Exercício de implementação 2 | ARRAYS, EXPRESSÕES E DIAGRAMA DE VORONOI

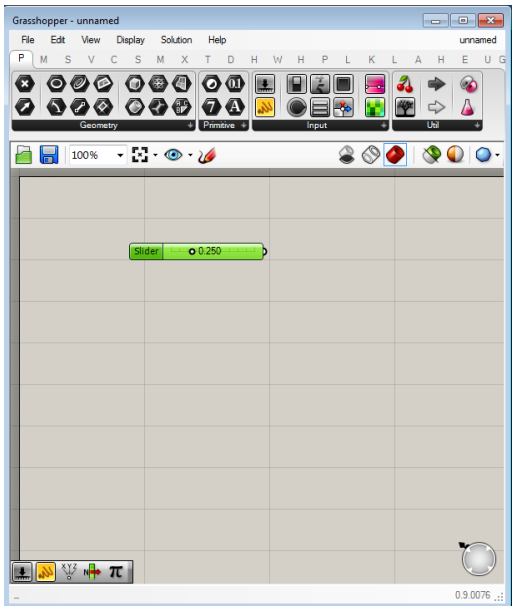
Este exercício de implementação foi traduzido e adaptado tendo como base o tutorial *F.2.2. Mathematics, Expressions & Conditionals*, Disponível na referência AKOS, G.; PARSONS, R.; Foundations - Grasshopper Primer. 2014. 3rd ed.. Explicações e comentários adicionais também se baseiam em outras fontes que estão em links ao longo do texto no item 6.1. Sugestão de leitura.

Este exercício de implementação tem o objetido de produzir um padrão bidimensional. Esta implementação será dividida em duas partes: na primeira parte exploraremos a produção de estruturas em série (Arrays) junto de funções trigonométricas e, na segunda parte, empregaremos componentes para a produção de diagramas de Voronoi que irão compor o padrão.

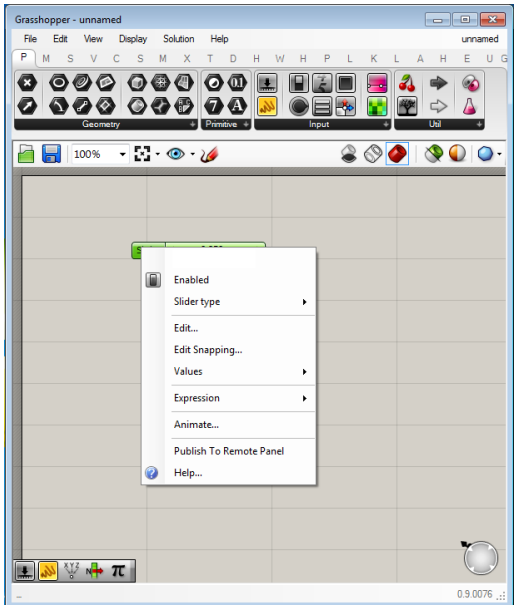
Diagramas de Voronoi são encontrados na natureza em diferentes escalas e sua produção pode ser empregada na concepção de objetos arquitetônicos. Exemplos de projetos que podemos modelar empregando diagramas de Voronoi são a fachada da Airspace House (Japão, https://www.arch2o.com/airspace-tokyo-faulders-studio/) e a cobertura da WestendGate (Alemanha, https://www.archdaily.com/175519/westendgate-just-burgeff-architekten-a3lab). Para saber mais sobre Diagramas de Voronoi e seu uso na arquitetura, sugerimos consultar BURRY e BURRY (2012) que discute a definição em seu glossário.

Exercício de Implementação

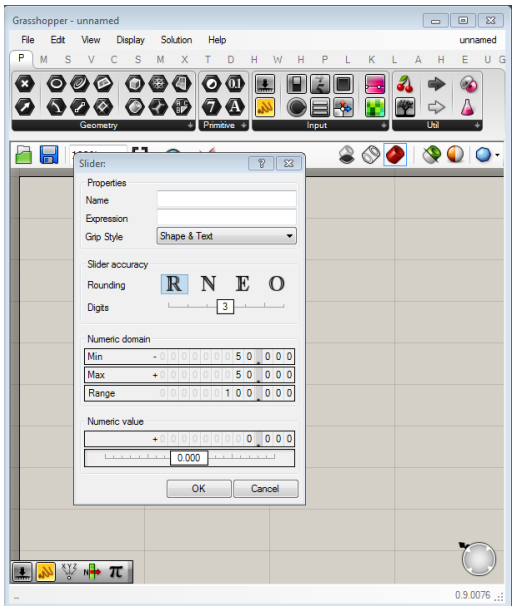
1. Crie um ponto. Este ponto será a geometria inicial para a construção do padrão. O ponto será criado pelo componente Construct Point e poderá ter suas coordenadas alteradas por componentes Number Slider.
   1.1. Adicione um Number Slider em: Params > Input > Number Slider



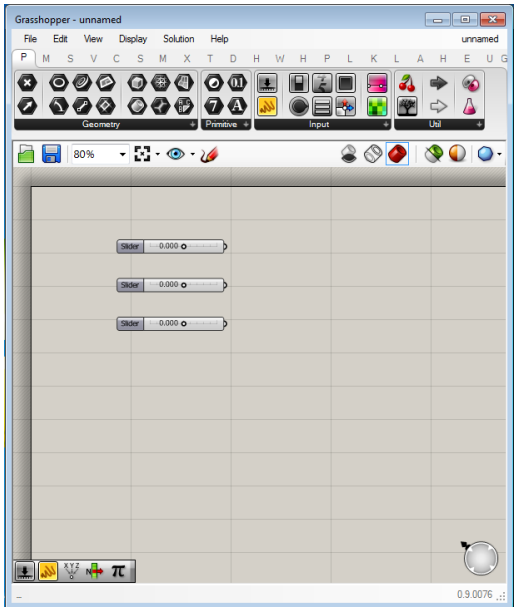
1.2. Clique com o botão direito sobre *Slider* e ,no menu que irá abrir, clique sobre a opção **Edit...**.



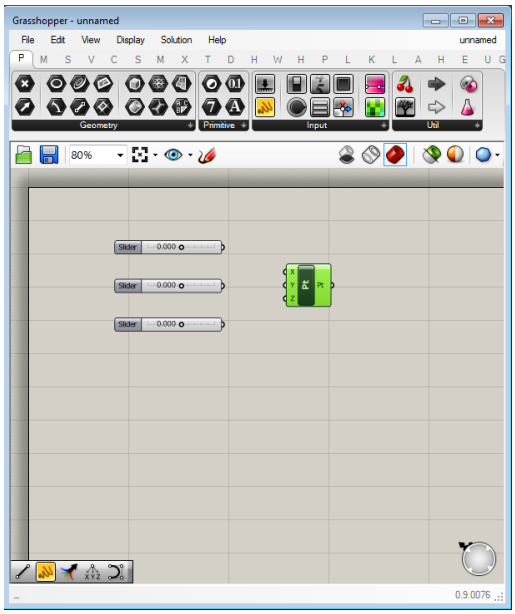
1.3. Configure o componente Number Slider para números reais (Floating Point Numbers) variando de -50 até 50 e clique em **OK**. O valor numérico deste componente Number Slider poderá ser alterado posteriormente para mudar as coordenadas do ponto, por enquanto podemos defini-lo como 0 (o ponto derá criado na origem do sistema de coordenadas)

