curvature' connected to the 'Vector A' input of 'Cross Product'.
- The 'Curvature' output of 'Curvature' connected to the 'Vector B' input of 'Cross Product'.
- The 'Tangent' output of 'Evaluate Curve' connected to the 'Unitize' input of 'Cross Product'.
- The 'Angle' output of 'Evaluate Curve' connected to the 'Unitize' input of 'Cross Product'.
- The 'Vector' output of 'Cross Product' connected to the 'Vector' input of another 'Cross Product' node.
- The 'Vector' output of 'Cross Product' connected to the 'Vector' input of another 'Cross Product' node.
- The 'Length' output of 'Cross Product' connected to the 'Length' input of another 'Cross Product' node.

The diagram illustrates a network of interconnected nodes, each representing a function or data structure. The nodes are arranged in a grid-like fashion, with lines connecting them to show dependencies or relationships. The nodes are:

- Curve**: A simple node.
- Geometry**: A node with sub-nodes **Center** and **Factor**.
- Scale**: A node with a sub-node **Transform**.
- Geometry**: A node with sub-nodes **Source** and **Target**.
- Orient**: A node with a sub-node **Transform**.
- Curves**: A node with a sub-node **Options**.
- Loft**: A node with a sub-node **Loft**.
- Interpolate**: A node with sub-nodes **Vertices**, **Degree**, **Periodic**, and **KnotStyle**.
- Curve**: A node with sub-nodes **Length** and **Domain**.
- Curvature**: A node with sub-nodes **Point**, **Curvature**, and **Curvature**.
- Evaluate Curve**: A node with sub-nodes **Point**, **Tangent**, and **Angle**.
- Cross Product**: A node with sub-nodes **Vector A**, **Vector B**, and **Unitize**.
- Construct Plane**: A node with sub-nodes **Origin**, **X-Axis**, and **Y-Axis**.
- Plane**: A node with a sub-node **Plane**.

The connections between the nodes are as follows:

- Curve** connects to **Geometry** (Center, Factor).
- Geometry** (Center, Factor) connects to **Scale** (Transform).
- Scale** (Transform) connects to **Geometry** (Source, Target).
- Geometry** (Source, Target) connects to **Orient** (Transform).
- Orient** (Transform) connects to **Curves** (Options) and **Loft** (Loft).
- Curves** (Options) connects to **Loft** (Loft).
- Interpolate** (Vertices, Degree, Periodic, KnotStyle) connects to **Curve** (Length, Domain).
- Curve** (Length, Domain) connects to **Curvature** (Point, Curvature, Curvature).
- Curvature** (Point, Curvature, Curvature) connects to **Evaluate Curve** (Point, Tangent, Angle).
- Evaluate Curve** (Point, Tangent, Angle) connects to **Cross Product** (Vector A, Vector B, Unitize).
- Cross Product** (Vector A, Vector B, Unitize) connects to **Construct Plane** (Origin, X-Axis, Y-Axis).
- Construct Plane** (Origin, X-Axis, Y-Axis) connects to **Plane** (Plane).

The diagram illustrates the relationships between various geometric and mathematical functions, organized into a network of nodes and connections. The functions are represented as nodes, and the connections show how they interact or depend on each other.

**Functions and their inputs/outputs:**

- Curve**: Inputs: Geometry, Center, Factor. Output: Geometry.
- Scale**: Inputs: Geometry, Transform. Output: Geometry.
- Interpolate**: Inputs: Vertices, Degree, Periodic, KnotStyle. Outputs: Curve, Length, Domain.
- Evaluate Curve**: Inputs: Curve, Parameter. Outputs: Point, Tangent, Angle.
- Curvature**: Inputs: Curve, Parameter. Outputs: Point, Curvature, Curvature.
- Cross Product**: Inputs: Vector A, Vector B, Unitize. Outputs: Vector, Length.
- Orient**: Inputs: Geometry, Source, Target. Output: Geometry.
- Construct Plane**: Inputs: Origin, X-Axis, Y-Axis. Output: Plane.
- Loft**: Inputs: Curves, Options. Output: Loft.

**Connections:**

- Curve** connects to **Scale** and **Interpolate**.
- Scale** connects to **Curve** and **Orient**.
- Interpolate** connects to **Curve**, **Evaluate Curve**, and **Curvature**.
- Evaluate Curve** connects to **Curve**, **Curvature**, and **Construct Plane**.
- Curvature** connects to **Curve**, **Evaluate Curve**, and **Construct Plane**.
- Cross Product** connects to **Construct Plane** and **Orient**.
- Orient** connects to **Curve**, **Curvature**, and **Loft**.
- Construct Plane** connects to **Orient** and **Loft**.
- Loft** connects to **Curve**, **Curvature**, and **Orient**.

The diagram illustrates the relationships between various geometric and mathematical functions, organized into a network of nodes and connections. The functions are represented as nodes, and the connections show how they interact or depend on each other.

