

Creating a Pavé Set Eternity Band



an exercise in Rhino 3D's
Grasshopper



by Christopher Mendola

Table of Contents

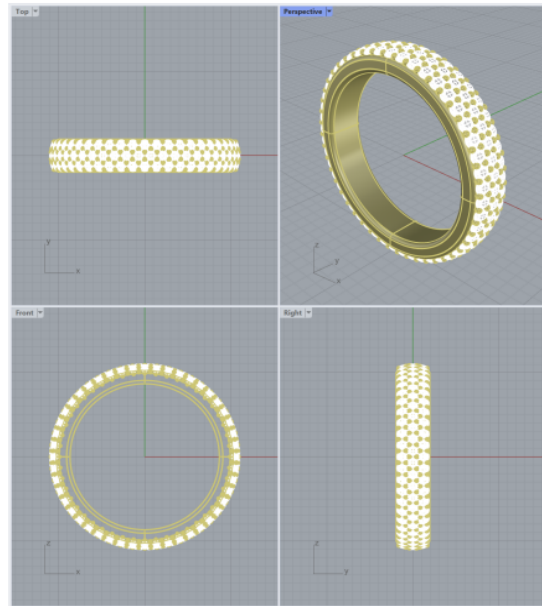
Introduction	pg 3
File Preparation	pg 4
The Ring Rail (Finger Size)	pg 6
The Ring Thickness	pg 10
Building the Ring's Profile	pg 11
The Stone Layout	pg 20
Arranging the Stones	pg 30
Creating the Prongs	pg 34
Adjusting the Parameters	pg 42
Creating the Gem Cutters (<i>for Rendering</i>)	pg 45
Creating Pilot Holes (<i>for Production</i>)	pg 47
Creating Prongs (<i>for Production</i>)	pg 48
Baking the Ring	pg 49
Rendering the Ring	pg 53

Introduction

Welcome! This tutorial introduces *parametric* modeling using Rhino 3D and its popular Grasshopper plug-in. Learning Grasshopper should allow you to model more efficiently.

Although the subject matter is jewelry, this exercise is about learning to use several of the Grasshopper components that can be applied to modeling in general. For the *jewelry designer*, parametric modeling can be a real time saver as certain measurements (ie finger size, thickness, width, amount of stones, etc.) and aesthetic components need to be *adjusted* often.

Moving forward I will refer to Grasshopper as **GH**.

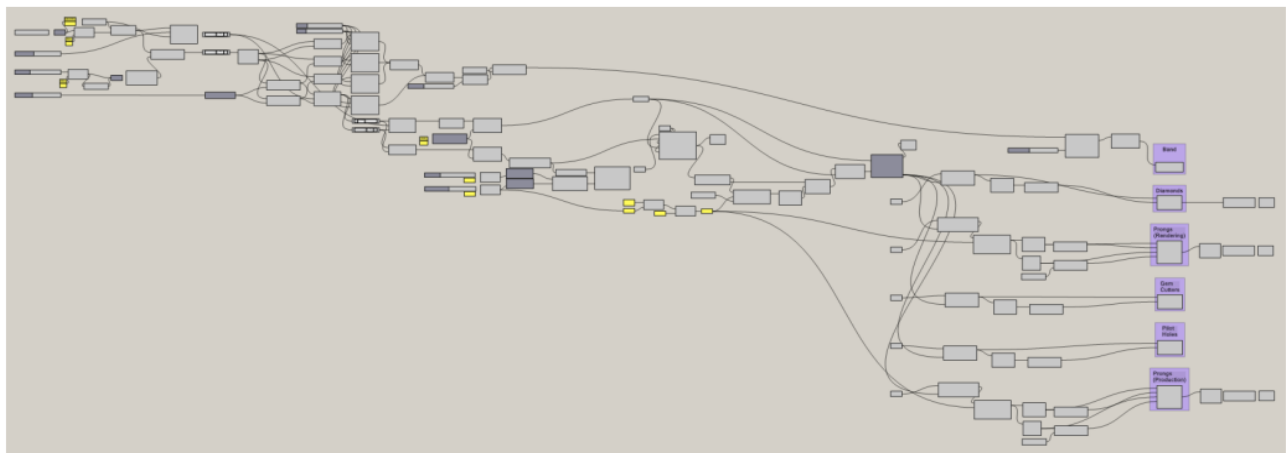


It is not necessary to know **GH** to follow along. Although I will only briefly introduce each of the many *components* we will use throughout. This *exercise* is meant to spark interest in those that are intimidated to open **GH**. It is by no means comprehensive.

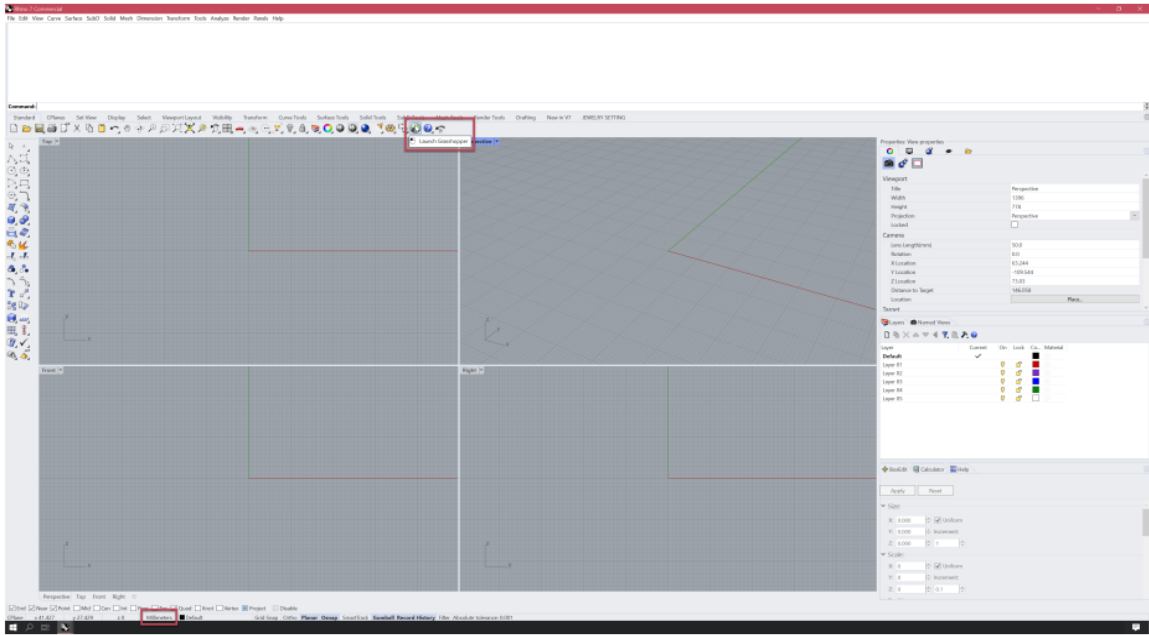
My hope is that those who have never even opened **GH** will be able to follow along and see its potential. I want to *motivate* and *encourage* further learning.

Below is a snapshot of the **GH** canvas we will be creating. It may seem overwhelming at this perspective but each step is very manageable and very similar in approach to building it entirely in Rhino 3D. As one learns **GH** they should see that it is an amazing plug-in that works hand in hand with Rhino and will save *time* in the long run.

Let's begin...

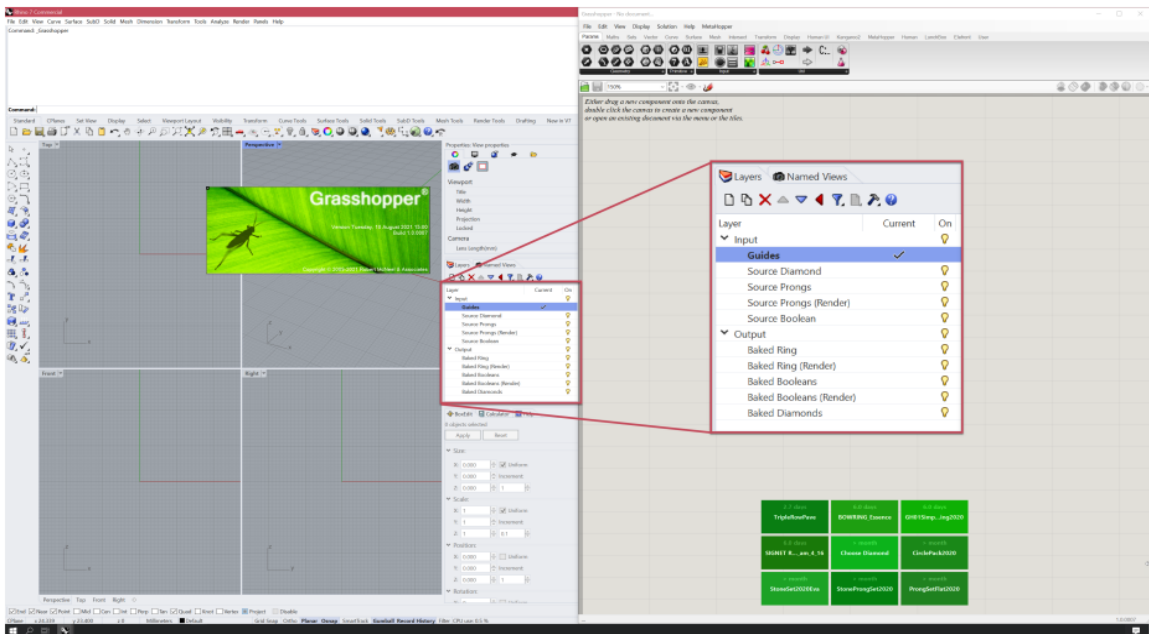


File Preparation



1

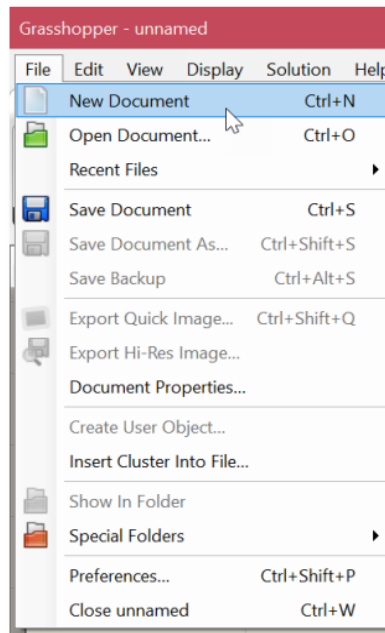
Start by opening a new file in Rhino using the **Small Objects - Millimeters** template. Then Launch **GH**.



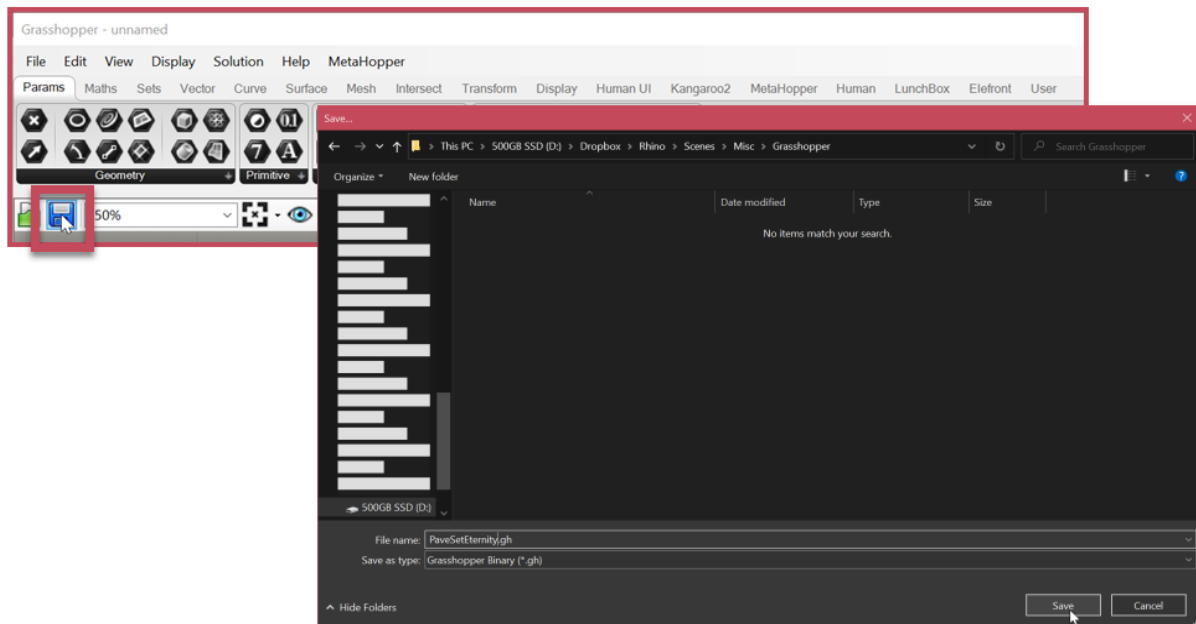
2

I like to set up my **GH Canvas** to the right half and adjust the Rhino **Viewports** and **Tool Panels** accordingly. I set up my **Layers** as above. However I suggest you set them as you prefer. We will eventually create some geometry in Rhino that we will use in **GH**. Then we will use **GH** to **Bake** the altered geometry back into Rhino.

File Preparation

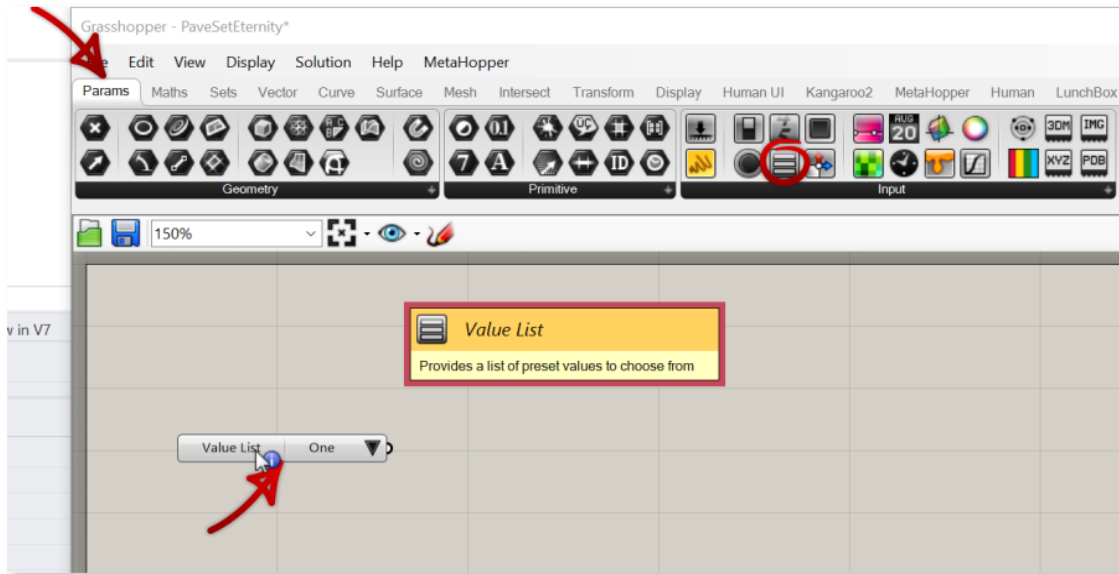


3 Create a New Document in GH via its Main menu bar.



4 Save both your *.3dm* (Rhino) and your *.gh* (Grasshopper) files separately. Those new to GH may be unfamiliar that you are creating a separate file that works within the GH Canvas. I used a similar file name for each. They can even have the same name as they will have a different file format.