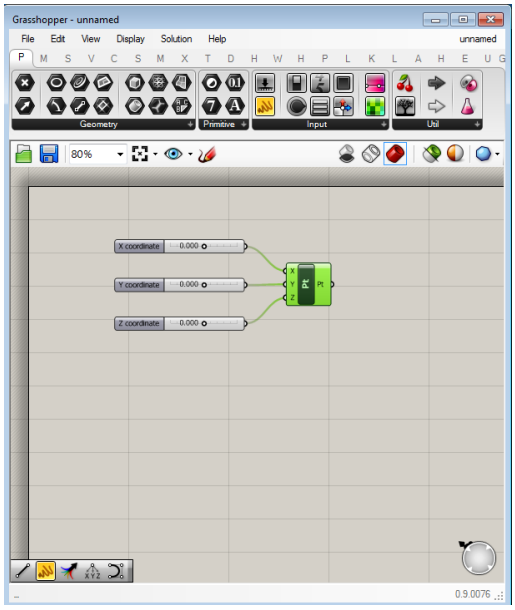


1.4. Copie e cole o Number Slider duas vezes, você terá três componentes igualmente configurados.



1.5. Adicione um componente **Construct Point** e conecte cada um dos componentes **Number Slider** nos parâmetros de entrada referentes às coordenadas X, Y e Z do componente **Construct Point.**



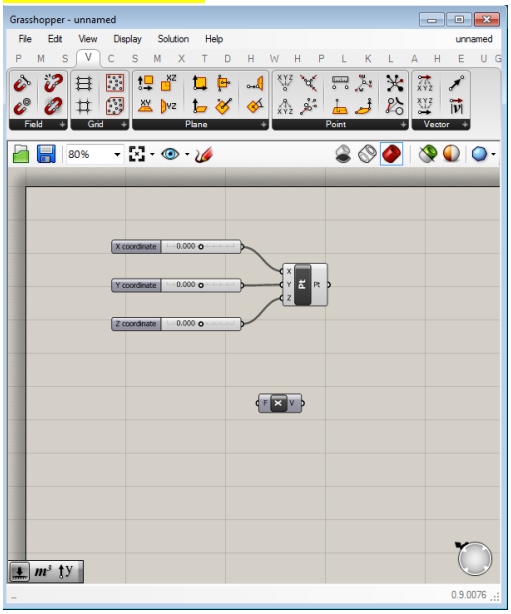


2. Agora iremos criar uma série de pontos para, posteriormente, fazer a sua interpolação. Para criar uma série, utilizaremos o componente **Linear Array** do Grasshopper. Para configurar este componente, precisaremos de três parâmetros de entrada: a **geometria que iremos replicar**, a **direção** e a **quantidade de elementos da série**.
    i. A **geometria** que iremos replicar: no nosso caso, essa geometria será o ponto criado mas, dependendo dos objetivos do algoritmo, podemos utilizar outras geometrias, como retas, planos, esferas, etc. No componente **Linear Array**, a geometria corresponde ao parâmetro de entrada **Geometry (G);**
    ii. A **direção** em que a geometria será replicada. Essa direção é dada por um vetor; no nosso caso, orientaremos que a série de pontos tenha a direção do eixo X, essa direção será fornecida pelo componente **Unit X**. No componente **Linear Array**, o parâmetro de entrada que corresponde à direção é o **Direction (D)**

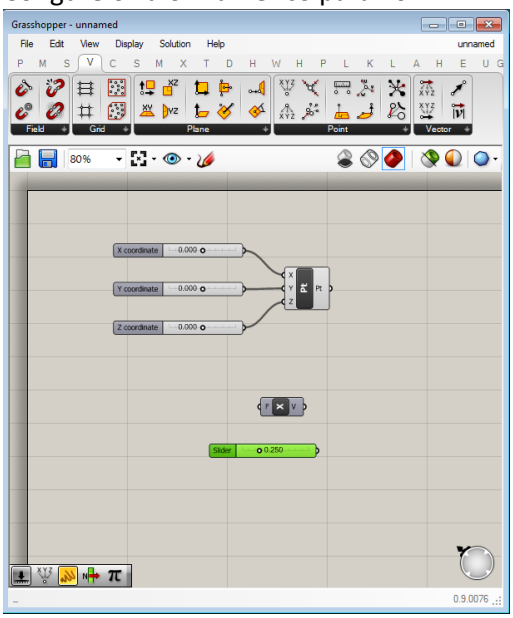
iii. A **quantidade de elementos da série**. Essa quantidade é dado por um número inteiro que, no nosso caso, será inserido por um componente Number Slider. No componente **Linear Array**, essa quantidade é dada pelo parâmetro de entrada **Count (N).**

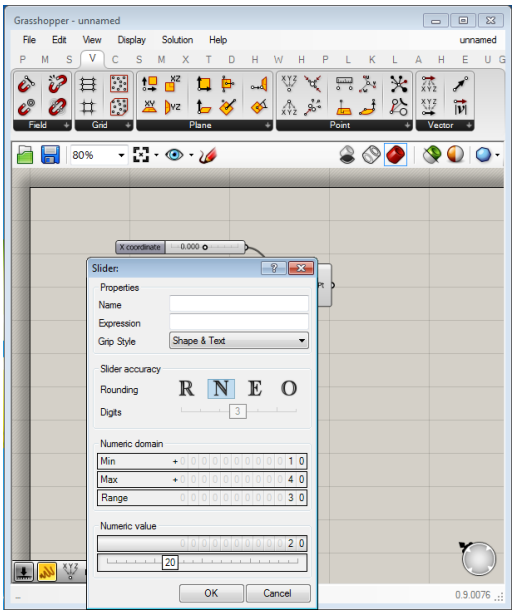
2.1. Agora iremos adicionar os componentes referentes aos parâmetros de entrada do componente **Linear Array** e configurá-los:

a. **Vetor Unitário no sentido X:** Adicione um vetor unitário pelo caminho: Vector > Vector > Unit X

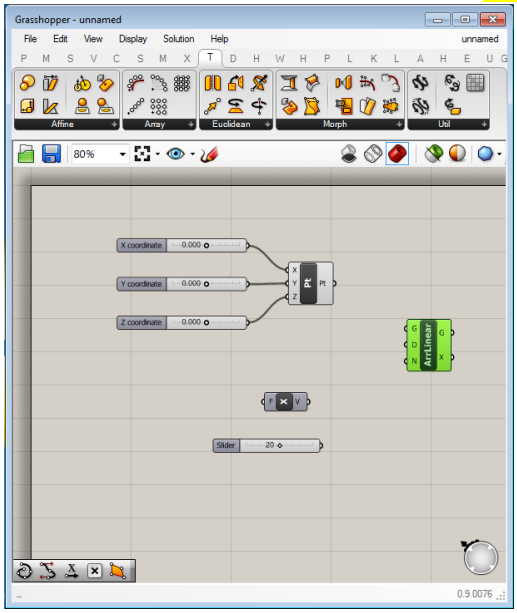


b. **Componente Number Slider**: Adicione um componente **Number Slider** e, em seguida, configure-o para valores inteiros (Integer Numbers) entre 10 e 40. Cofigure o valor numérico para 20.