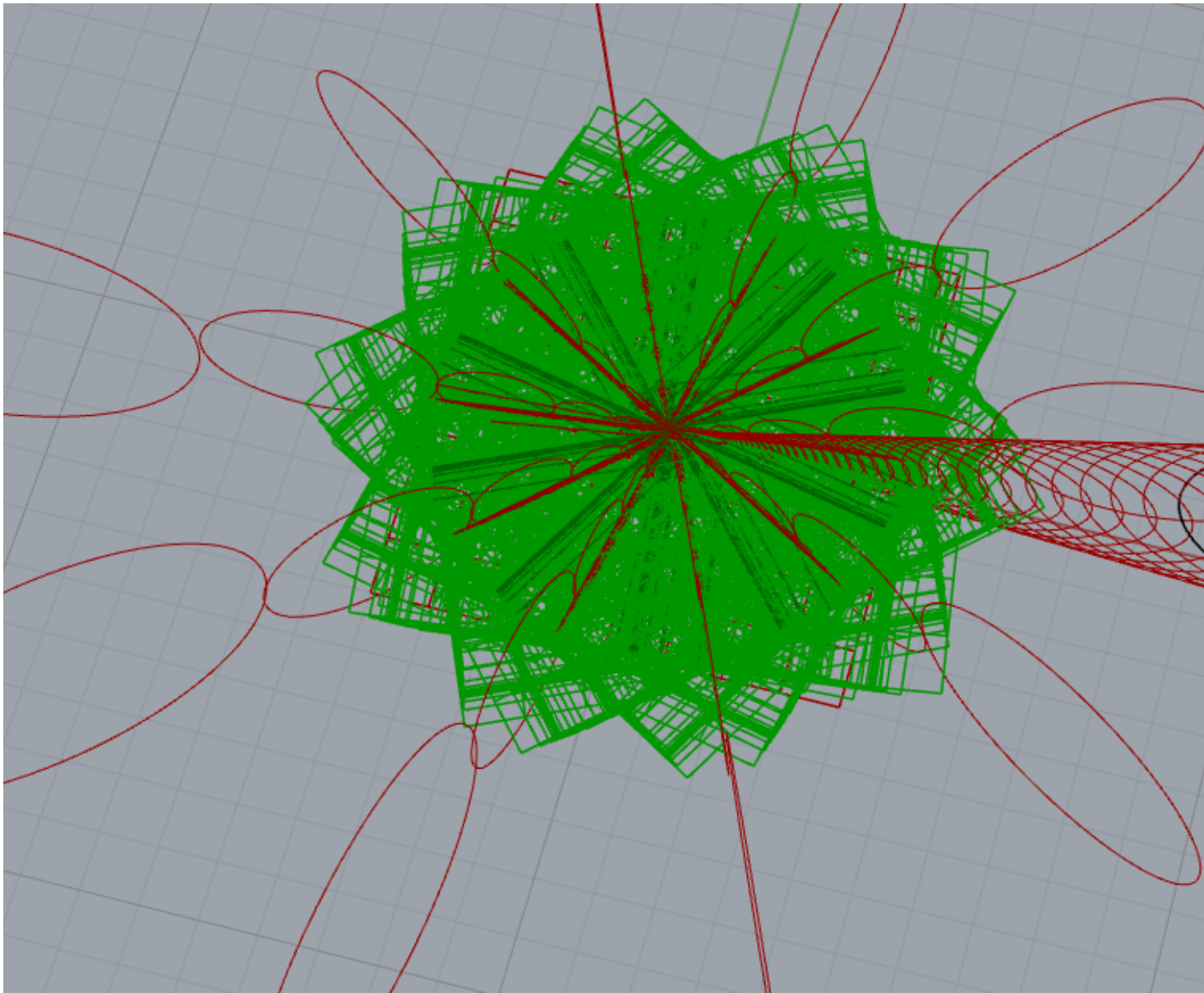
**Functions and their inputs/outputs:**

- Curve**: Inputs: Geometry, Center, Factor. Output: Geometry.
- Scale**: Inputs: Geometry, Transform. Output: Geometry.
- Interpolate**: Inputs: Vertices, Degree, Periodic, KnotStyle. Outputs: Curve, Length, Domain.
- Evaluate Curve**: Inputs: Curve, Parameter. Outputs: Point, Tangent, Angle.
- Curvature**: Inputs: Curve, Parameter. Outputs: Point, Curvature, Curvature.
- Cross Product**: Inputs: Vector A, Vector B, Unitize. Outputs: Vector, Length.
- Orient**: Inputs: Geometry, Source, Target. Output: Geometry.
- Construct Plane**: Inputs: Origin, X-Axis, Y-Axis. Output: Plane.
- Loft**: Inputs: Curves, Options. Output: Loft.