The Ring Rail (Finger Size)



The tools for **GH** are called **Components**. They are found in the **Collection of components** (below the **Main menu bar**). Click and drag the required component onto the **GH canvas**. As you get to know their names you may load each component using a short cut, which is to double-click the canvas and type its name into the resulting *search box*.

1

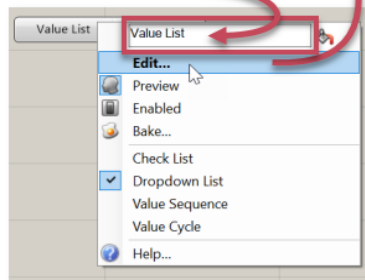
When introducing each new component I will reveal its location within its **Category tab** in the collection. Ctrl-Alt-Click on a component (*already on the canvas*) to reveal its "home" location with **red finder arrows**.

We begin by creating a list of *ring sizes*. This will allow us to adjust our model's *ring rail diameter* at any time moving forward. For this lesson, I used data from the **International Chart of Finger Ring Sizes (by Gary Dawson)** offered at [food4Rhino](http://food4rhino.com).

2

Our first component for this project is the **Value List** (see above for its location). Once you drag it onto the canvas right-click on it to reveal its settings. Select **Edit...** so that you may enter each *ring size* and its *diameter* equivalent into the **Value List Constants** box.

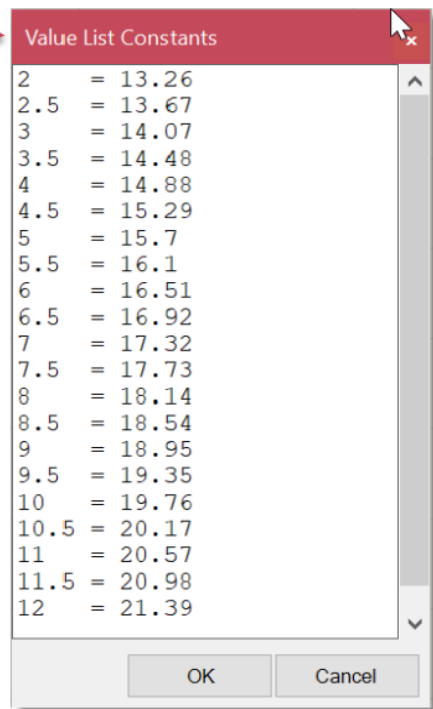
Rename here



3

Enter each *equation* on separate lines just as you see above. Press **OK** when done. Rename the component "**Ring Size**". We now have a component that converts *ring size* to a value that represents our ring rail *diameter*.

From the *drop down arrow* set the **Ring Size** to **7**.

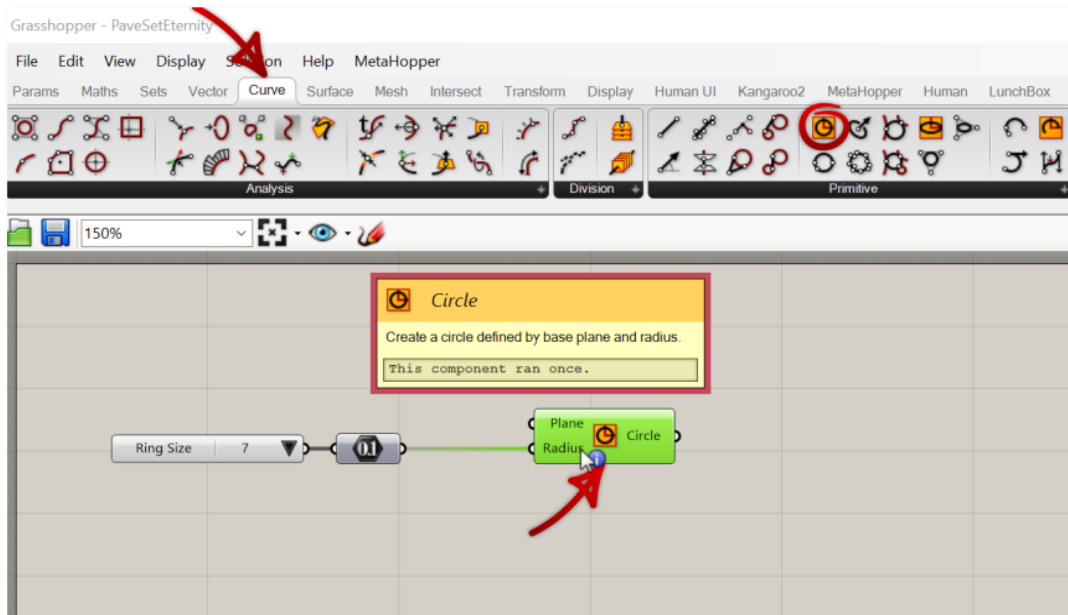


The Ring Rail (Finger Size)



4

It's good practice to use a **Number** component as a container. Although it is not necessary here, now is a great time to introduce it. Connect the *output node* from the **Value List** component to the *input node* of the **Number** component by clicking and dragging out a *wire* between them.

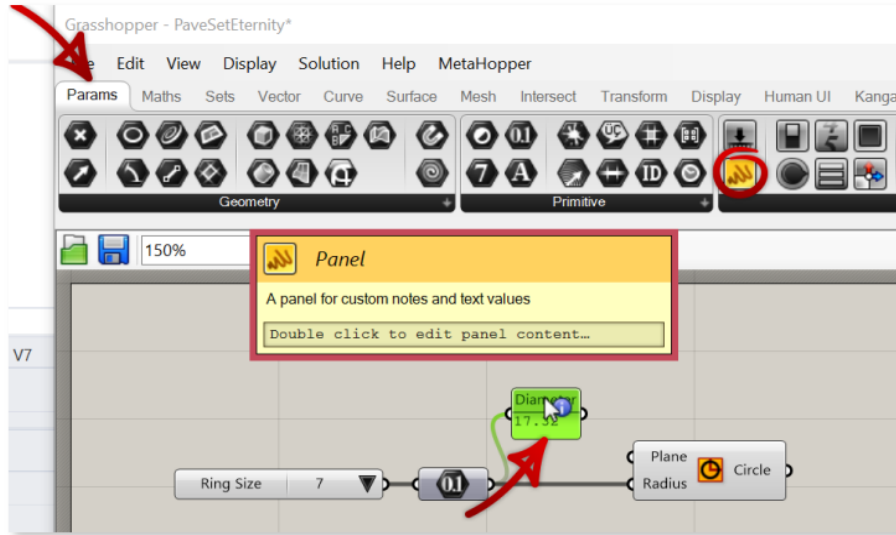


5

Now let's use our number to create geometry with a **Circle** component. Plug in our **Number** *output* into the **Radius** *input*.

You may have noticed, however, we have a number which represents the circle's *diameter* not *radius* so we shall have **GH** do some math (or **Maths** as the case may be) next...

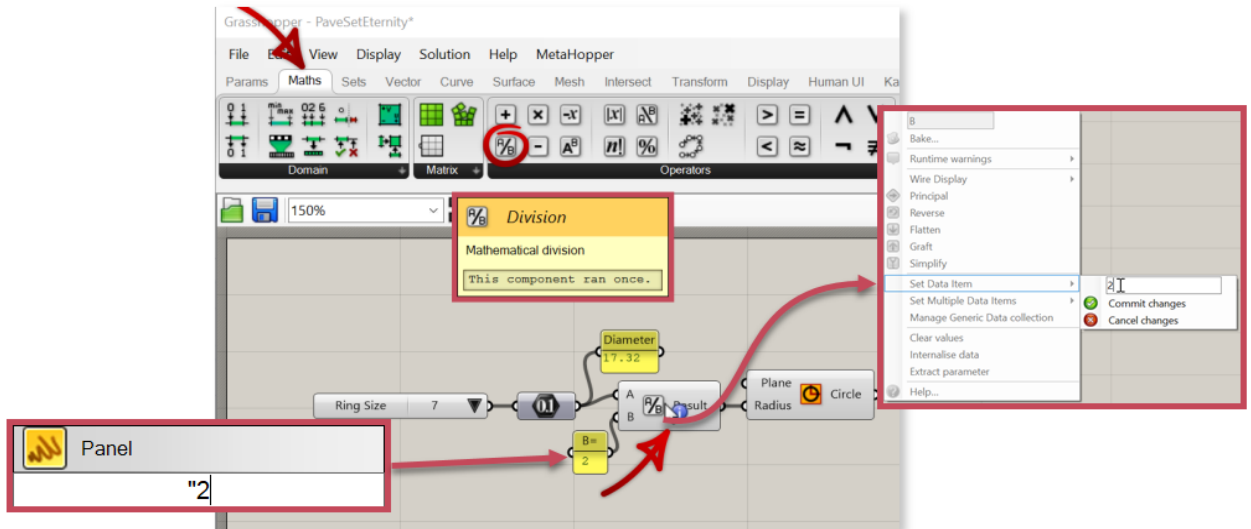
The Ring Rail (Finger Size)



First we will display our number. We can do that with a **Panel** component. This is a very versatile component as it can be used to *input* values and display *output* data as in this case. We attach it to the **Number** output node.

6

It is also useful to *relabel* or sometimes *remove the label* altogether. Here I renamed the component "**Diameter**" (*right-click* the component to do so). We can move it off to the side (or up) as it is only used for our reference.



The **Division** component is needed to divide our number in half (converting *diameter* to *radius*). We want our *diameter* as the **Dividend (A)** and our **Divisor (B)** is 2.

7

Plug the *output* from the **Number** component into the **A** input node. Then we may *right-click* on **B** input to enter "2" in the **Set Data Item** field or connect another **Panel** component with 2 in it. I prefer this method so I can quickly see it later if needed.

NOTE: A quick way to create this panel would be to *double-click* the canvas and type "2".

The Ring Rail (Finger Size)

Only draw preview geometry for selected objects

8

So let's check our progress in Rhino. First before things get really confusing we want to set our *preview*. I like to set **GH** to *only draw geometry when a component is selected*. You can quickly select this mode in the **Canvas toolbar** located below the **Collection of components**.

With the **Circle** component selected we should now see a *preview* of our work in Rhino.

9

For this exercise, we want to model our ring *vertically* so we need to change the plane to the **XZ**. *Hover over* the **Plane** input of our **Circle** component and you should see it is currently set for **XY**.

NOTE: I set my *display color* to bright green (**Document Preview Settings** - also on **Canvas toolbar**)

Grasshopper - Pavé Set Eternity*

File Edit View Display Solution Help MetaHopper

Params Maths Sets Vector Curve Surface Mesh Intersect Transform Display Human UI Kangaroo2 Meta

Field Grid Plane

150% XZ Plane
World XZ plane.

Diameter 17.32

Ring Size 7

01

A B 7/6 Result

B= 2

Origin XZ Plane

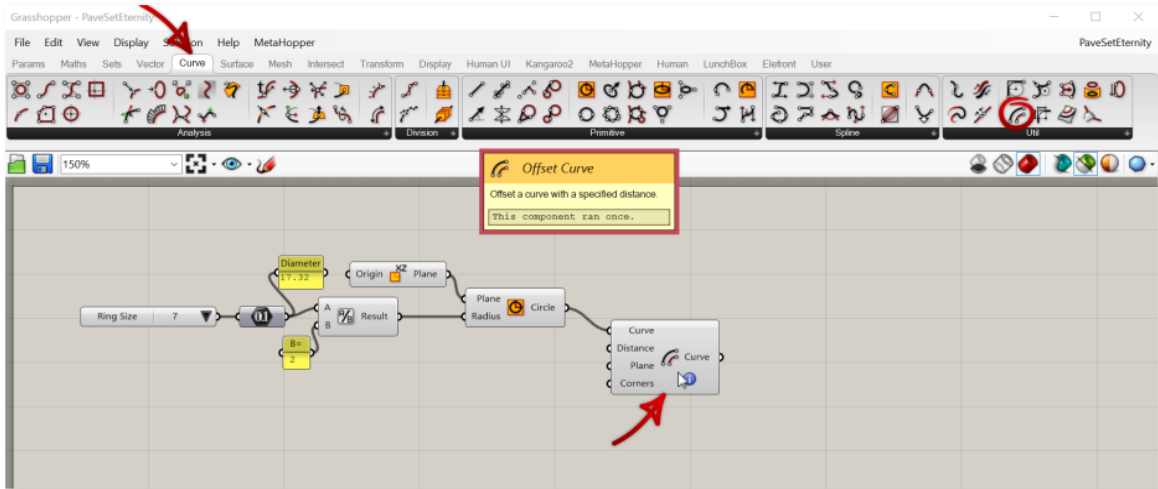
Plane Radius Circle

10

To change the plane all we need to do is drag an **XZ Plane** component and plug it into the **Plane** input of our **Circle** component and we have ourselves a functioning *finger rail*.

Play around with changing the **Ring Size** in the **Value List** component's drop down list. However, set it back to **7** before we proceed so we are all on the same page.

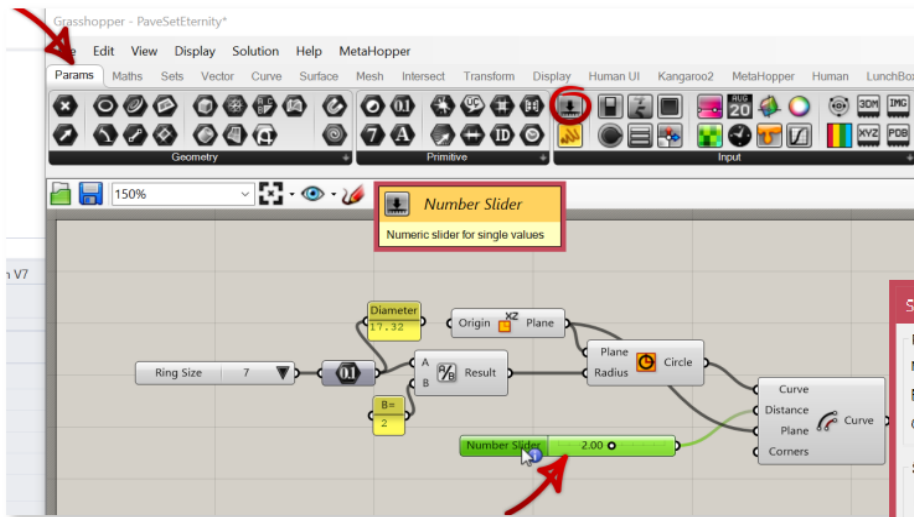
The Ring Thickness



Similar to Rhino, to create the *thickness* of the ring, we use the **Offset Curve** component. We want to feed it three things.

1

For the **Curve** input we connect the **Circle** component's *output*. We want the **XZ Plane** again and we can draw out another *wire* from the component already on the canvas (see below). Feed that into the **Plane** input.