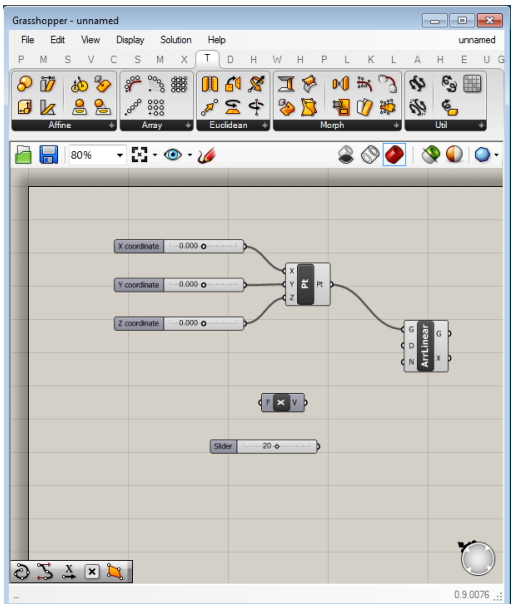




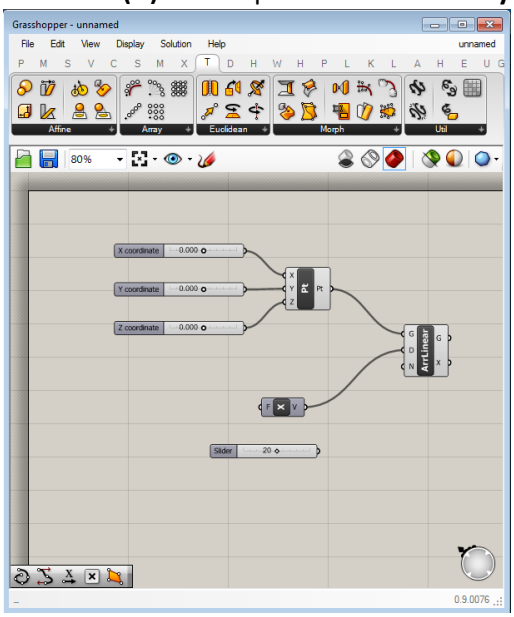
2.2. Adicione um componente **Linear Array**: Transform> Array > Linear Array



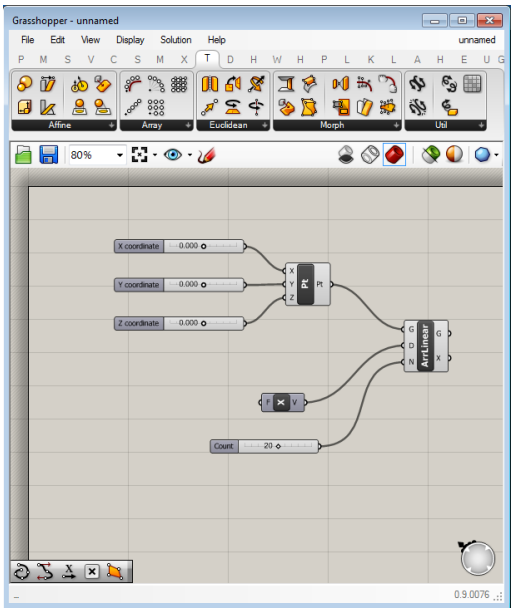
2.3. Conecte o parâmetro de saída do componente **Construct point** no parâmetro de entrada **Geometry (G)** do componente **Linear Array**.



2.4. Conecte a saída do componente do **vetor unitário X** ao parâmetro de entrada **Direction (D)** do componente **Linear Array**.



2.5. Conecte o último componente **Number Slider** (item 2.1.b) configurado ao parâmetro de saída **Count (N)** do componente **Linear Array**.

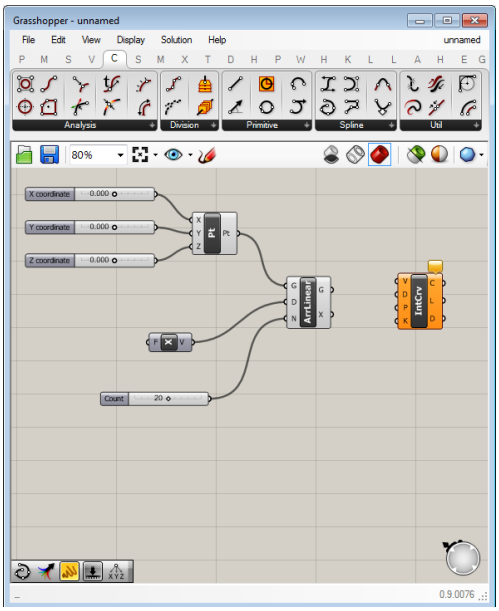


Exploração: Na visualização do programa Rhinoceros, você deve estar vendo uma série de pontos. Se você alterar os **Number Sliders** referentes às coordenadas do ponto criado, verá que o ponto inicial se desloca e, consequentemente, todos os demais pontos da série também se deslocam mantendo as respectivas distâncias. Por outro lado, se alterar o slider Count, mudará a quantidade de elementos da série.

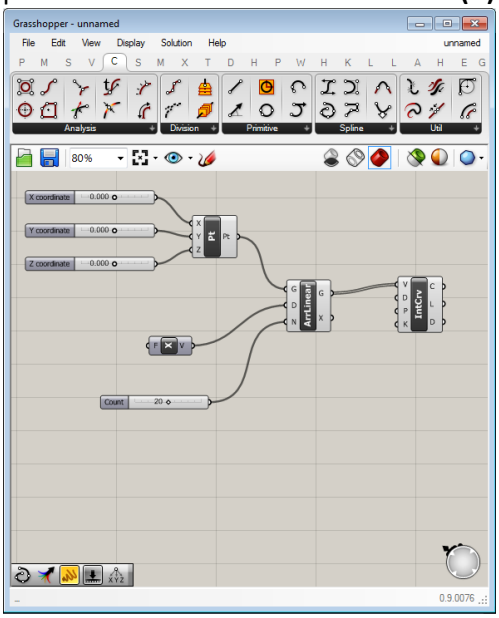
2.6. Agora, iremos realizar a interpolação dos pontos utilizando o componente **Interpolate.** Inicialmente, para configurar este componente iremos precisar apenas dos pontos que serão interpolados. No nosso caso, os pontos serão dados pela série que criamos utilizando o componente **Linear Array**. A saída do componente **Linear Array** fornecerá os valores de entrada para o parâmetro **Vertices as List (V)** do componente **Interpolate**.

Note: o componente Interpolate permite também a configuração de outros parâmetros de entrada que não iremos utilizar neste exercício. Caso queira saber mais sobre os demais parâmetros do componente **Interpolate**, você pode acessá-lo na página de documentação do Grasshopper, pelo link: https://grasshopperdocs.com/components/grasshoppercurve/interpolate.html

a. Adicione o componente **Interpolate** pelo caminho: Curve > Spline> Interpolate



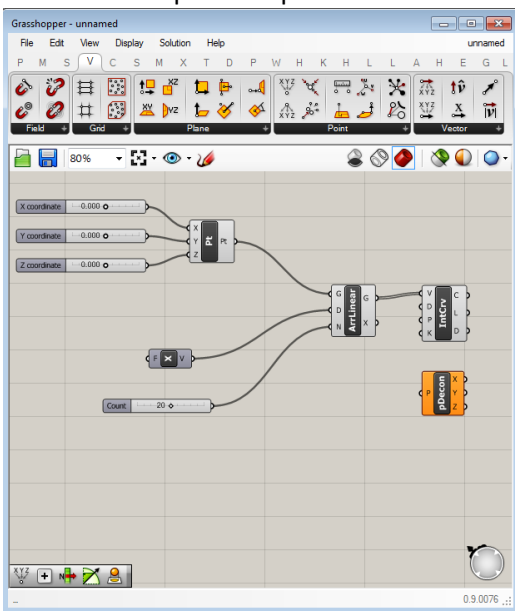
b. Conecte o parâmetro de saída **Geometry (G)** do componente **Linear Array** ao parâmetro de entrada **Vertices as List (V)** do componente **Interpolate Curve**



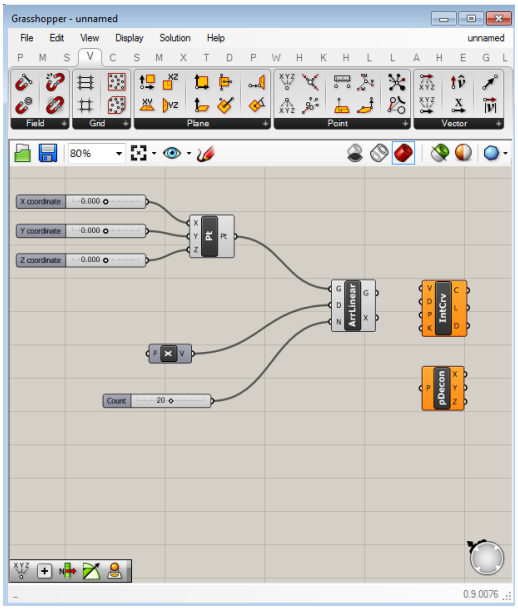
Exploração: Nesta etapa, criou-se uma reta conectando a série de pontos gerados pelo componente **Linear Array**. Agora, ao alterar os valores dos componentes **Number Slider**, a reta gerada acompanhará as alterações de maneira coordenada.