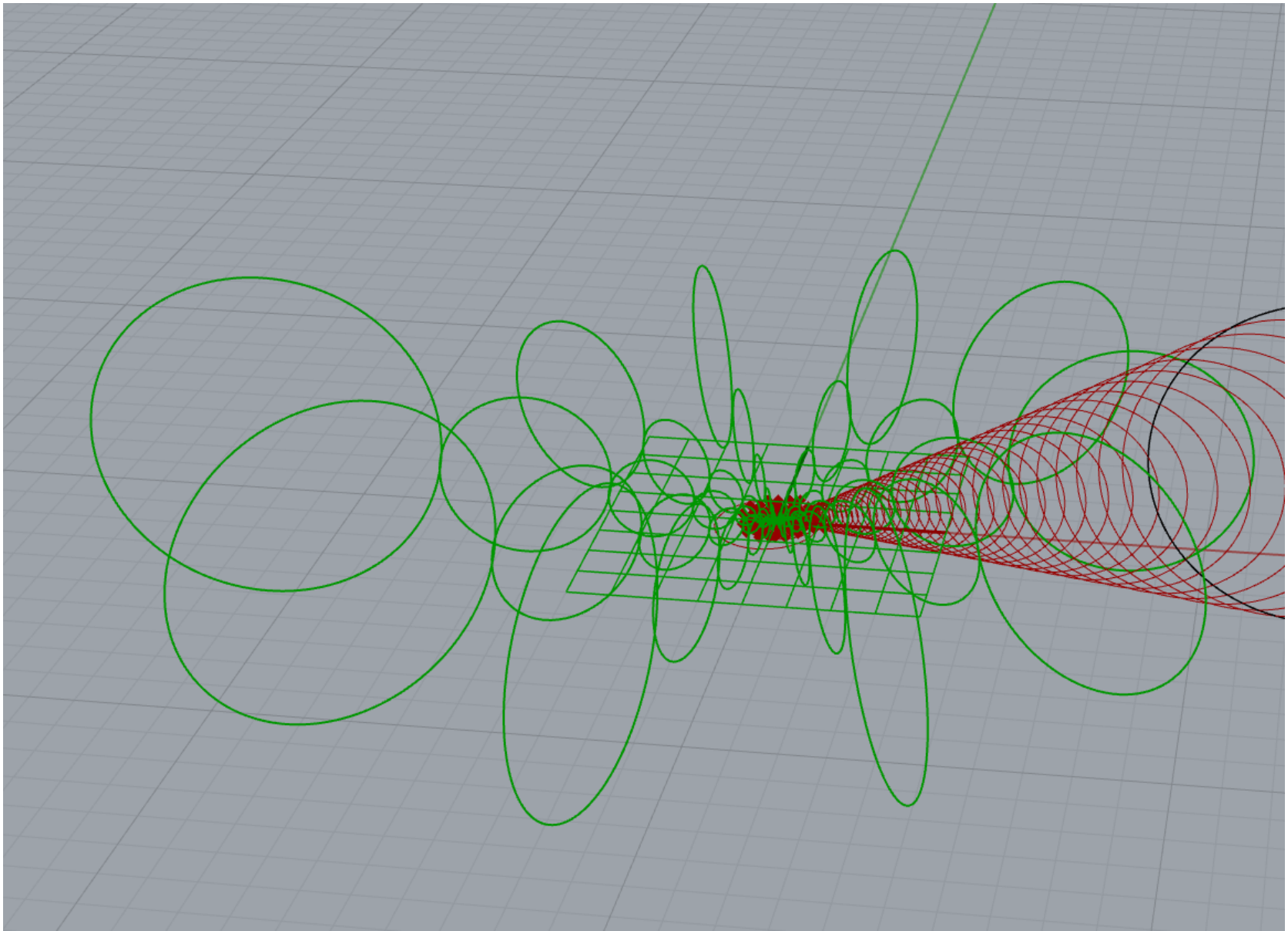
**Connections:**

- Curve** connects to **Scale** and **Interpolate**.
- Scale** connects to **Curve** and **Orient**.
- Interpolate** connects to **Curve**, **Evaluate Curve**, and **Curvature**.
- Evaluate Curve** connects to **Curve**, **Curvature**, and **Construct Plane**.
- Curvature** connects to **Curve**, **Evaluate Curve**, and **Construct Plane**.
- Cross Product** connects to **Construct Plane** and **Orient**.
- Orient** connects to **Curve**, **Curvature**, and **Loft**.
- Construct Plane** connects to **Orient** and **Loft**.
- Loft** connects to **Curve**, **Curvature**, and **Orient**.