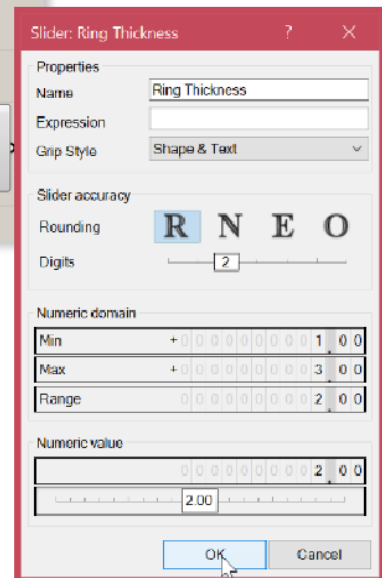


For the **Distance** input we will use a **Number Slider** component. This will allow us to easily tweak the value later. For now we will set it to 2.00 with a **Numeric domain** from 1.00 to 3.00.

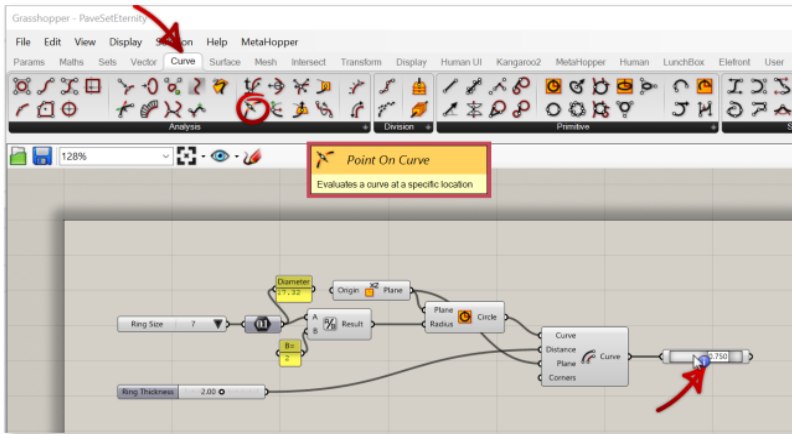
2

If you double-click on the component it will open its **dialog box** allowing us to adjust its domain, current value, and its accuracy. If we want decimals we need to set the **Rounding** to **R (Floating Point numbers)**. For 2 decimal places set the **Digits** to 2.

Finally *rename* it to **Ring Thickness** so we know what the value represents. When everything matches our example click **OK**.



Building the Ring's Profile

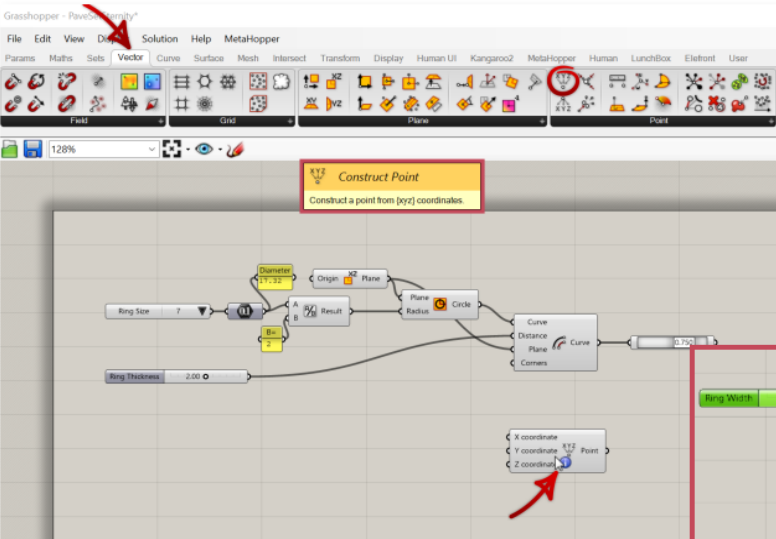
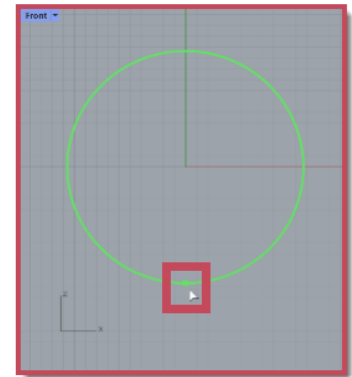


There is a wonderful component called **Point On Curve**. We can use this to get a point along the curve. For instance a value of .250 would place a point exactly at the quarter mark of the curve.

1

As I am accustomed to building a profile at "6 o'clock" we want the three quarter mark (.750). Slide the slider appropriately or right-click and select 3/4 (three quarters).

HINT: Zoom in for fine adjustments.

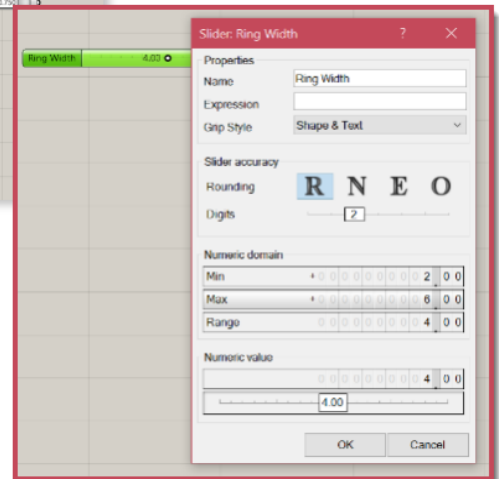


2

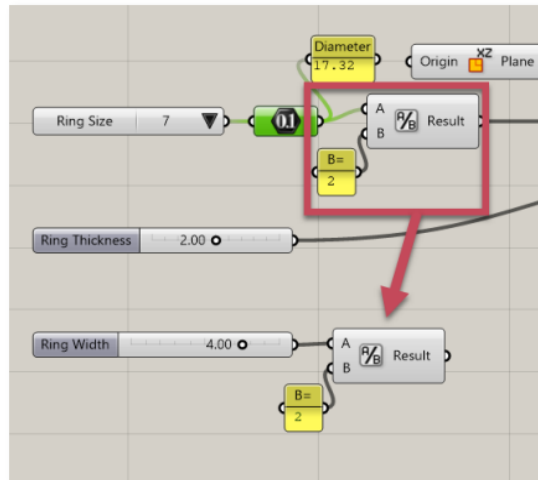
For the *width* of our profile we need to define two points. First lets drag the **Construct Point** component onto the canvas.

3

Next we need to feed it two points in the **Y coordinate input** so we can drag in another **Number Slider** component that we will label **Ring Width**. I set the domain from **2.00** to **6.00** and we will start with **4.00** for the **Numeric value**.



Building the Ring's Profile



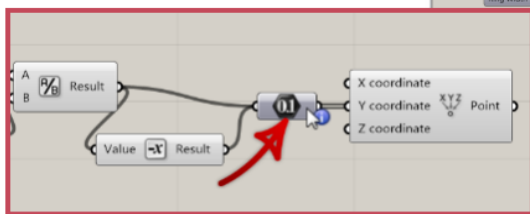
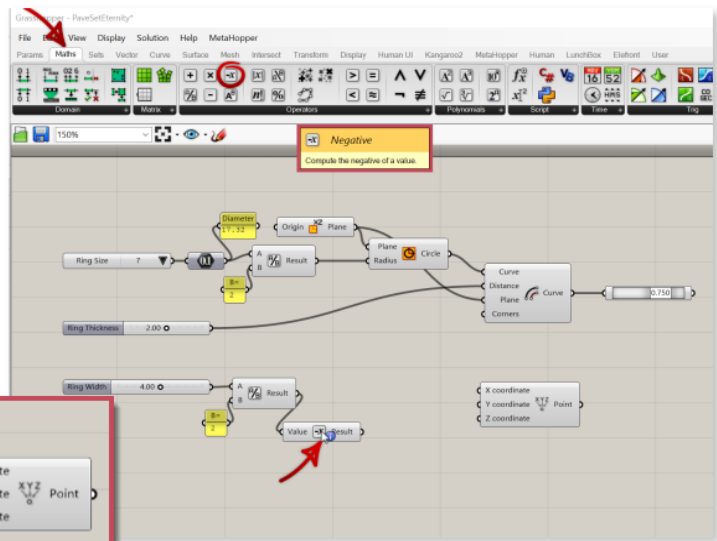
Creating a profile in Rhino, may start by using a **Single Line** cmd (from **Y=0**) using **BothSides** in the **command line**, then entering a **half value** of the line's length. The concept is similar in **GH**. As we have a number representing the width of our profile we can simply divide that number for our **positive Y** coordinate and find its **negative result** for the **negative Y** coordinate.

4

We already know how to divide using the **Division** component and it already has the correct **Divisor** of 2. So let's copy that by selecting both and dragging them down the canvas. While dragging **tap** the **Alt** key. Next connect the **"Ring Width"** slider into the **A input**. This will replace its previous input.

5

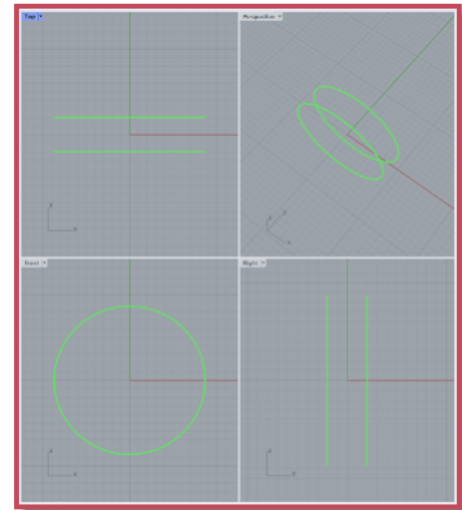
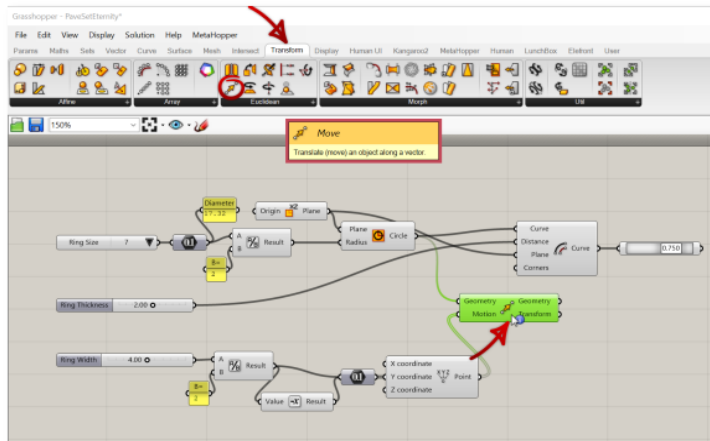
For the negative result we use a **Negative** component.



6

To combine both numbers let's feed them both into a **Number** component. You can do this by connecting a wire from the **Division** output into the **Number** input. Then do the same from the **Negative Result** output but this time hold the **Shift** key. This allows us to connect multiple inputs to a node. Plug in the **Number** output into the **Y coordinate** input of the **Construction Point** component.

Building the Ring's Profile

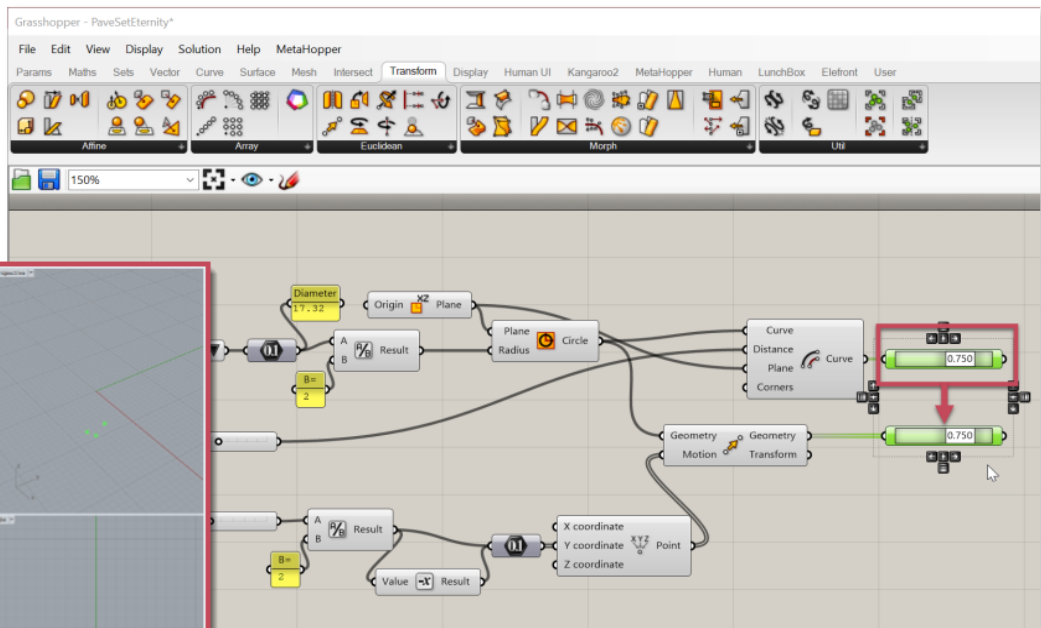
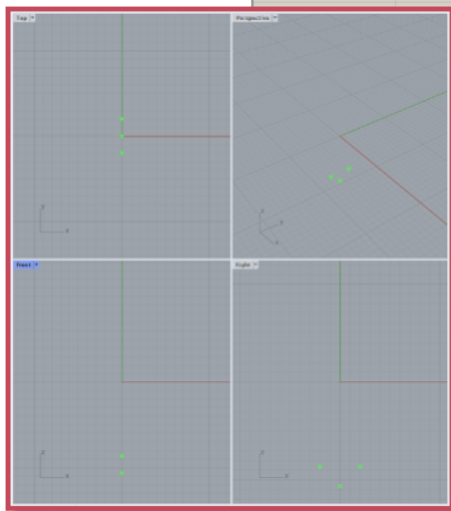


We now have two points with assigned *Y coordinates*. Now we need to set their *Z coordinates* to the *finger rail's "6 o'clock"*. First we will use a **Move** component.

7

If we plug in our **Circle** output into the **Geometry** input of the **Move** component and use our points *Y coordinate* data we create two additional circles representing the ring's width. Connect the *output* of the **Construction Point** component into the **Motion** input.