3. Nesta etapa iremos modificar o algoritmo para que ele gere, ao invés de uma reta, curvas baseadas nas funções trigonométricas seno e cosseno. Partiremos do algoritmo já definido, porém precisaremos calcular novas coordenadas X e Y para cada um dos pontos gerados na série, para isso utilizaremos os componentes **Sine** e **Cosine**. Cada um dos pontos gerados inicialmente na série será decomposto em coordenadas pelo componente **Deconstruct** e, após calculadas as novas coordenadas, determinarão novos pontos através do componente **Construct Point**.

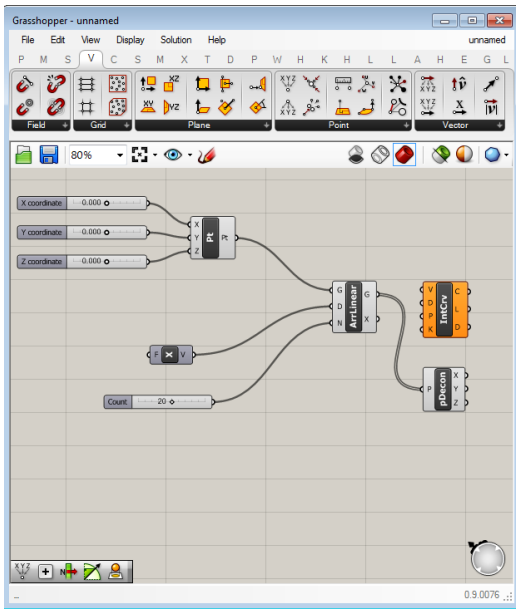
3.1. Primeiro, iremos decompor em coordenadas os pontos gerados na série, usaremos o componente **Deconstruct**, adicione-o pelo caminho: Vector > Point > Deconstruct. Com este componente podemos obter cada uma das coordenadas de cada ponto.



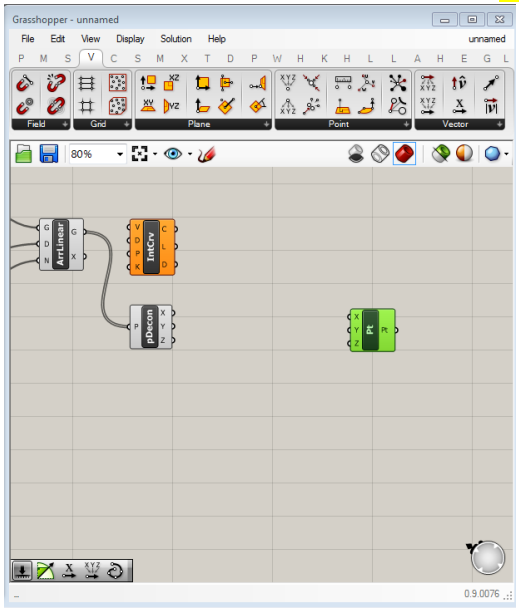
3.2. Desconecte o parâmetro de entrada **Vertices as List (V)** do componente **Interpolate Curve**



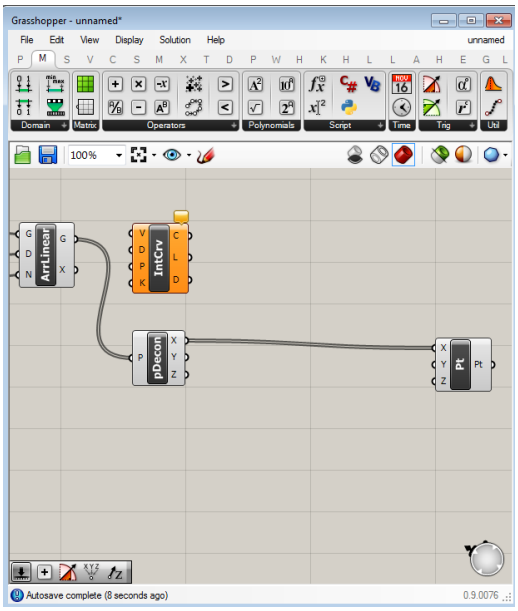
3.3. Conecte o parâmetro de saída **Geometry (G)** do componente **Linear Array** ao parâmetro de entrada **Point (P)** do componente **Deconstruct**



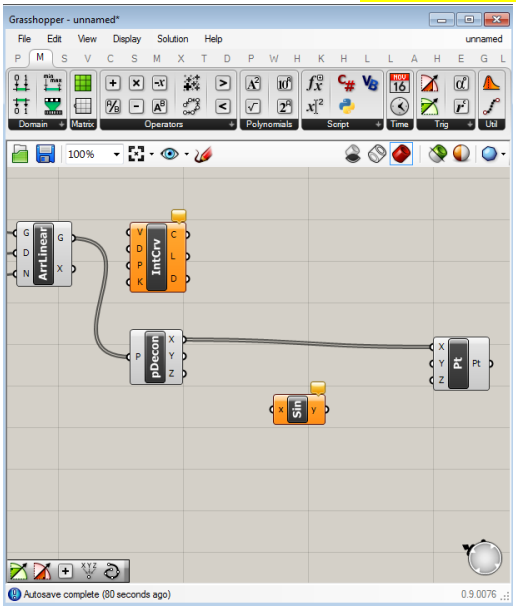
3.4. Adicione o componente **Construct Point**: Vector > Point > Construct Point



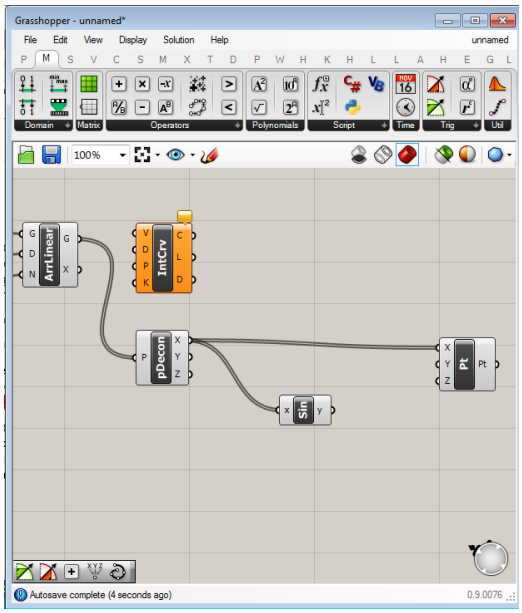
3.5. Conecte o parâmetro de saída **X Component (X)** do componante **Deconstruct** ao parâmetro de entrada **X Coordinate (X)** do componente **Construct Point**



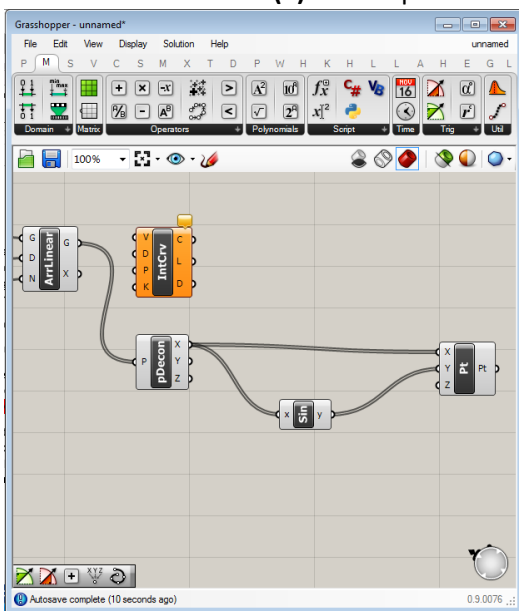
3.6. Adicione o componente **Sine**: Maths > Trig > Sine