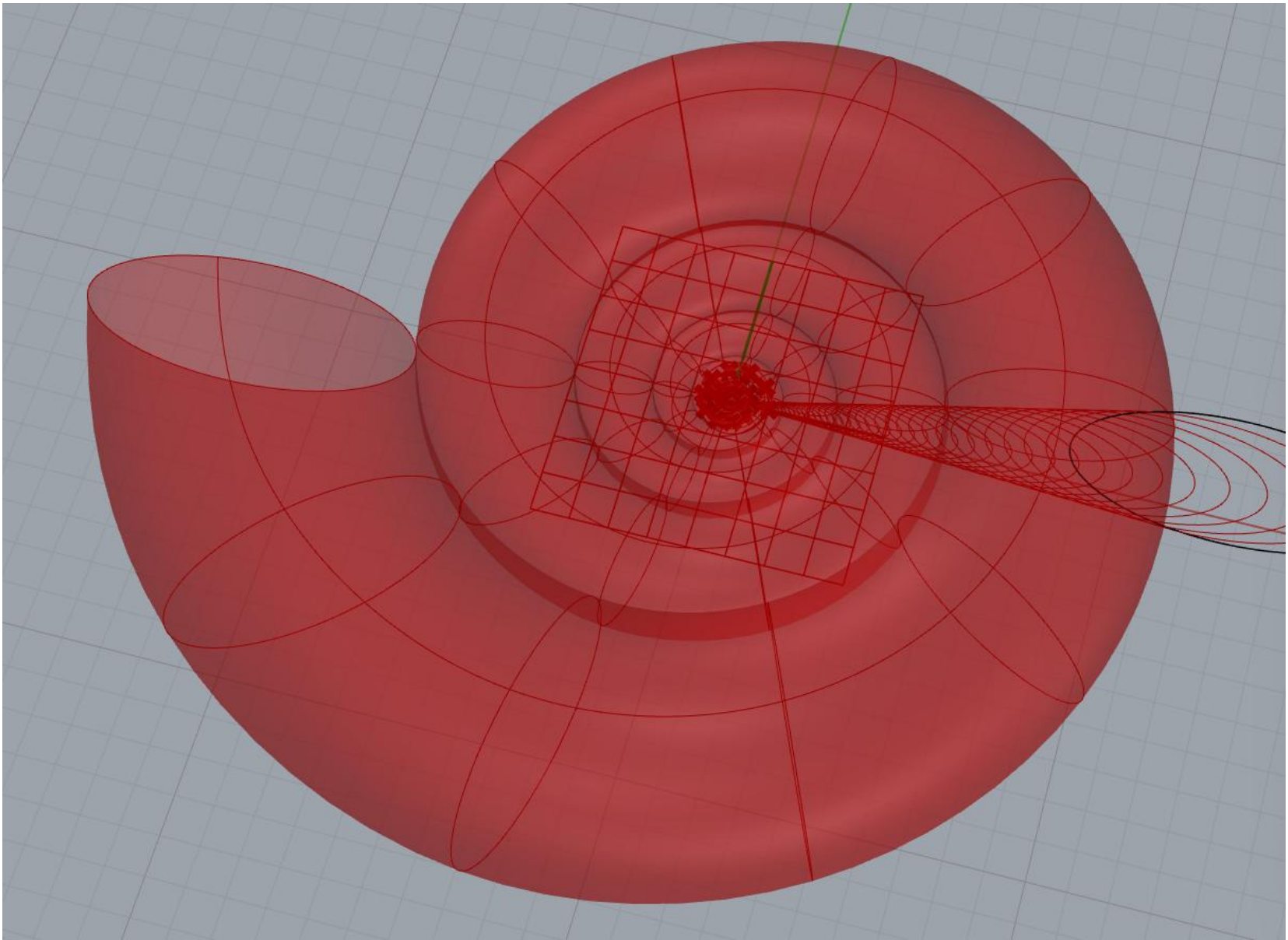
a



Construct planes  
Planos orientadores

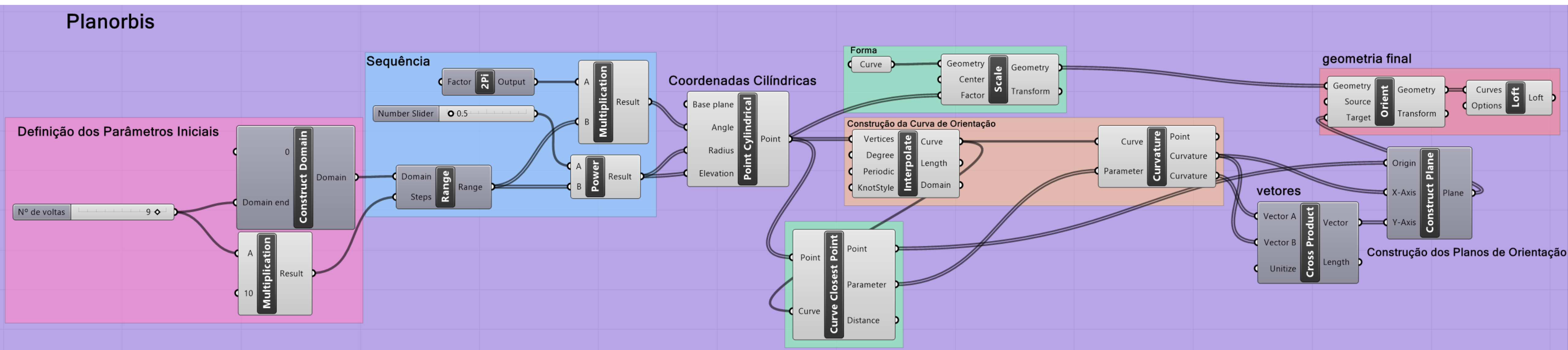


Oriental  
Distribuição das circunferências

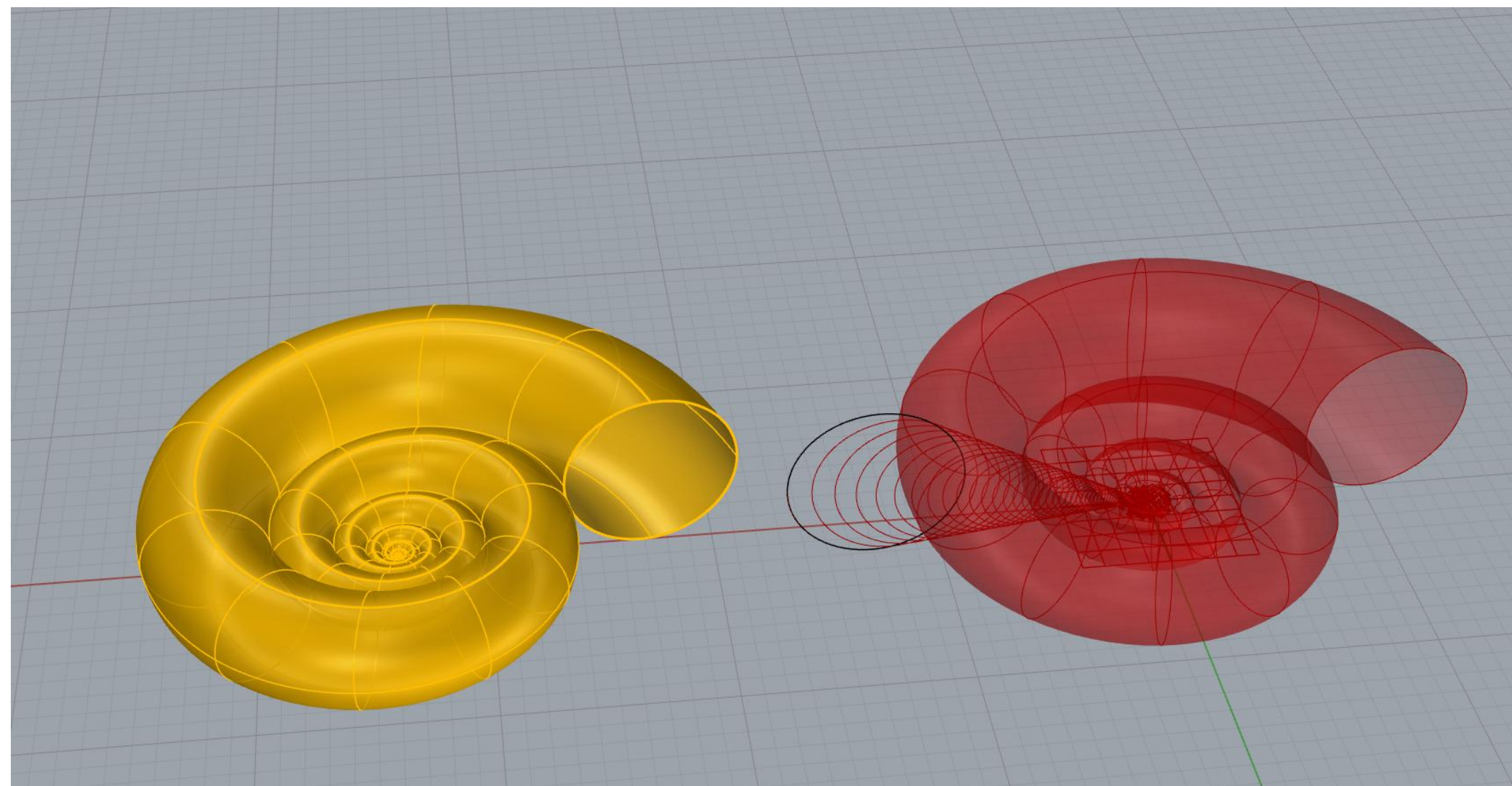