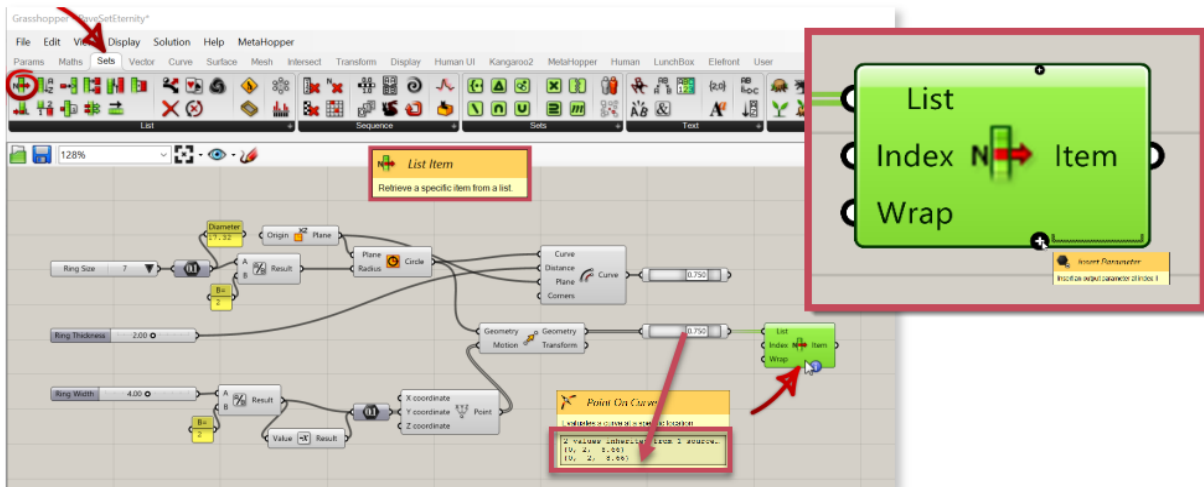
Now we need each circle's point at "6 o'clock" and we have done this operation before with...



8

the **Point On Curve** component. Let's *copy* that and plug in the resulting **Geometry** from the **Move** component. With both selected we see in Rhino that our profile is beginning to take shape.

Building the Ring's Profile

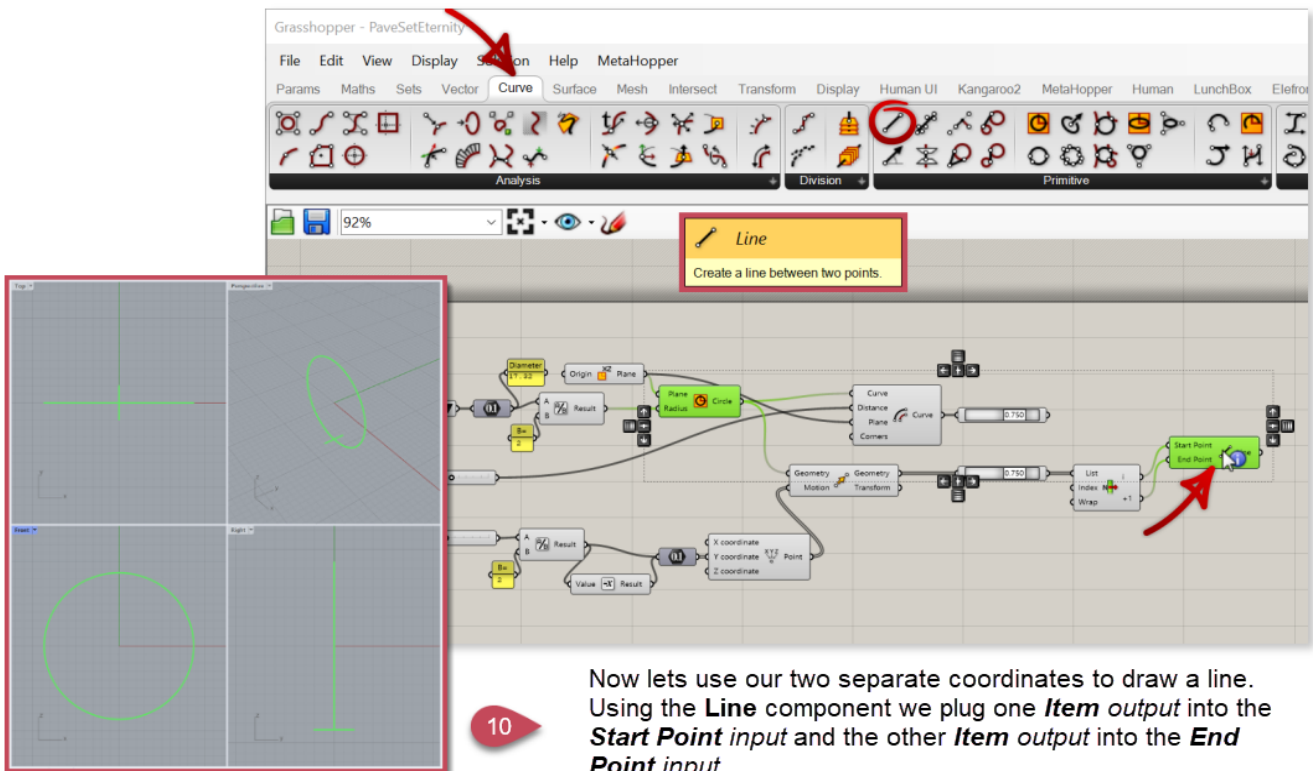


Components house data that we feed into them. GH has multiple ways of *sorting and retrieving* the data we need for our project. In our last component (**Point On Curve**) we have multiple coordinates (hover over the component to see).

9

Also, notice that there are **double lines** between the last three components. These represent a *list* of data rather than a *single data* source.

To separate data on a list we can use a **List Item** component and *add* outputs by *zooming in* until you see the **Insert Parameter** button. Add one more item.



10

Now lets use our two separate coordinates to draw a line. Using the **Line** component we plug one *Item* output into the **Start Point** input and the other *Item* output into the **End Point** input.

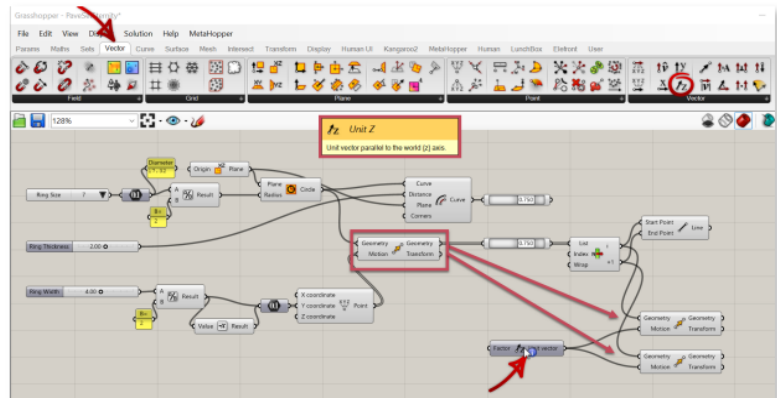
Building the Ring's Profile

11

To shape our profile further we can use the **Move** component on our "width" points. Bring the **Move** component to the canvas *twice*.

12

For their **Motion** input we want to add a **Unit Z** component. This will establish a vector along the **Z-Axis** (moving the points up or down depending on the value we feed it).

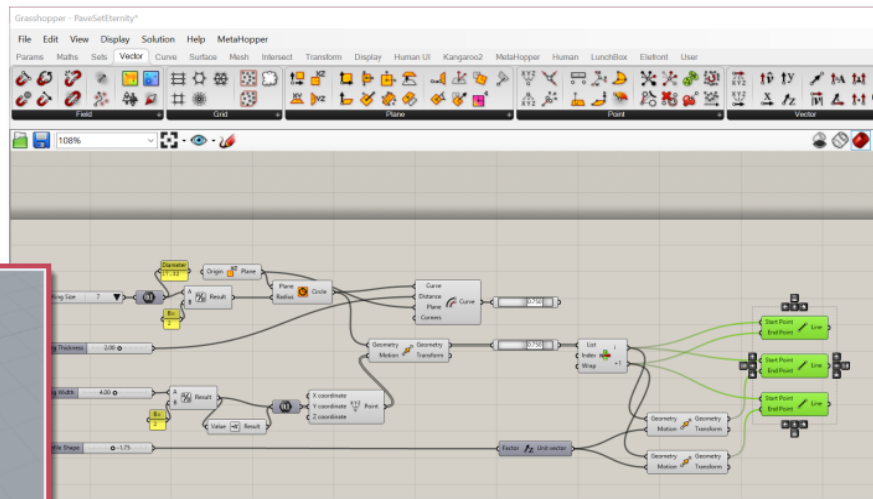
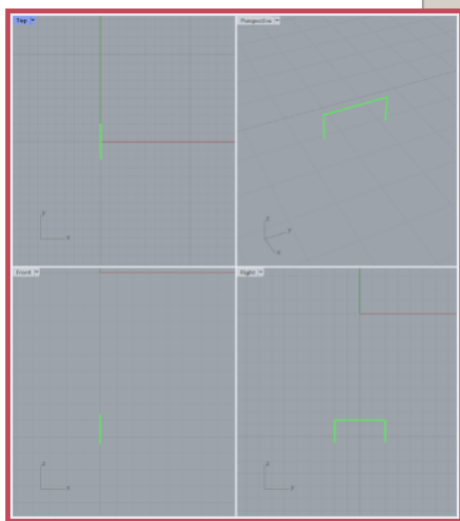
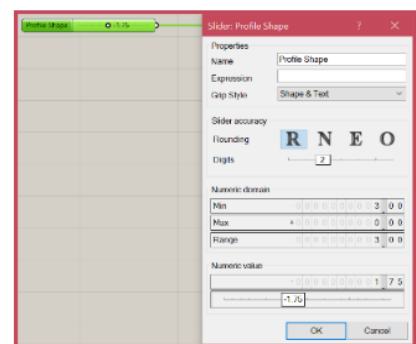


Let's create another **Number Slider** to do so.

13

As we don't want to move these points higher (into the finger) we set the **Max domain** to 0.00. Set the **Min** to -3.00 and **Numeric value** to -1.75.

I called it "**Profile Shape**" as it is more about *aesthetics* than precise measurements. Plug the slider into the **Factor** input of the **Unit Z** component.

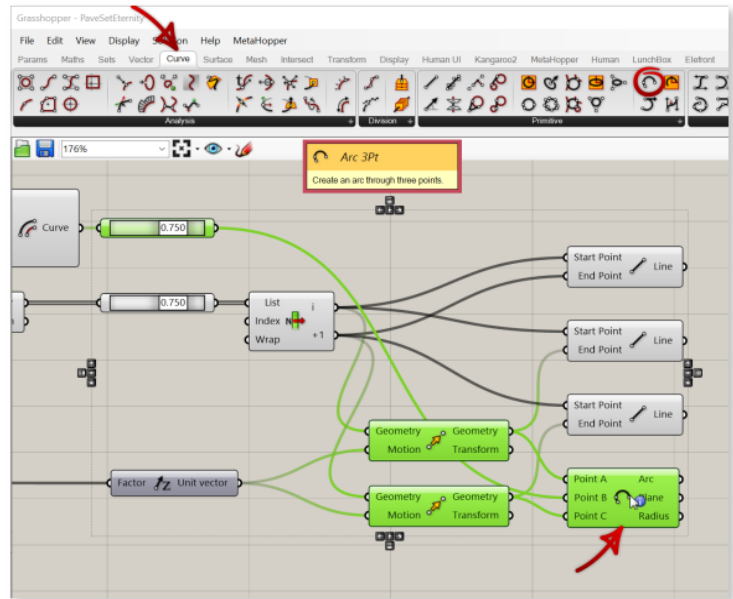
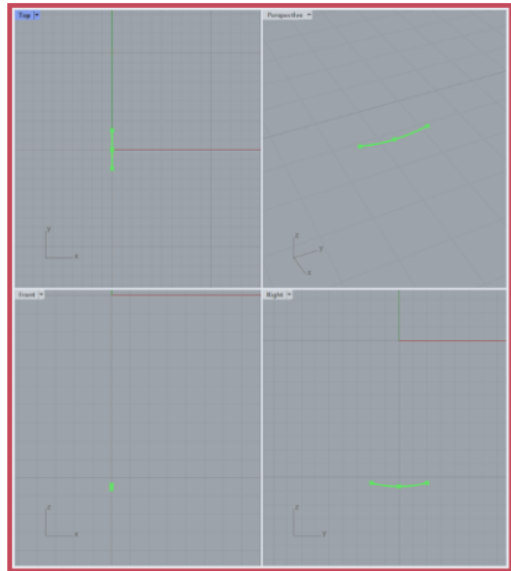


14

Using the same method as before we will connect the points by drawing two lines. We use the **Line** component *twice* more and pair each previous point to their respectively **Z-translated** counterpart.

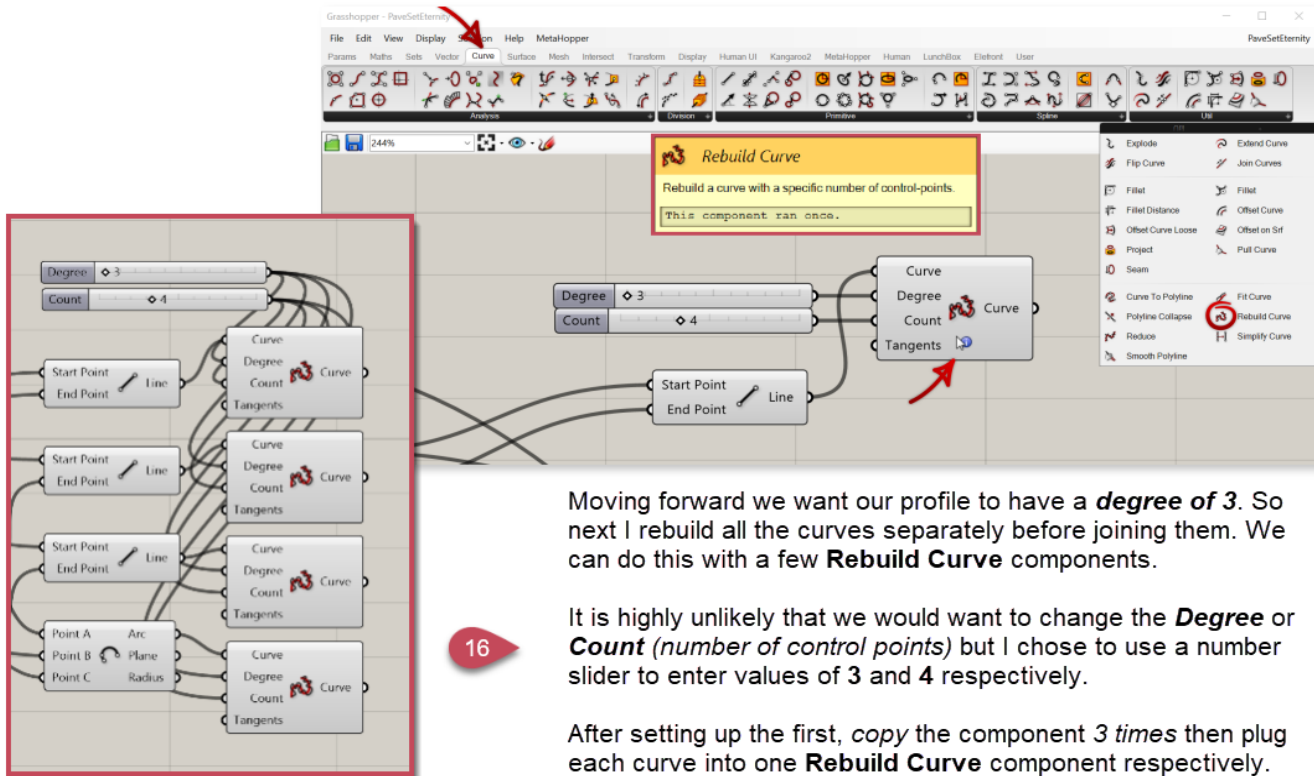
With all three **Line** components selected check your progress in Rhino. Should you have different results (*like a diagonal line*) you most likely have your wires crossed (*literally*).