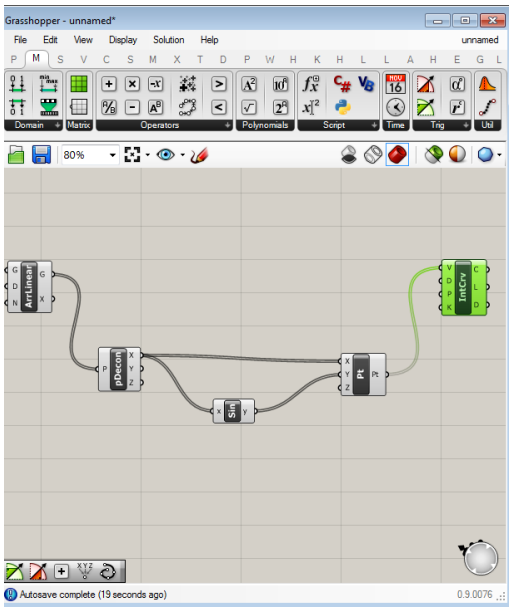
3.7. Faça uma segunda conexão do parâmetro de saída **X Component (X)** do componente **Deconstruct** ao parâmetro de entrada **Value (x)** do componente **Sine**.



3.8. Conecte o parâmetro de saída **Result (y)** do componente **Sine** ao parâmetro de entrada **Y coordinate (Y)** do componente **Construct Point**.

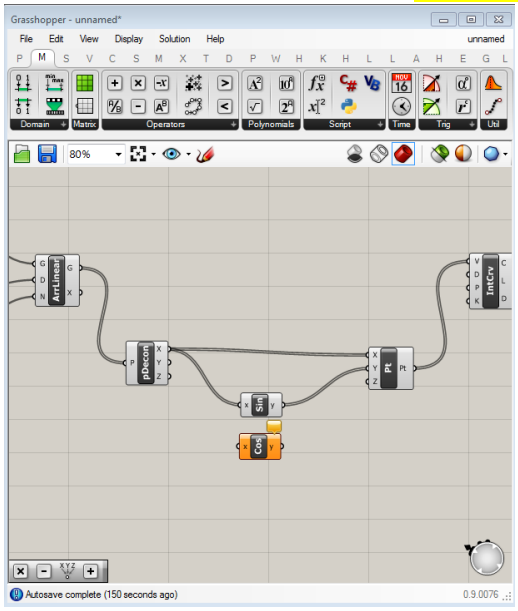


3.9. Conecte o parâmetro de saída **Point (Pt)** do componente **Construct Point** ao parâmetro de entrada **Vertices (V)** do componente **Interpolate**.

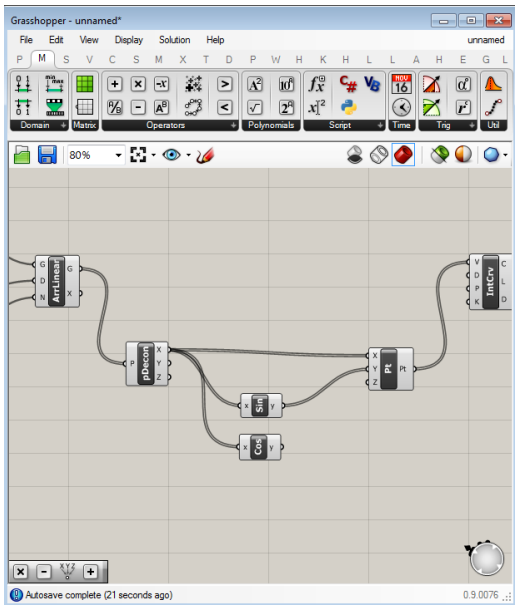


Nesta etapa, a reta inicial deve ter se tornado uma curva senoidal ao longo do eixo X na visualização do Rhinoceros.

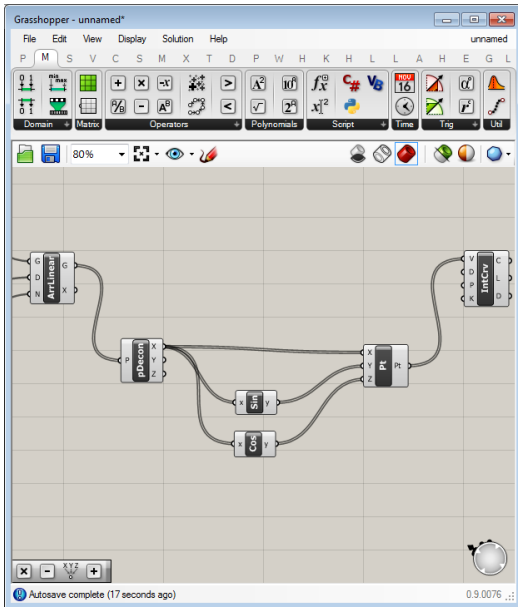
4. Agora faremos um procedimento análogo para produzir alterações na coordenada Z usando um componente da função cosseno.
   4.1. Adicione um componente **Cosine**: Maths > Trig > Cosine



   4.2. Faça uma terceira conexão do parâmetro de saída **X component (X)** do componente **Deconstruct** ao parâmetro de entrada **Value (x)** do componente **Cosine**



4.3. Conecte o parâmetro de saída **Result (y)** do componente **Cosine** ao parâmetro de entrada **Z coordinate (Z)** do componente **Construct Point**

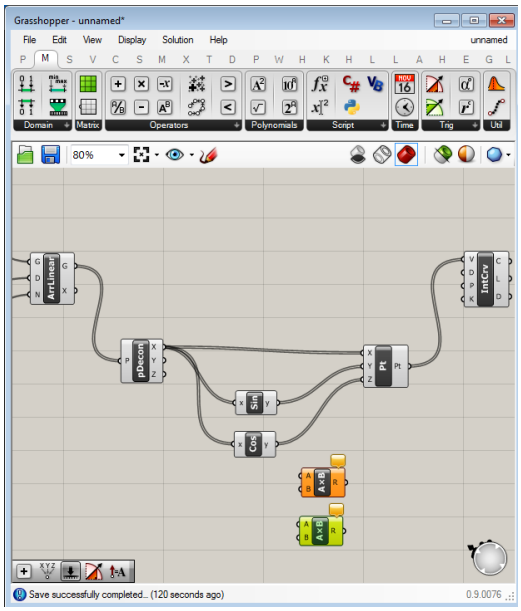


Alterando a coordenada Z de cada ponto da série, nós criamos uma curva espacial em hélice. Para visualizar esta nova curva, orbite a visualização em perspectiva no Rhinoceros.

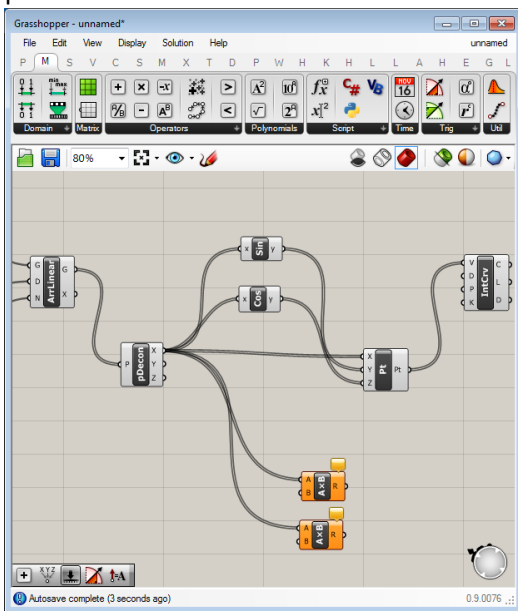
Exploração: Uma exploração possível é alterar a ordem das conexões dos componentes **Sine** e **Cosine** no componente **Construct Point**. O Grasshooper também oferece o componente **Tan**, com a função tangente.