Loft  
Camada exterior

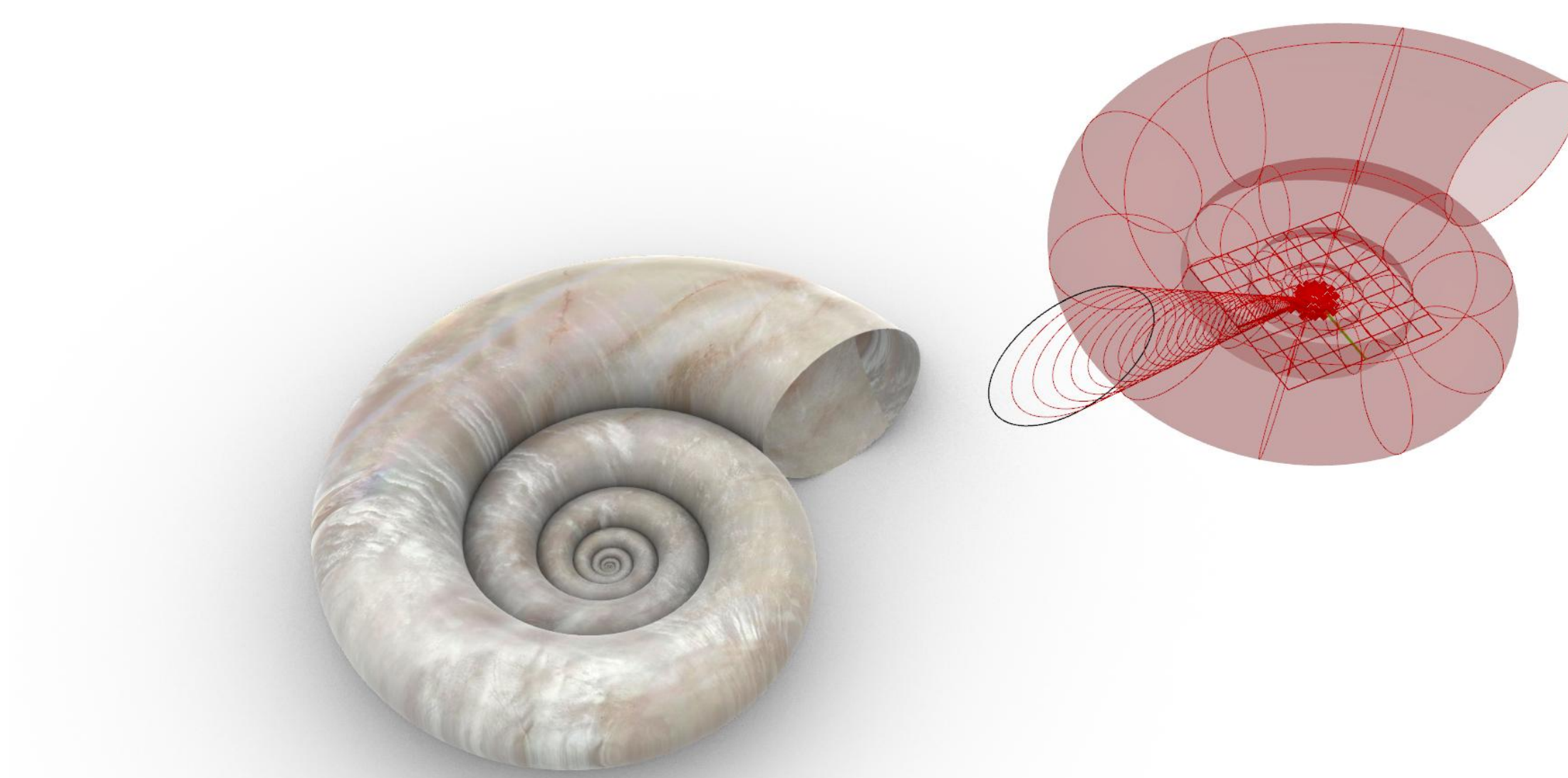
Sequência final do GrassHopper:



Perspetiva em Ghosted:

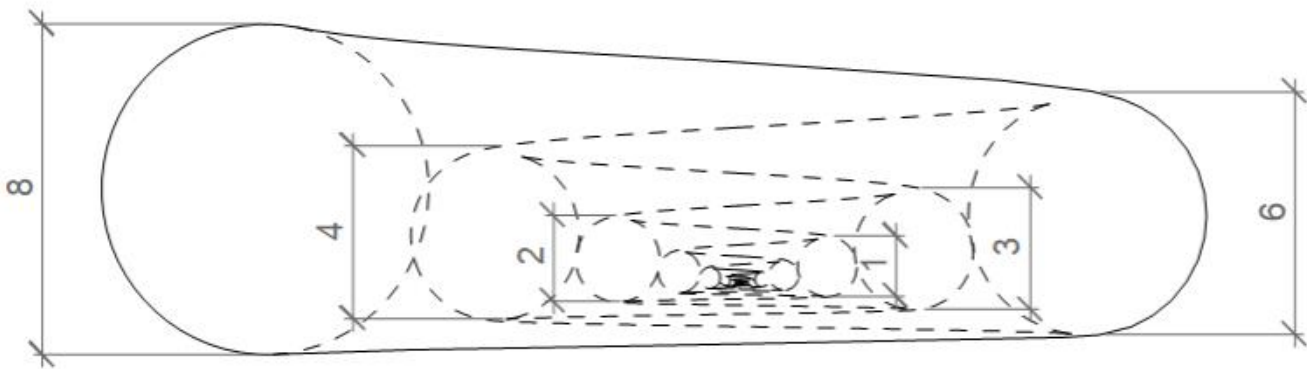


Perspetiva em Rendered:



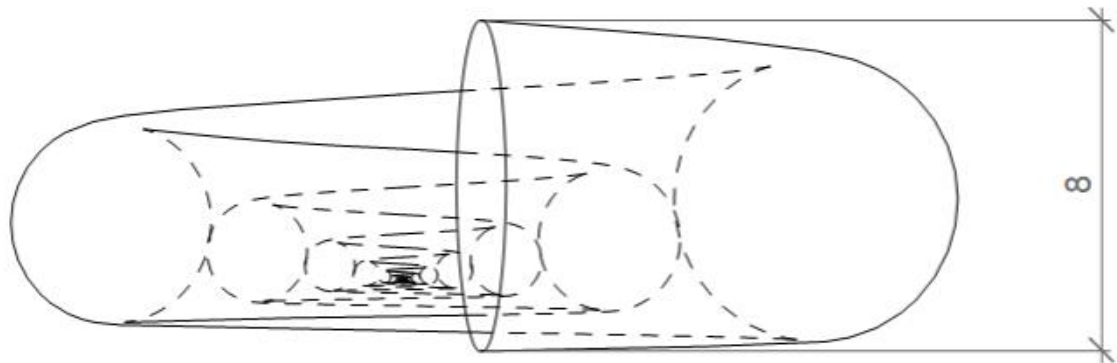
Grasshopper

Trabalho 3- Planorbis



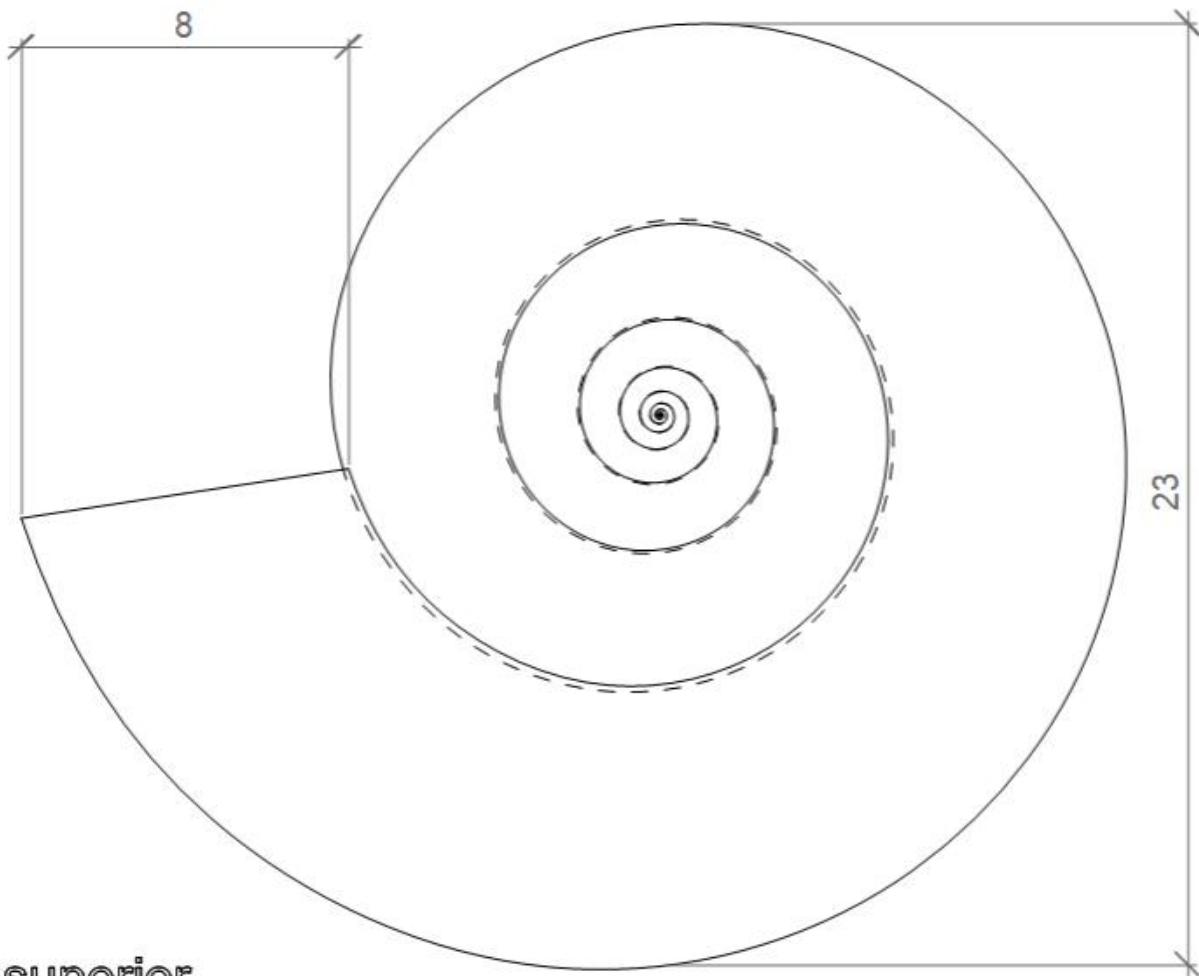
Vista frontal

1:2



Vista lateral direita