Building the Ring's Profile



15

Let's close the bottom of our profile with an arc. Using the **Arc 3Pt** component we plug in **Point A** and **Point C** using the *output* from our **Move** components. **Point B** comes from the *output* of our first **Point On Curve** *output*.



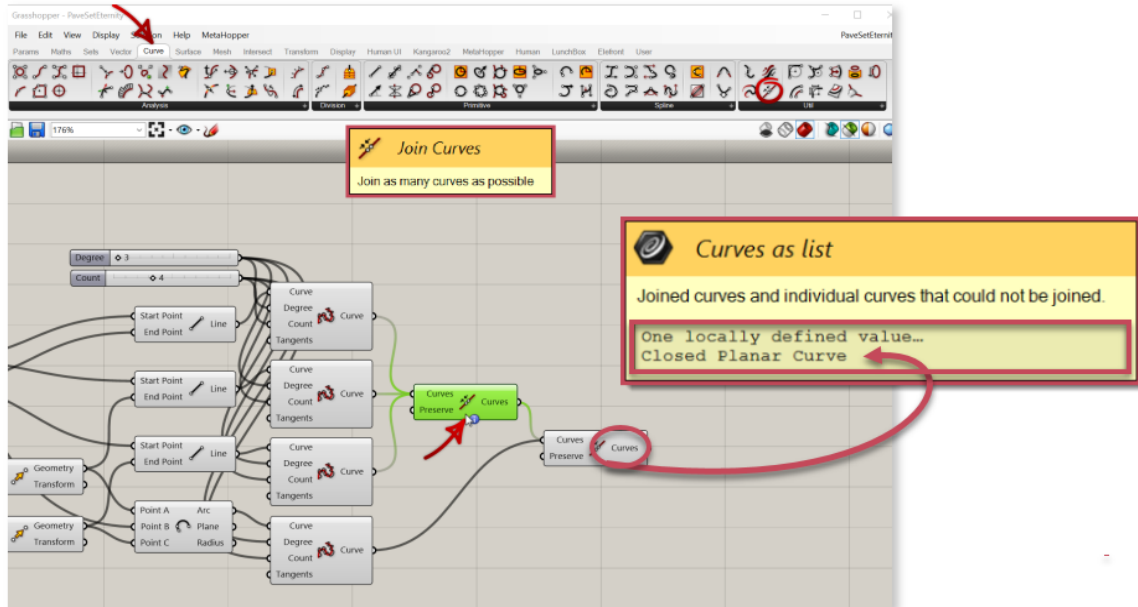
16

Moving forward we want our profile to have a **degree of 3**. So next I rebuild all the curves separately before joining them. We can do this with a few **Rebuild Curve** components.

It is highly unlikely that we would want to change the **Degree** or **Count** (number of control points) but I chose to use a number slider to enter values of 3 and 4 respectively.

After setting up the first, *copy* the component 3 *times* then plug each curve into one **Rebuild Curve** component respectively.

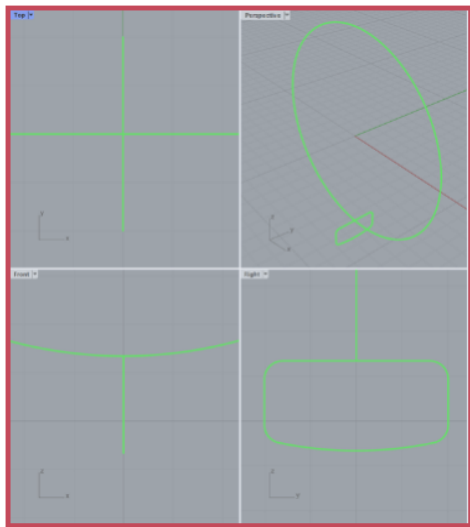
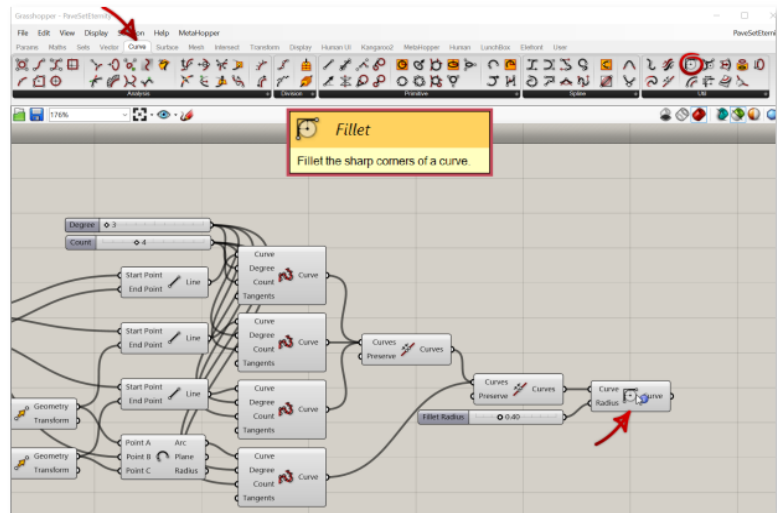
Building the Ring's Profile



Now use the **Join Curves** component but it works best to join the lines first then run the component a second time adding the arc.

17

Always check your output (hover over the **Curves** output) as it needs to be one **Closed Planar Curve**.

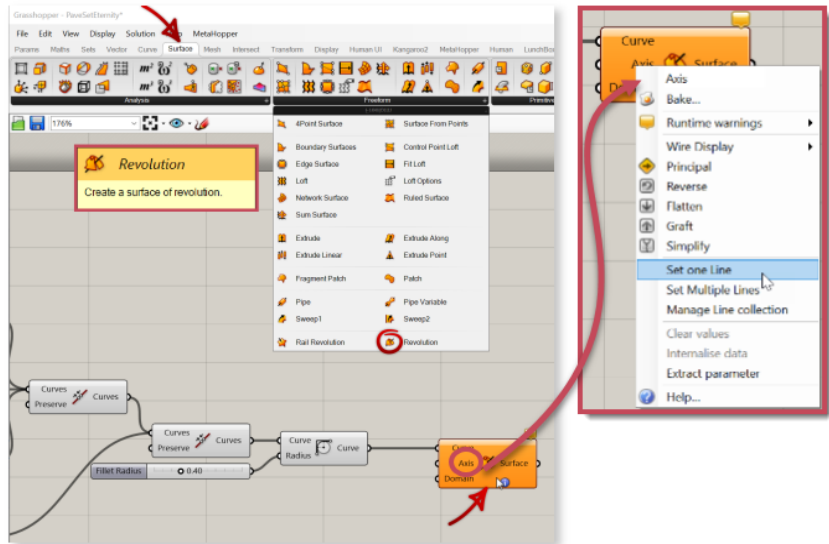
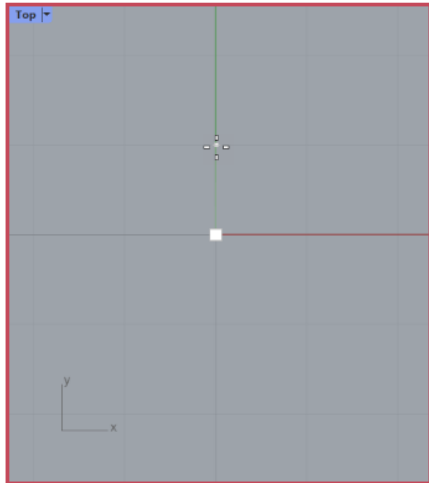


18

Now we can fillet the curve to our liking. Drag in the **Fillet** component. The *radius* can be changed later so we want to input a **Number Slider** (with 2 decimal places) for the **Radius** input.

Check your progress by selecting the **Fillet** component. Here I have also selected the **ring rail Circle** component to preview both simultaneously.

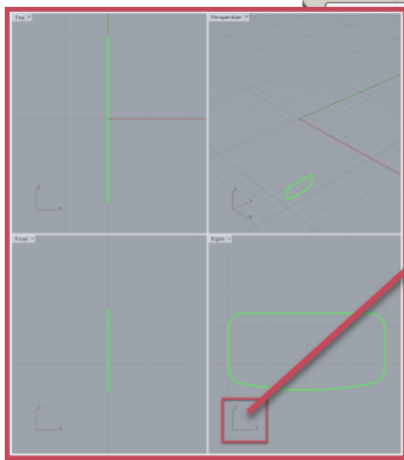
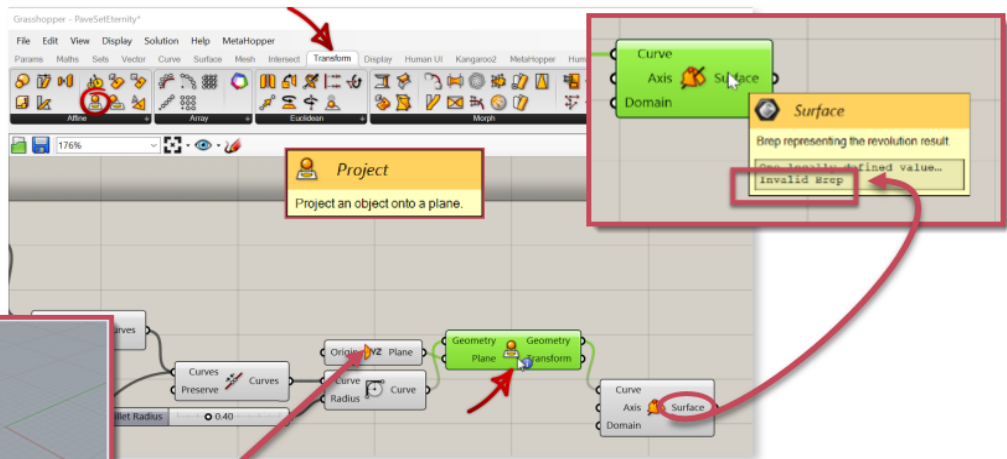
Building the Ring's Profile



To create our ring's surface we can use the **Revolution** component.

19

You will need to set an axis and I find quickest way to do that is to right-click on the **Axis** input then draw a line in the **Top Viewport** in Rhino. The **Start of line = 0**. **End of line = Shift-click 1 unit** along the **Y-axis**.



20

Check your results by hovering over the **output**. I get an **Invalid Brep**. What's a Brep? It stands for **boundary representation** and it is a **combination of multiple surfaces**. You will use them often but what we are looking for here is **One Untrimmed Surface**.

Let's project our curve onto a plane using the **Project** component and see if that fixes it. We want to project (flatten) it onto the **YZ plane** so we use a **YZ Plane** component from the **Plane** panel in the **Vector** tab.

Building the Ring's Profile

21

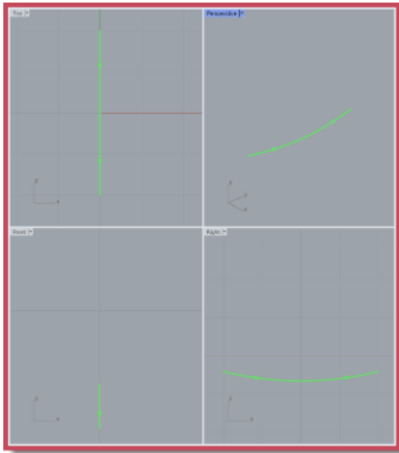
Success! We now have a surface we can *bake*. But our *preview* is amiss as our *normals* are flipped. Let's straighten those out with a **Flip** component. It should now look like the image below.

22

Let's end this section by *labeling* and *grouping*. To place a label on the canvas we use the **Scribble** component.

Double-click it and change the name to "**Band**". To *group* the components select both and type **Ctrl-G**.

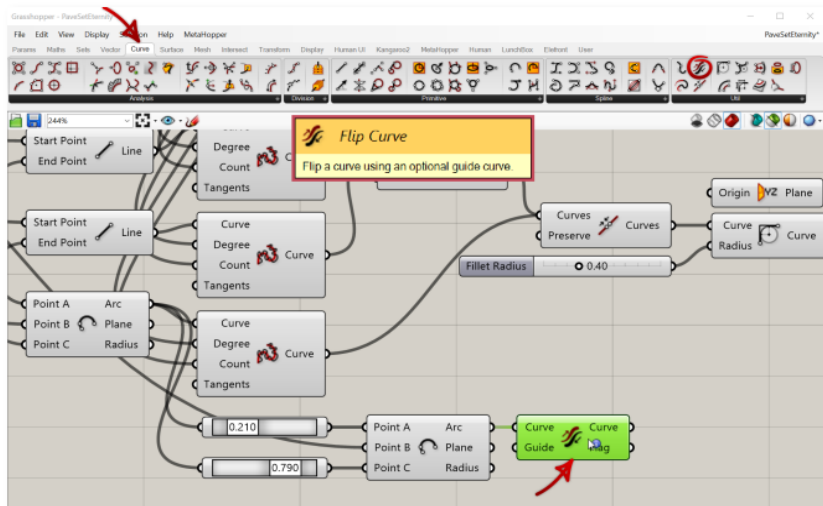
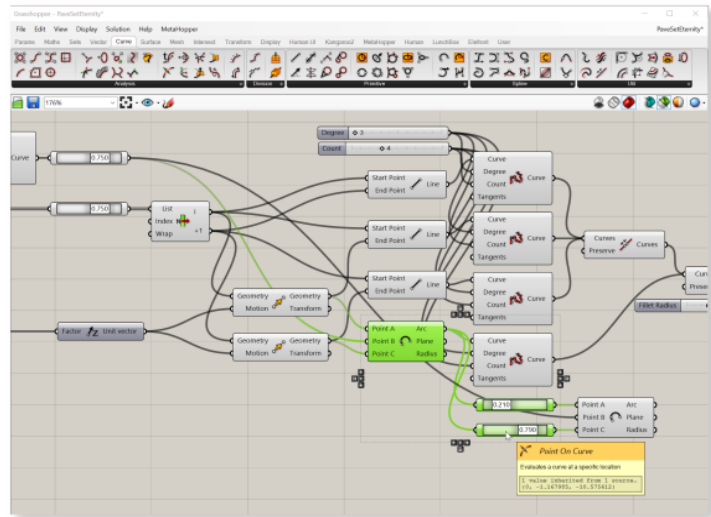
The Stone Layout



For the stone layout we are going to use a component called **Sporph**. This process is similar to how we would use the **Flow Along Surface** command in Rhino. The idea is to *unroll* the 3-D area we want the stones, beads, and cutters so that they can be applied to points in UV space (2-dimensional), then assign those points to the nearest 3-dimensional space once it is "*rolled*" back into a ring. While there is a little stretching involved it can be minimized by applying it to half the ring then mirroring that to keep its *symmetry*.

To begin we need to define the area on the ring where we want to place the stones. We can use the arc of our profile to define this area's width. Let's use **Point On Curve** to find points around 1/5th into the arc from either side. Here, I begin with .210 and .790. We will adjust these later when the stones are in place. Plug those into another **Arc 3Pt** component and create a new arc.