Agora iremos transformar a curva criada no item anterior em uma curva espiral.

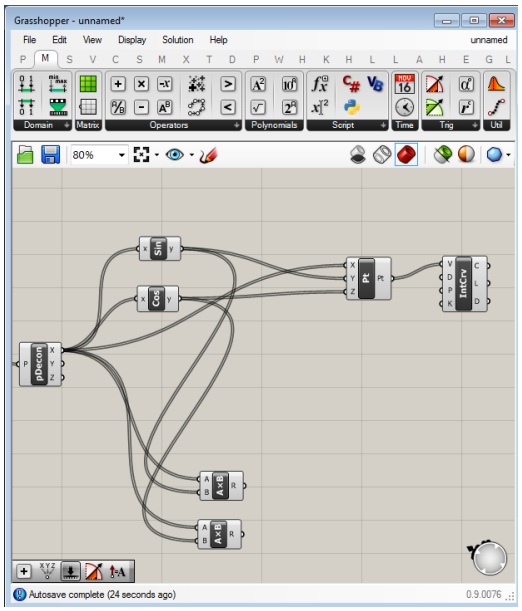
4.4. Adicione dois componentes de Multiplicação: Maths > Operators > Multiplication (você pode duplicar o componente copiando [Ctrl + C] e colando [Ctrl + V] ou realizando novamente o caminho destacado)



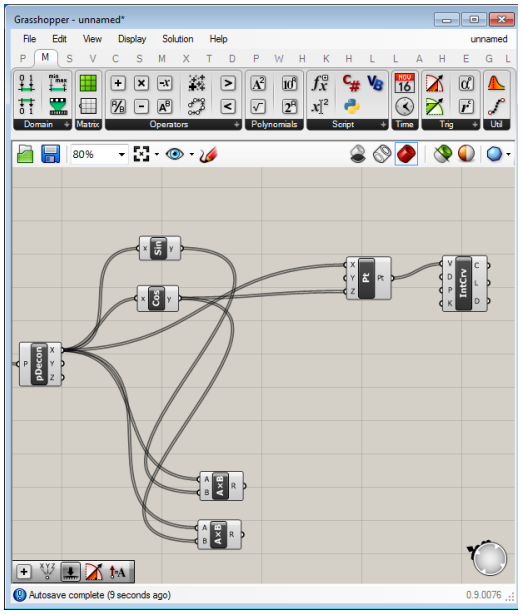
4.5. Conecte o parâmetro de saída **X component (X)** do componente **Deconstruct** ao parâmetro de entrada **A** de cada um dos componentes de multiplicação



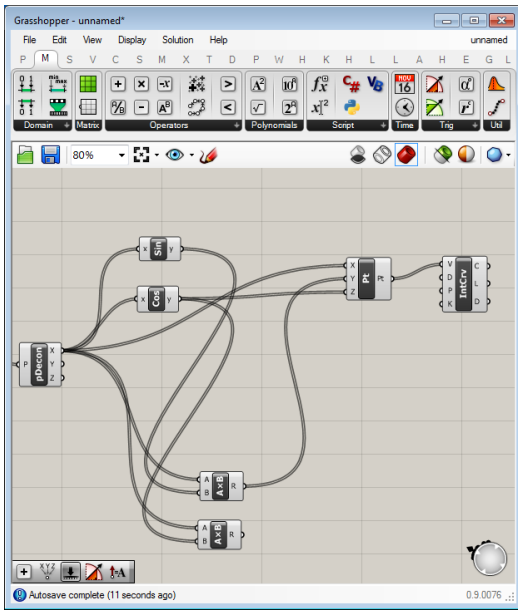
4.6. Conecte o parâmetro de saída **Result (y)** do componente **Sine** ao parâmetro de entrada **B** de um dos componentes **Multiplication**. Em seguida, conecte o parâmetro de saída **Result (y)** do componente **Cosine** ao parâmetro de entrada **B** do outro componente **Multiplication**



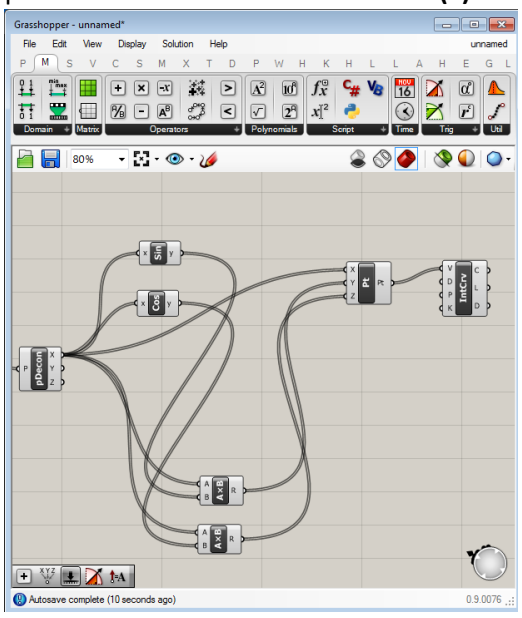
4.7. Desconecte o parâmetro de entrada **Y Coordinate (Y)** do componente **Construct Point**.



4.8. Conecte o parâmetro de saída **Result (R)** do primeiro componente **Multiplication** ao parâmetro de entrada **X Coordinate (X)** do componente **Construct Point**



4.9. Conecte o parâmetro de saída **Result (R)** do segundo componente **Multiplication** ao parâmetro de entrada **Z Coordinate (Z)** do componente **Construct Point**.



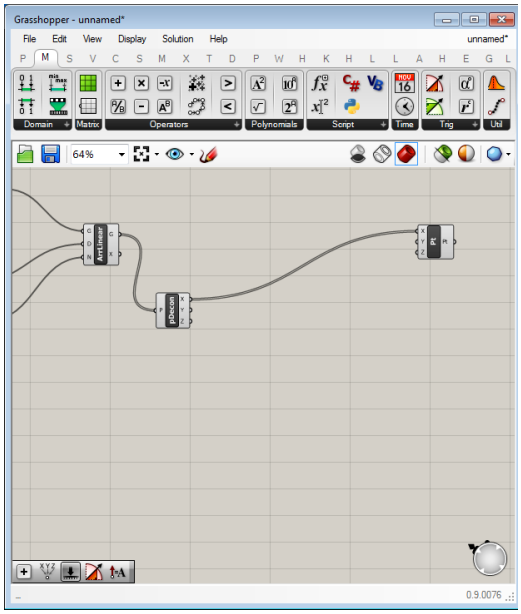
Agora verifique, orbitando a janela de visualização em perspectiva do Rhinoceros, a curva gerada.

5. Agora iremos explorar a produção de um padrão empregando um componente que nos permite definir expressões e um outro que constrói diagramas de Voronoi.

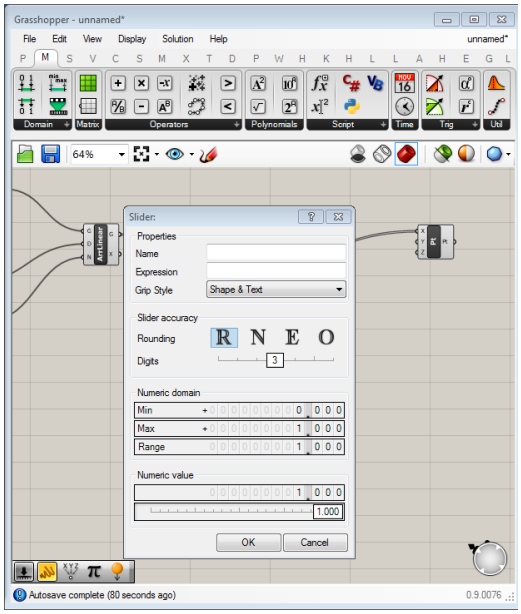
5.1. No algoritmo já produzido, delete os componentes **Sine**, **Cosine**, **Multiplication** e **Interpolation**.