1:2



Vista superior

1:2

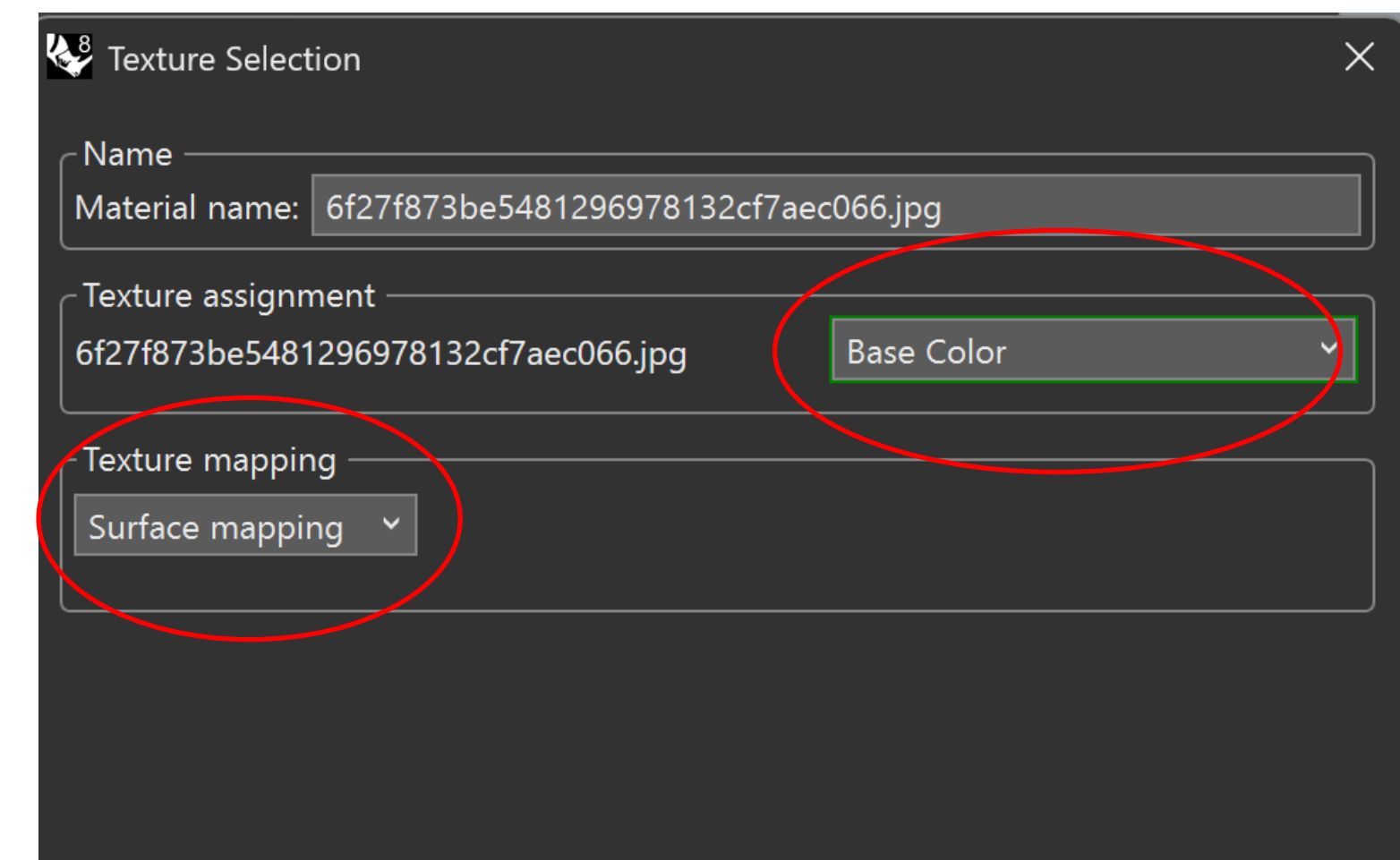
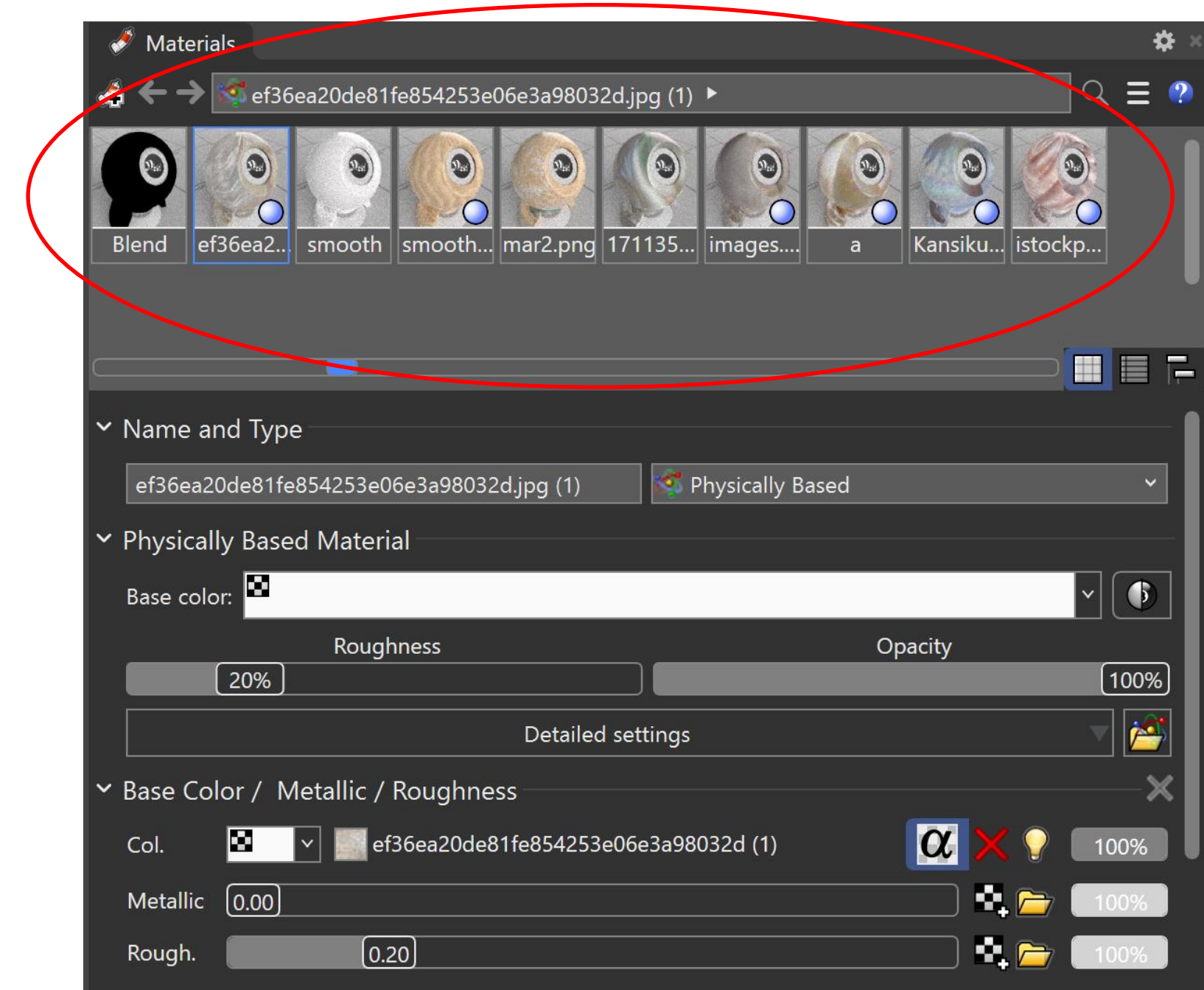
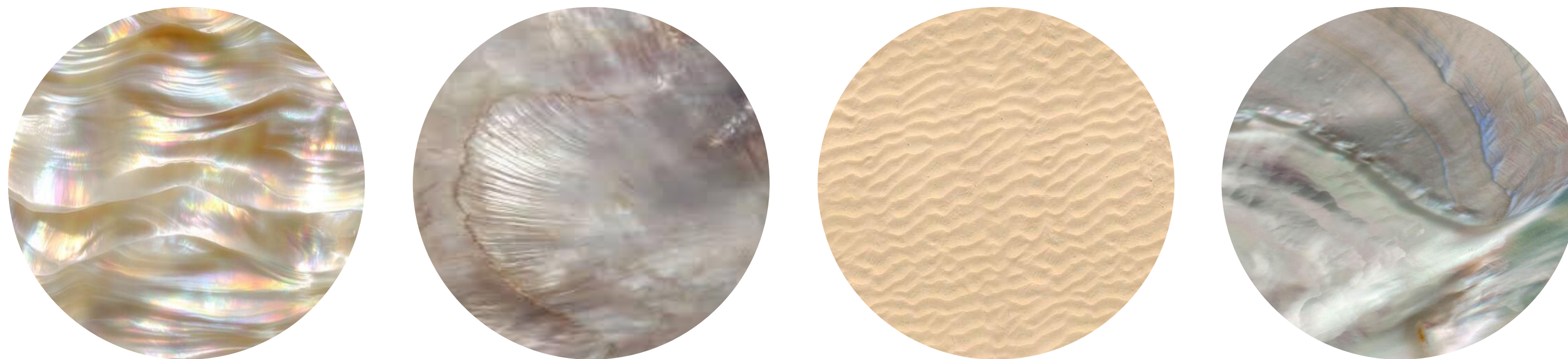