1



You are most likely familiar with *normals* not always pointing in the direction you want them. This is because the directions we choose to build our elements of a surface may not always be in union with one another. No worries, we often, and easily change their directions.

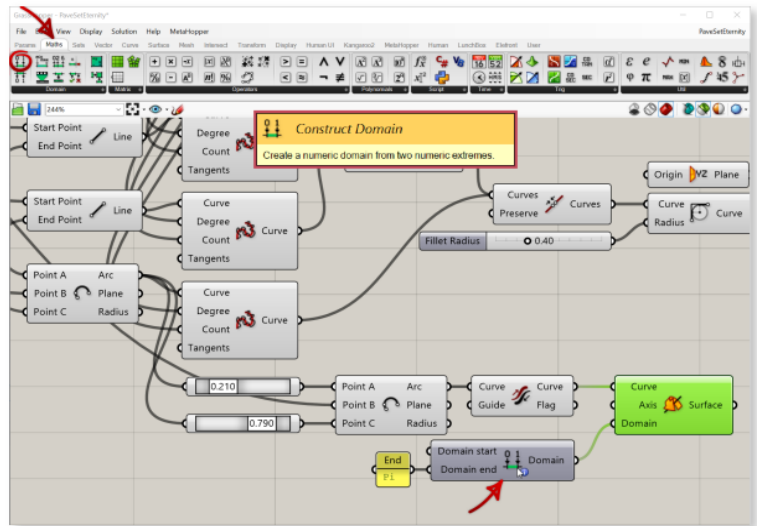
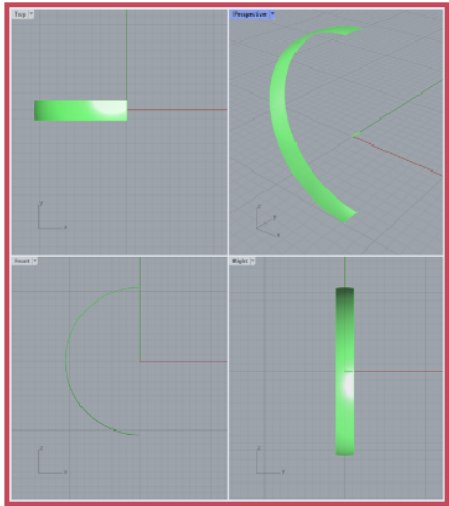
2

We will eventually need to *flip the normals* so that the tables of our diamonds face outward. I find it easiest to *flip* our curve now. Use the **Flip Curve** component to do so.

The Stone Layout

3

Next we want to use the **Revolution** component again. Set the **Axis** input as you had previously (see pg 18; step 19). Here I only want *half* a revolution (180°).

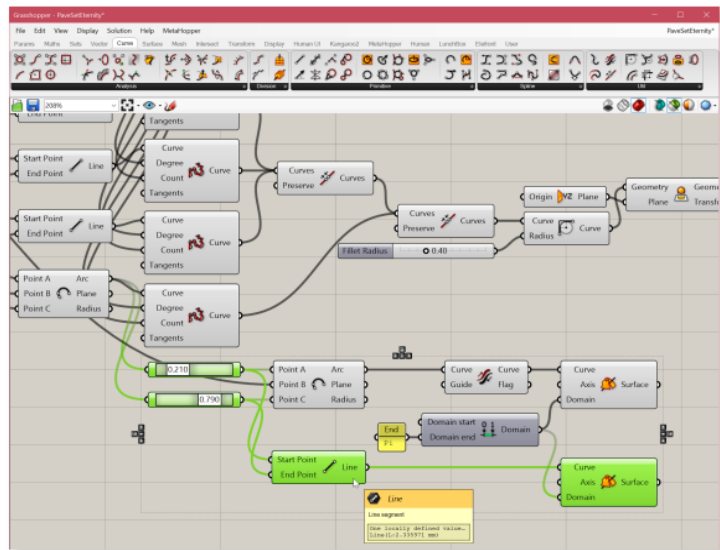
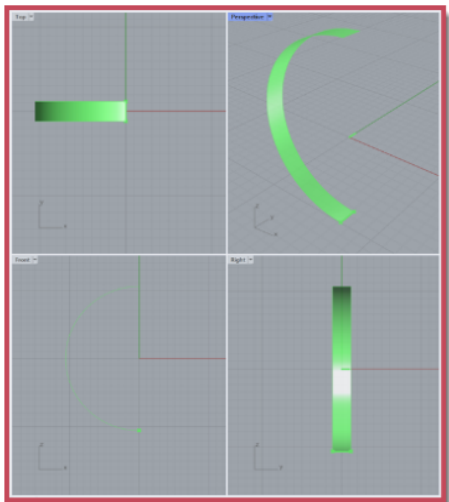


4

We assign this by plugging a **Construct Domain** component into the **Domain** input. The **Domain start** remains 0. For the **Domain end** we want to use π (Pi) not 2π (the full circumference).

Simply create a **Panel** with "Pi" (double-click on the canvas and type "Pi").

Simultaneous to building our 3-D area we need to create a source 2-D space for our point layout. We start this process in 3-D space first as we need a *flat-ribbon* representation of the space that we will *unroll* next.



5

We will simply take the **end points** of our arc and connect those with a **Line** component.

Then **copy** our **Revolution** component and plug in the same **Construct Domain** component.

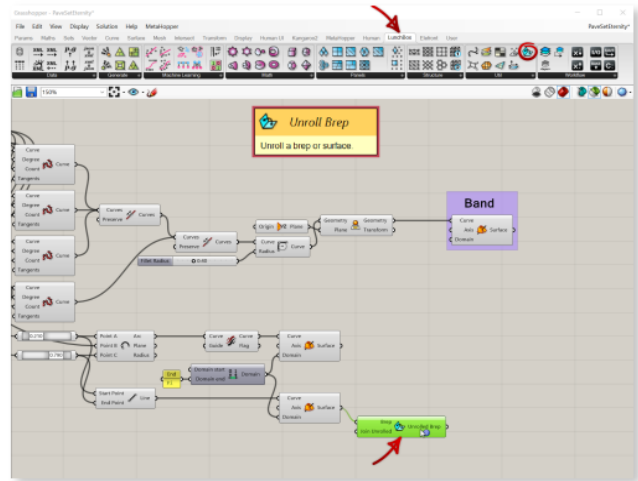
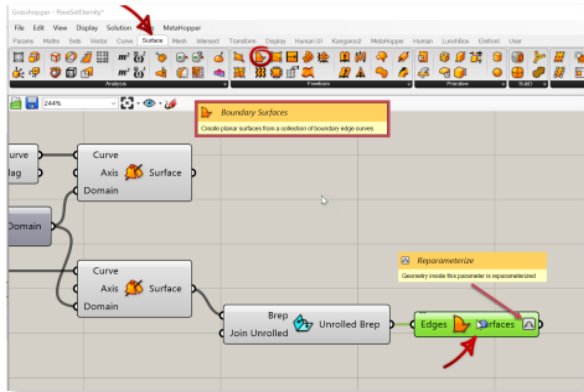
NOTE: It is not necessary to flip this line.

The Stone Layout

6

Our next component that I have alluded to is **Unroll Brep**. It is from the **LunchBox plug-in** for Grasshopper. Here is the link to install it: <https://www.food4rhino.com/en/app/lunchbox>

Plug in your "flat-ribbon" surface.



7

Follow this with a **Boundary Surfaces** component. Finally right-click on the **Surfaces** output and select **Reparameterize**. This is necessary to convert our points into *UV space*.

Reparameterize allows us to take incoming data and *reassign* its *domain values* from **0 to 1**. It is a valuable short cut that helps us to select data in percentages. For instance, if you *reparameterize* values of **0 to 500**. The value of **125** now becomes **.25** (25% of 500).

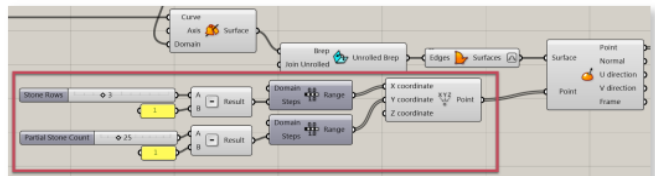
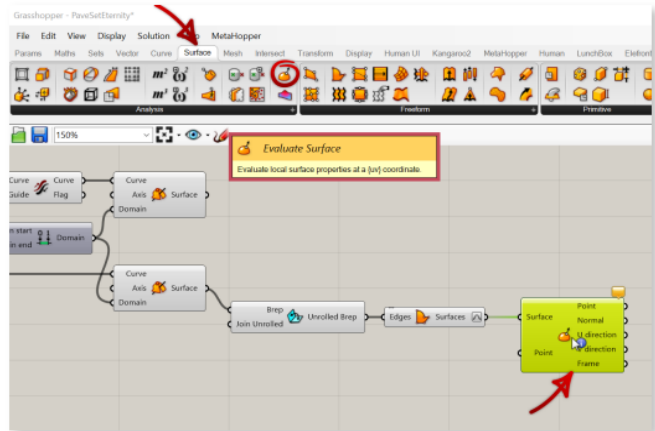
Next we need to figure how many diamonds. This depends on their size and ring size, width, and thickness. For this exercise, the client wants to use several **one-pointer diamonds**. This refers to **0.01 carat weights** which is roughly 1.3mm. I'll add a little more for variances so we will use a fixed measurement of **1.35mm** for our stones, thickness, and gaps between stones and we come up with a number of **48**.

We use an **Evaluate Surface** component to convert the points that will represent the locations of our diamonds.

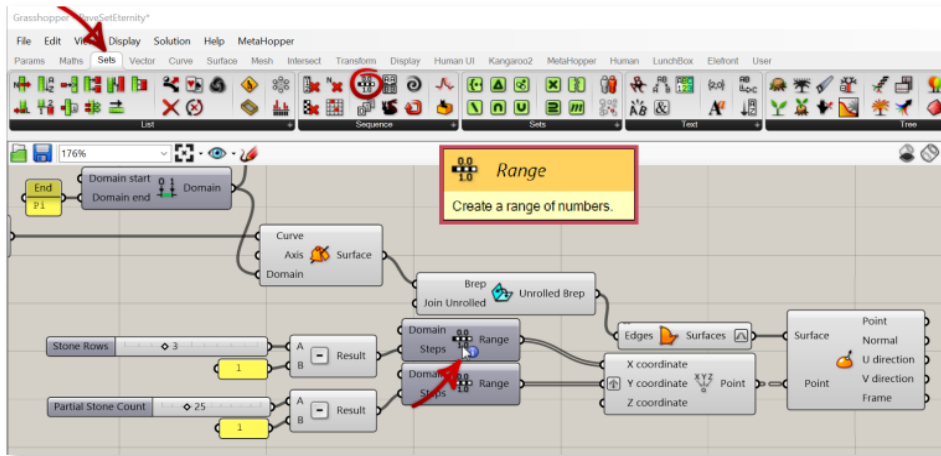
8

As we are working with *half* of the ring there will be an overlap at the top and the bottom. Therefore we want to have **25 stones mirror** them and then **subtract the two duplicates**.

Furthermore, our input needs to be **1 less** than what we might think because of the way **GH** properly divides the space. Therefore we want to create our points using the displayed method above. I will break it down next...



The Stone Layout

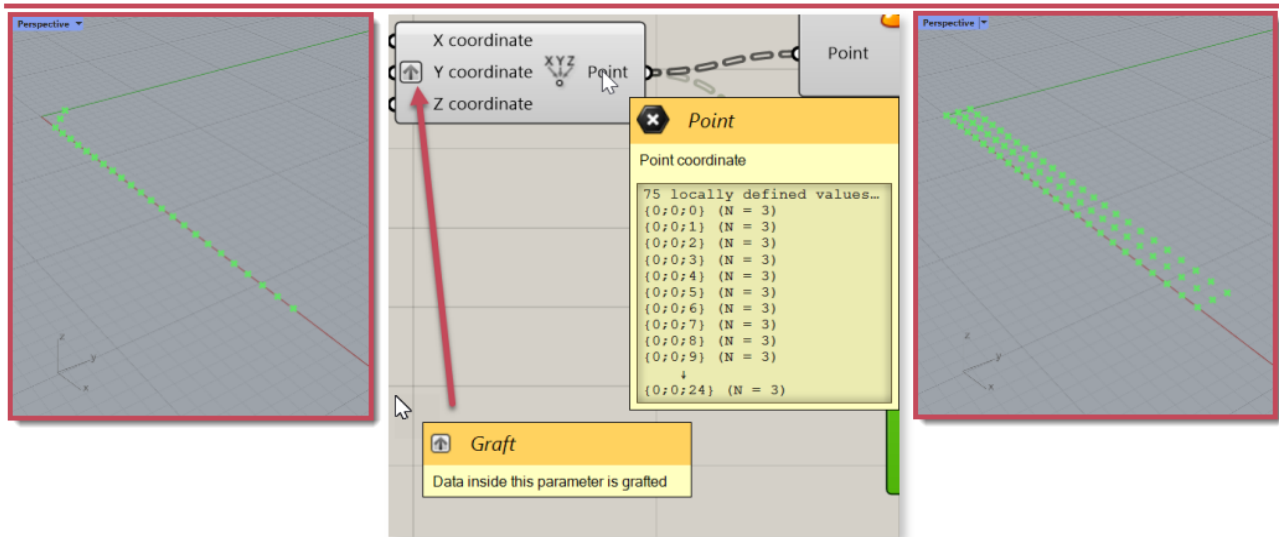


First we need a **Construct Point** component. Now say I want **three rows** with **25 diamonds** each. We can use a **Range** component to supply our points. This component allows us to set **Steps** which divides our **Domain (0-1)**. Because there will be a point on either end of our **step (division)** we will have one additional point.

9

So we use the **Subtraction** component. You don't need to subtract if you set your **Number Sliders** to **2** and **24** but then labeling them becomes an issue.

Feed your **Stone Rows Range** into the **X coordinate input** and your **"Partial Stone Count" Range** into the **Y coordinate input**.

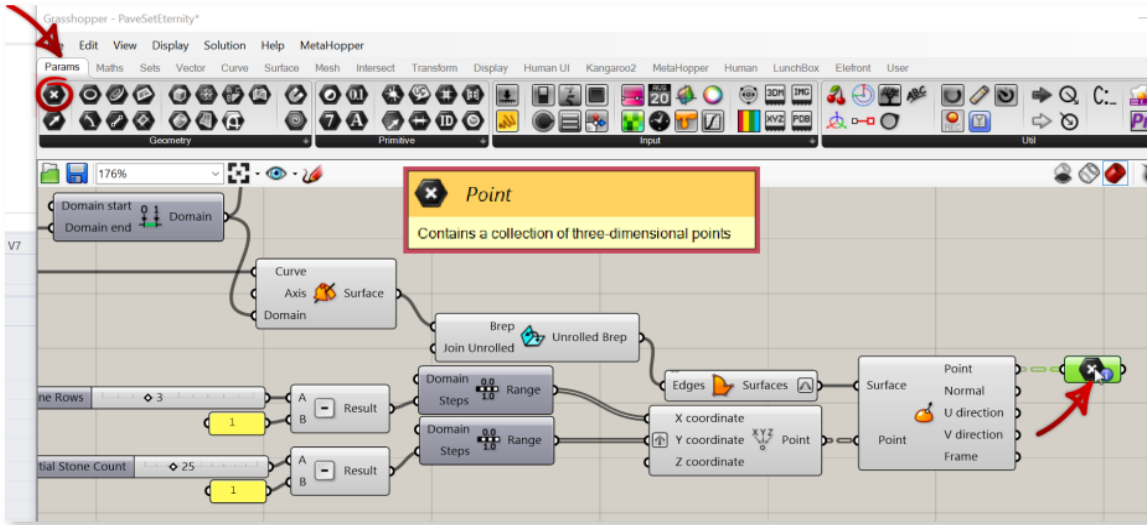


Turn on **Graft** (up arrow) for the **Y coordinate input**. The illustration above shows our points without grafting (left). This has a total of **25 points**. Our goal is to have **25 groups of 3 points** for a total of **75 points** (right). **Grafting** is a way to group our data differently. **Grafting** either the **X** or the **Y coordinate** (not both) will group our points in a way that gives us **75 points**.