Nota: Caso queira, você pode salvar o algortimo produzido antes deletar os componentes indicados, vá em File > Save Document. Na janela que irá abrir, escolha o local para salvar, nomeie o arquivo e clique em

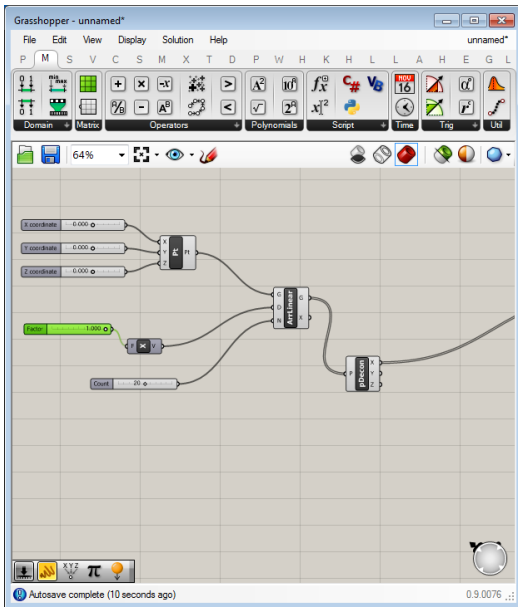
Salvar. O arquivo salvo é do tipo Grasshopper Binary e tem a extensão .gh



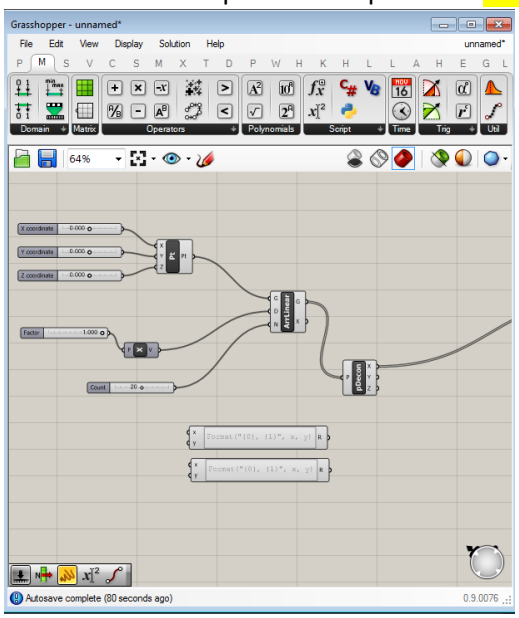
5.2. Adicione um componente **Number Slider** e o configure para valores reais (Floating Point Numbers) variando entre 0 e 1. Também defina o valor numérico como 1.



5.3. Conecte o componente **Number Slider** ao parâmetro de entrada **Factor (F)** do componente **Unit X** (o mesmo que foi adicionado no item 2.1.a deste tutorial).

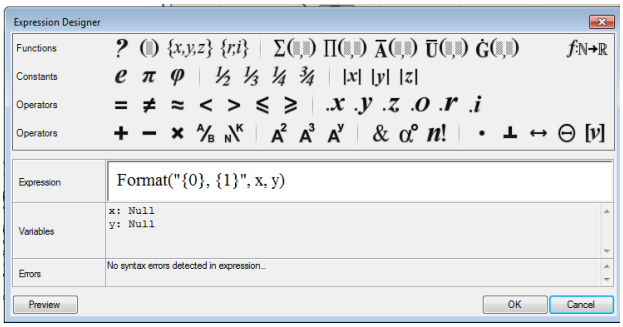


5.4. Adicione dois componentes Expression: Maths > Script > Expression

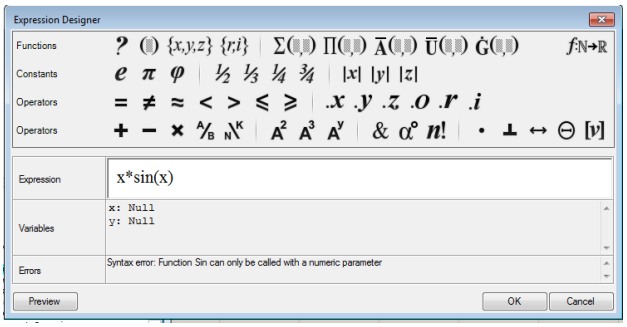


5.5. Agora iremos configurar cada um dos componentes **Expression**:

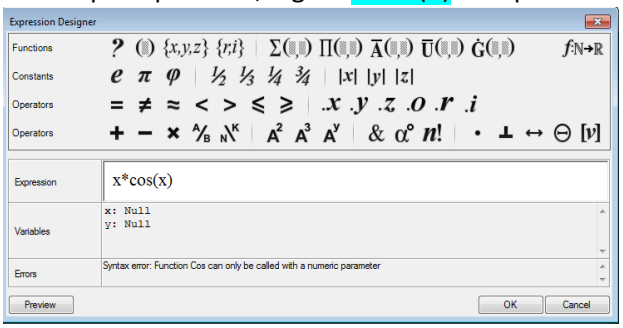
a. No primeiro componente **Expression**, dê um duplo clique para abrir o editor. Abrirá uma janela como esta:



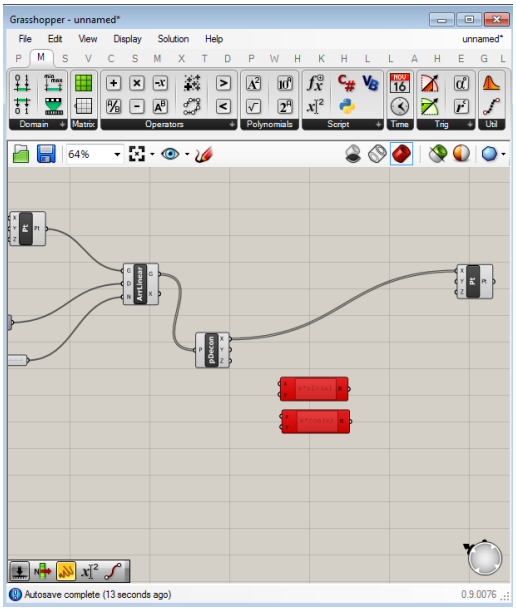
b. No campo Expression, digite: x*sin(X) e clique em **OK**



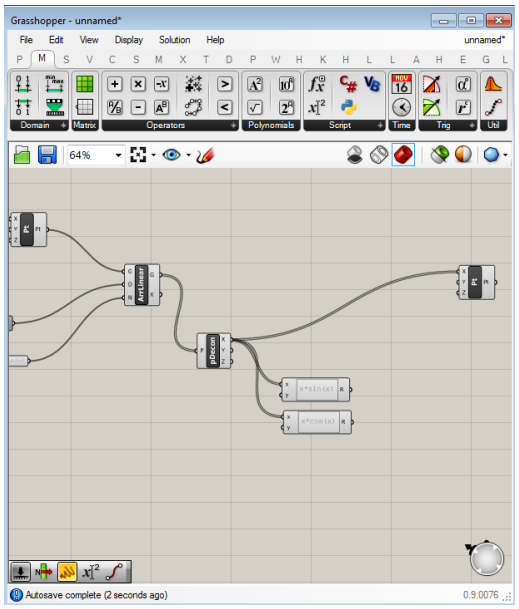
c. No segundo componente Expression, dê um duplo clique para abrir o editor e, no campo Expression, digite: x*cos(X) e clique em **OK**



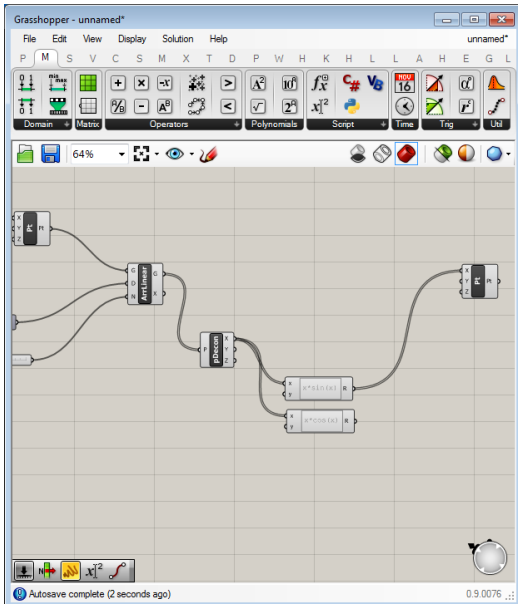
d. Após essas operações, os componentes **Expression** ficarão vermelhos, isso ocorre pois ainda não foi coletado nenhum valor de entrada para o parâmetro x da equação que definimos, faremos isso nos próximos itens.