Perspetiva

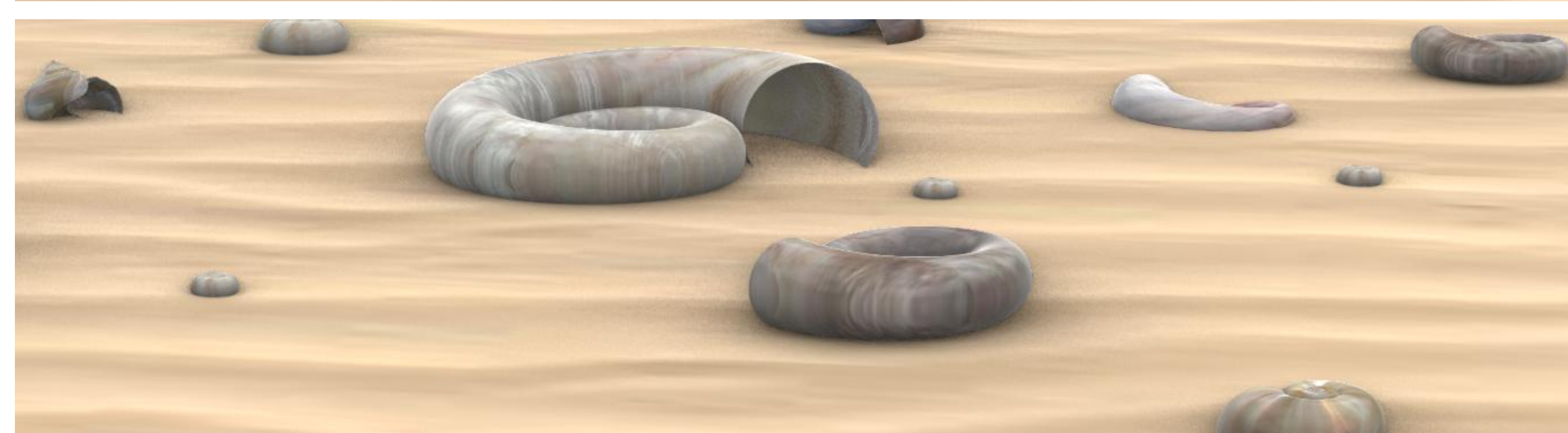
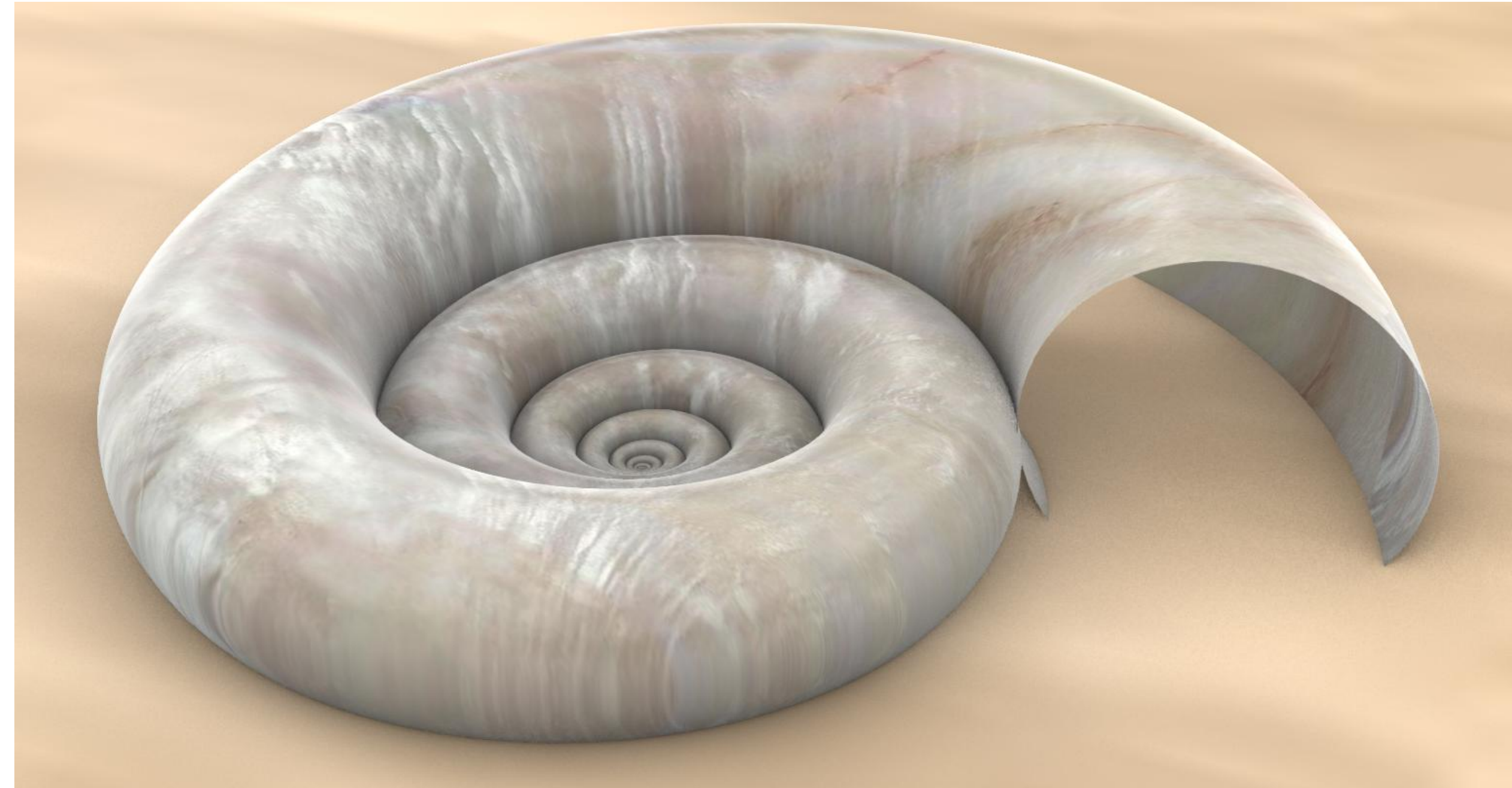
Para criar um cenário mais realista e perceptível tirei partido da secção rendering da barra do menu. Para criar os renderings da Planorbis, escreve-se materials do Rhino. Através da opção Materials > Create a Physically Based Material from Texture Files, foi selecionada uma imagem e aplicada como Base Color. É necessário escolher Surface Mapping, para que ocupasse toda a superfície de forma contínua, evitando a repetição da foto em quadrados ao longo do objeto.

A cena escolhida foi uma praia com várias conchas realizadas ao longo das últimas aulas.

### Exemplos de texturas:



Renderings do Planorbis juntamente com as diversas conchas realizadas em aula:



Nota: Para se poder visualizar as texturas é necessário alterar a perspectiva de ghosted para rendered.

**Grasshopper**

**Trabalho 3- Planorbis**