10

Grafting the **X coordinate input** and not the **Y** will organize our data in **3 rows of 25**. This may sound perfect, however, so that I may use a future component, I want to graft the **Y coordinate input** and not the **X**. It should make more sense to you as we proceed.

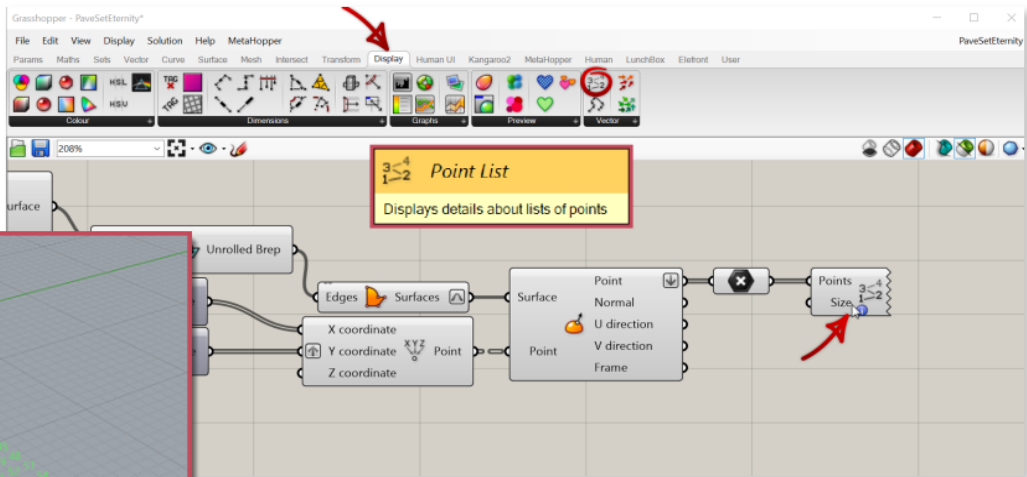
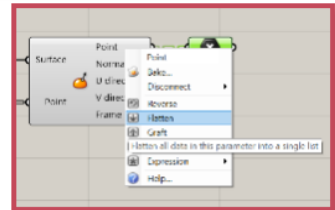
The Stone Layout



We need to hook up a **Point** component to the **Point output** of our **Evaluate Surface** component. This is a container that will store the data of our 75 points.

11

We often want to **flatten** our data which will change the way our points are organized again. It sort of "ungroups" them into a single list of data. **Flatten** the **Point output**.

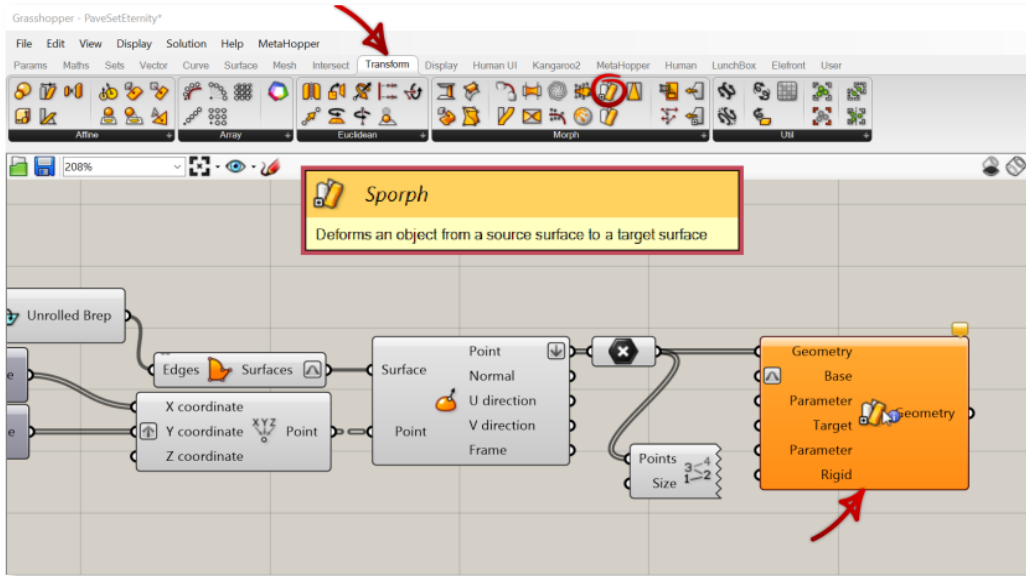


12

One of the most valuable tools for helping understand when to **graft** and **flatten** your data is the **Point List** component. Bring that in and supply it with the point data. This will number the points in 3D space in order they are listed. You can change the size of the numbers by setting a value for the **Size** input.

Notice how the numbers correspond to their point. Look at the difference in the list if you turn **Flatten** back off or changing the way we previously **grafted** our data. Afterward be sure to set everything back. We want our numbers to look like the above example.

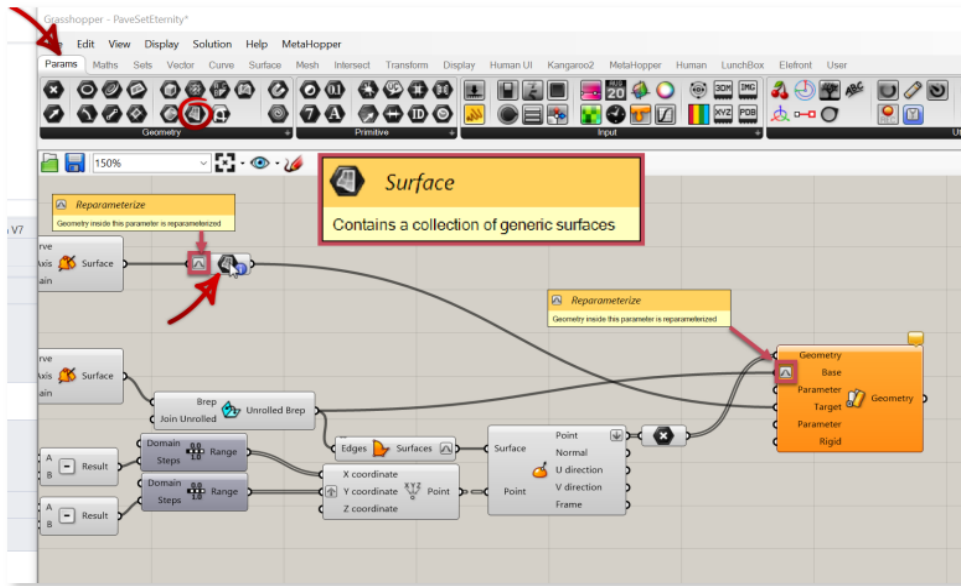
The Stone Layout



13

We have finally arrived to the crux of our project. The **Sporph** component allows us to bring our flattened points back into the 3-D realm. Connect our *flattened* point data into the **Geometry** input.

Similar to *Flow Along Surface* we need a reference for the **Base input** and one for our **Target input**. Good news! They are done already. Mostly...

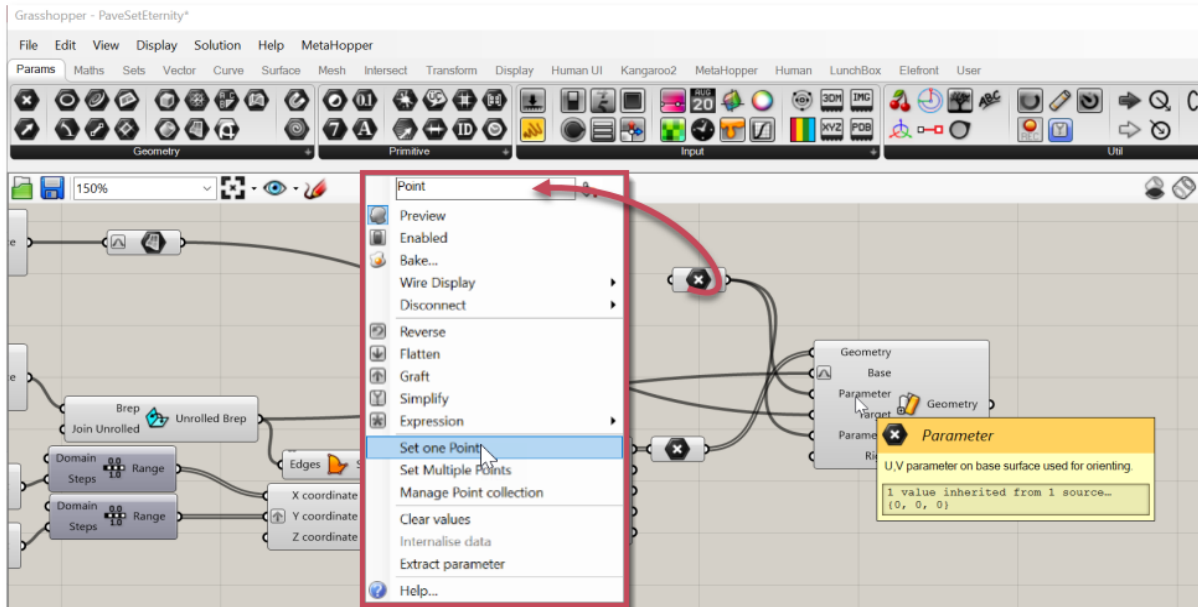


14

We are going to take **Unrolled Brep** output from the **Unroll Brep** component and feed that into our **Base input** (set it to **Reparameterize** the data).

Next we are going to connect the **Surface** output from the **Revolution** component (*step 3; pg 21*) to a **Surface** component. **Reparameterize** the *incoming* data. Plug the *output* into the **Target input**.

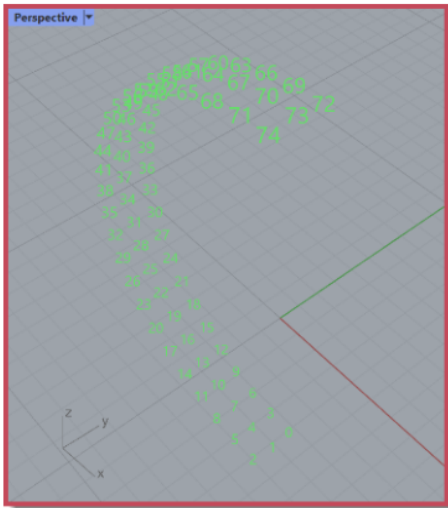
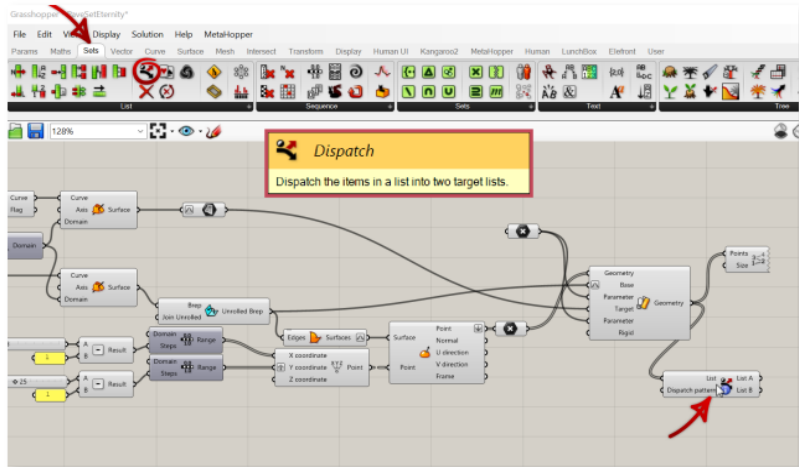
The Stone Layout



The **Parameter Inputs** need to correspond. This represents *UV space* (similar to picking near a corner for *Flow Along Surface*). Here I want to set both to $(0,0,0)$.

15

I chose to use a **Point** component. Right-click and select **Set one Point**. Create a $0,0,0$ point in Rhino. Plug the **Point** into both **Parameter inputs**.



We want to shift the points in the middle row so that our diamonds are staggered from each other. Here I use the **Dispatch** component to *split* our point list based on a pattern.