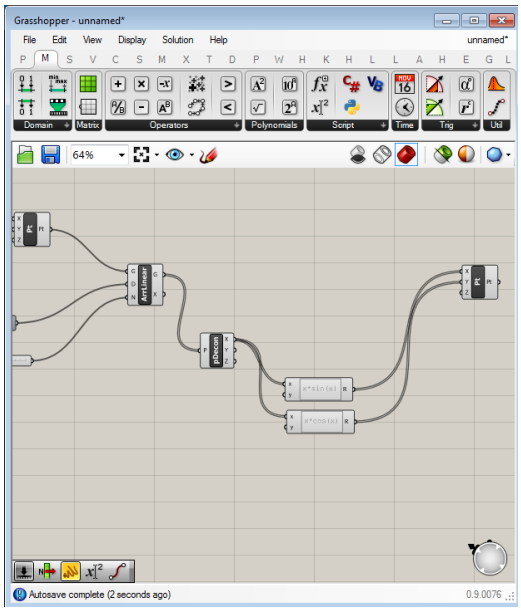
5.6. Conecte o parâmetro de saída **X component (X)** do componente **Deconstruct** ao parâmetro de entrada **Variable x (x)** de cada um dos componentes **Expression**.



5.7. Conecte o parâmetro de saída **Result (R)** do primeiro componente **Expression** ao parâmetro de entrada **X coordinate (X)** do componente **Construct Point**.

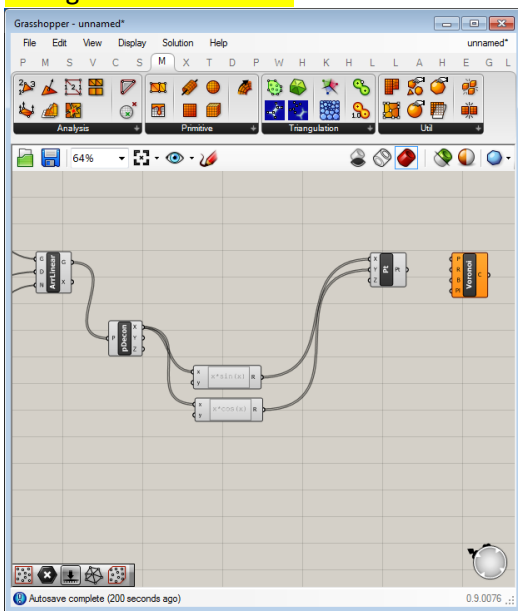


5.8. Conecte o parâmetro de saída **Result (R)** do segundo componente **Expression** ao parâmetro de entrada **Y coordinate (Y)** do componente **Construct Point**.

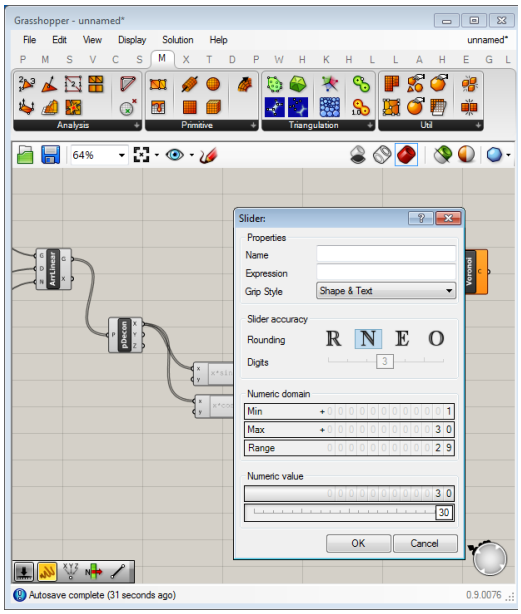


Verifique que essa última operação gerou um conjunto de pontos disposto de forma espiralada na visualização do Rhinoceros.

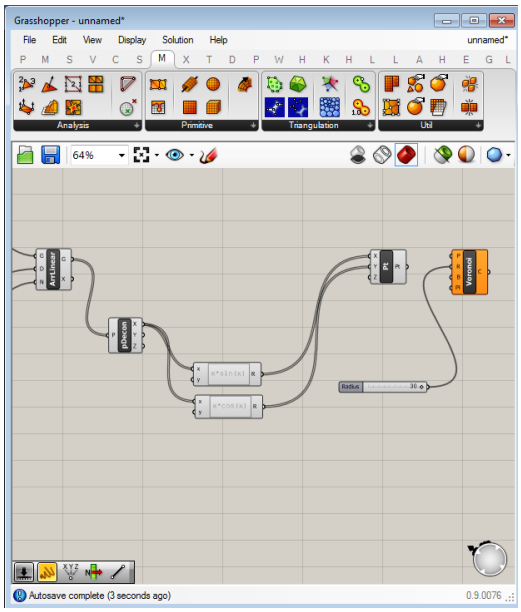
5.9. Agora iremos adicionar o componente de diagrama de **Voronoi**: Mesh > Triangulation > Voronoi.



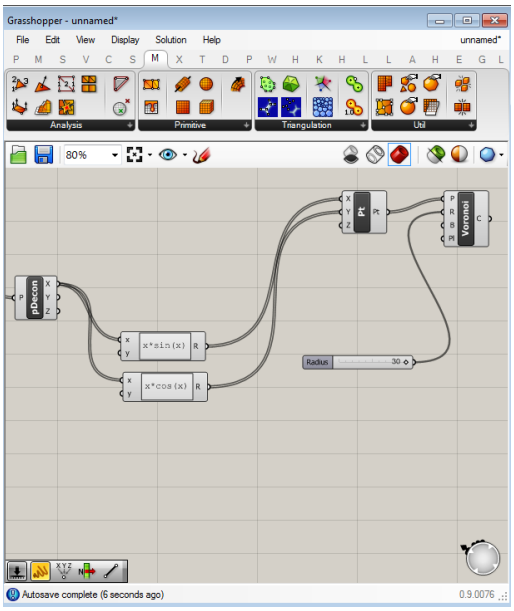
5.10. Adicione um componente **Number Slider** e o configure para valores inteiros (Integer numbers) variando entre 1 e 30. Depois, defina o valor numérico 30.



5.11. Conecte o **Number Slider** ao parâmetro de entrada **Radius (R)** do componente **Voronoi**.



5.12. Conecte o parâmetro de saída **Point (Pt)** do componente **Construct Point** ao parâmetro de entrada **Points (P)** do componente Voronoi.



Exploração: Agora você pode explorar diferentes valores para os parâmetros de cada um dos **Number Slider** do algoritmo. Também é possível usar diferentes funções no componente **Expression**, inclusive com mais variáveis que podem ter ser originadas de diferentes fontes.

6. Sugestão de Leitura

BURRY, J.; BURRY, M.. *The New Mathematics of Architecture*, 2012.