16

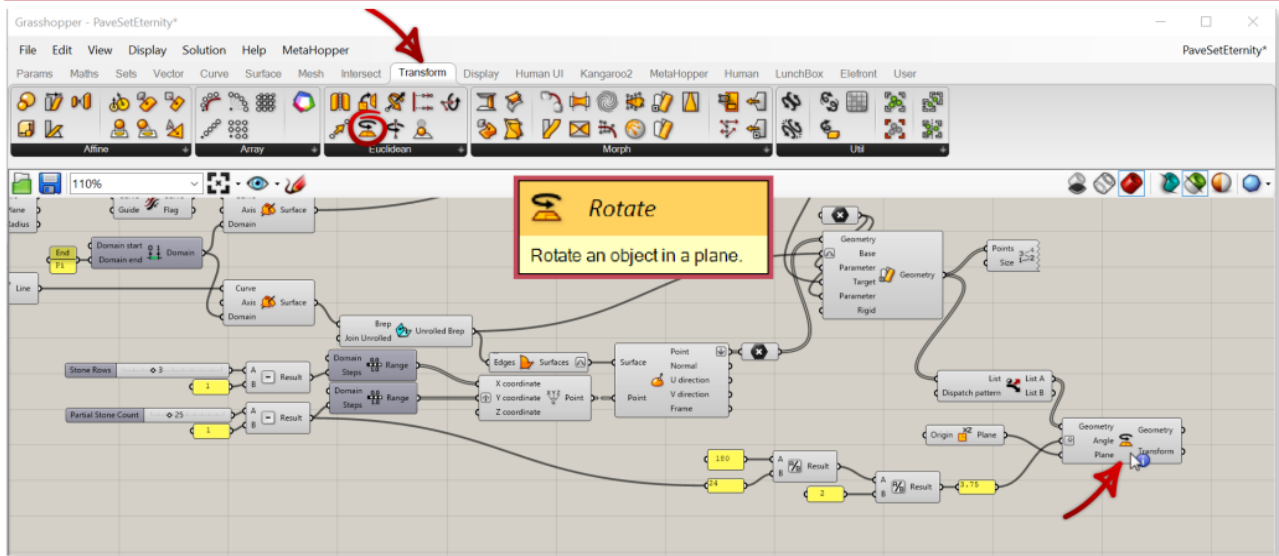
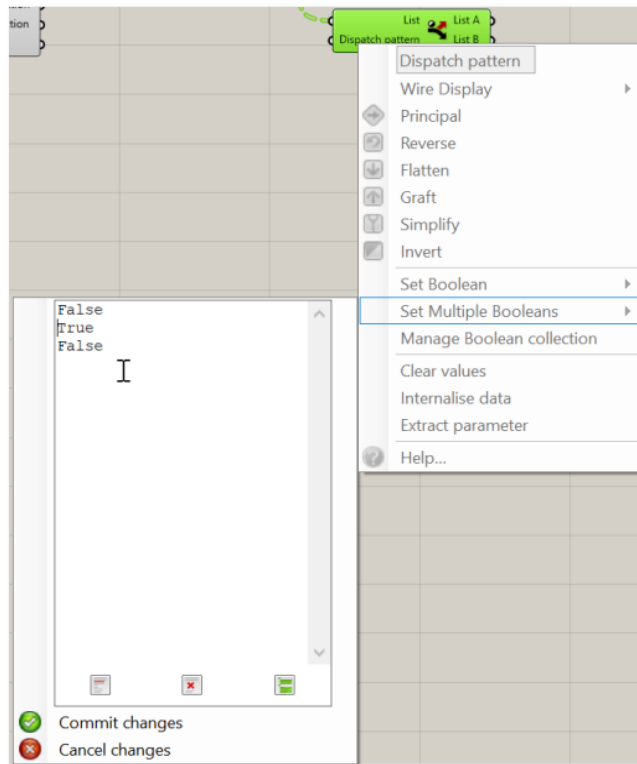
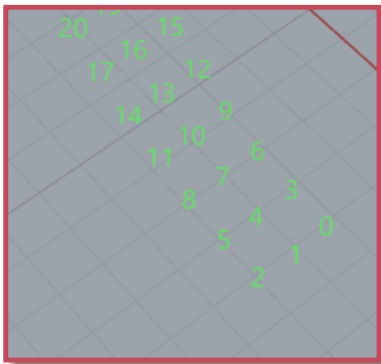
Remember how we set up our *grafting*? Rather than 3 rows of 25 we have 25 sets of 3. Now we can retrieve our data using a simple pattern. I want to *dispatch* data (the middle row) into **List A** output that we can manipulate further. Look at our **Point List**. We want $1,4,7,10,13, etc...$

The Stone Layout

We **Set Multiple Boolean** (*right-click*) to **False, True, False**. Through a *boolean pattern* we are directing the component to dispatch all **True** to **List A**.

17

The pattern begins with *False* at 0. Then *1 = True* and *2 = False*. Cycle through again and *3 = False*, *4 = True*, *5 = False*, and so on. Our **List A** becomes the *middle row*, while **List B** is everything else.

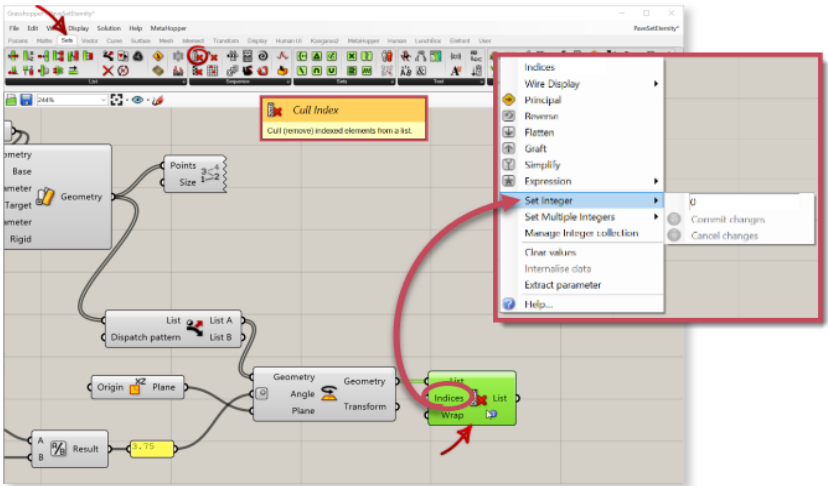


Now that we have our **List A** output we want to plug it into the **Rotate** component (**Geometry** input). Note that **GH** work natively with *radians* not *degrees*. Be sure to change the **Angle** input to accept **Degree** data.

18

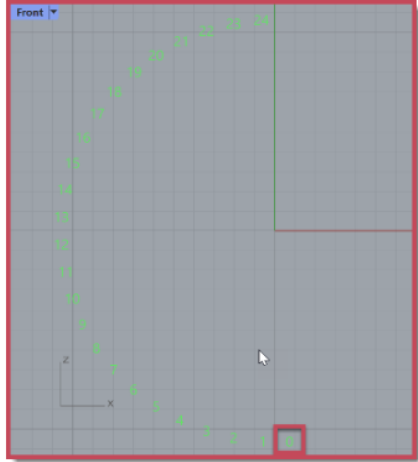
We want to stagger these points *midway* between the points on the edge rows. To do this we *divide 180* by our **Count-1** (**24**). Then *divide* that in *half* again. In this case it is **3.75 degrees**. Plug this into the **Angle** input.

The Stone Layout



Cull Index
Cull (removes) indexed elements from a list.

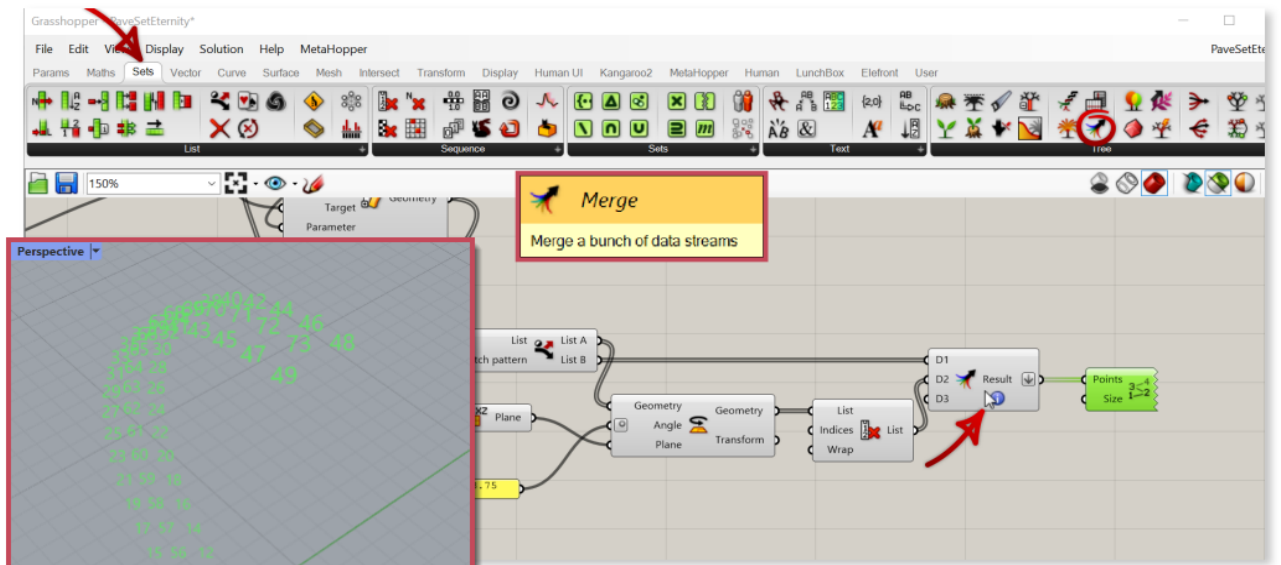
List
Indices: 0
Set Integer: 0
Set Multiple Integers
Manage Integer collection
Clear values
Internalise data
Extract parameter



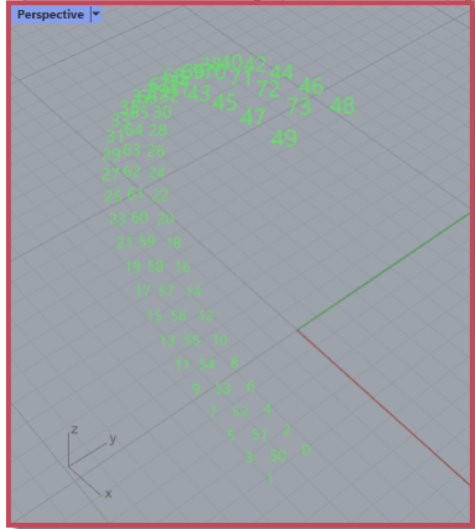
19

You will notice that this rotation has moved our **0** point past the **Y-Axis** (our *symmetry* line). If we were to *mirror* this we would have a duplication issue. If we remove this point we will be in good shape. It will also leave us with **24** points. Mirror that across the **Y-axis** and you will have **48**, just what we wanted.

Use **Cull Index** to remove an item from a list. Set **Indices** to **0**.



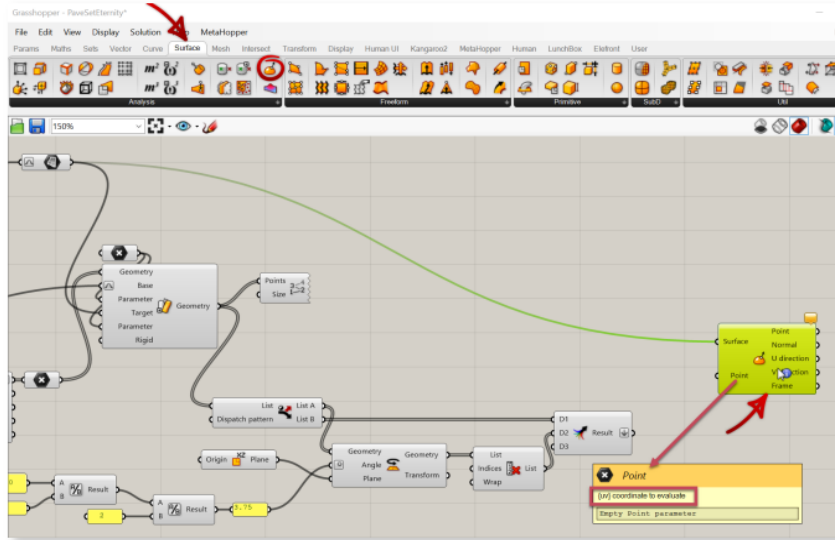
Merge
Merge a bunch of data streams



20

We need to use a **Merge** component to merge all data back together so we combine **List B** output with our latest "Culled list" data. We want to **flatten** the **Result** so we have a **Point List** that looks like above. Notice after *flattening* we have points **0-73**. This is *one less* than what we had.

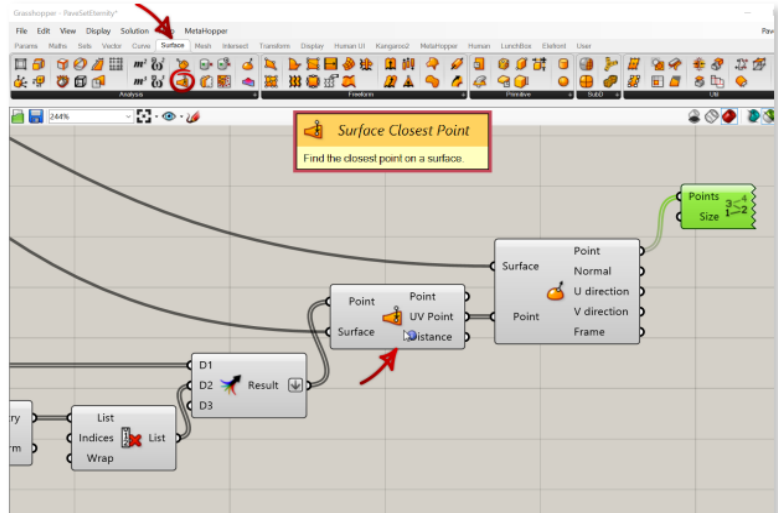
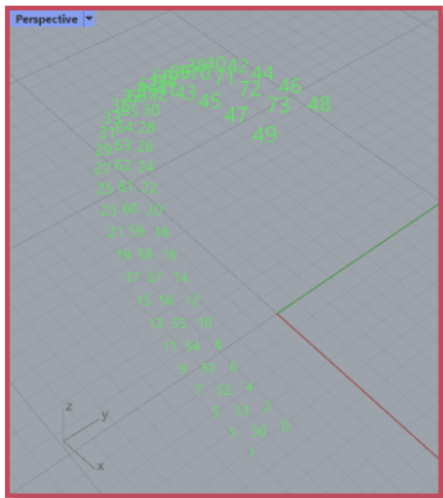
The Stone Layout



21

OK. We have our point locations set. The final thing we need to do for our point layout is prepare the *normal* directions for our stones or here specifically we will use the **Frame** output of another **Evaluate Surface** component.

Without this information all of our stones would be oriented in the same direction: *up*. Again we plug the *surface* from (step 3; pg 21). Our points need to be *UV coordinates* and not the *3-D coordinates* we currently have in our list. So...

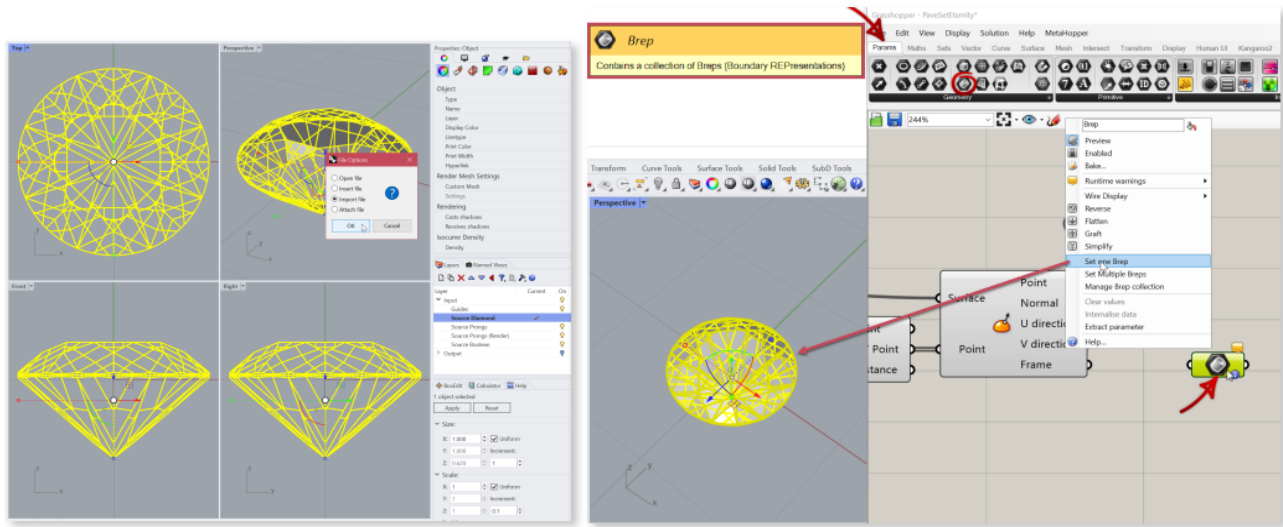


22

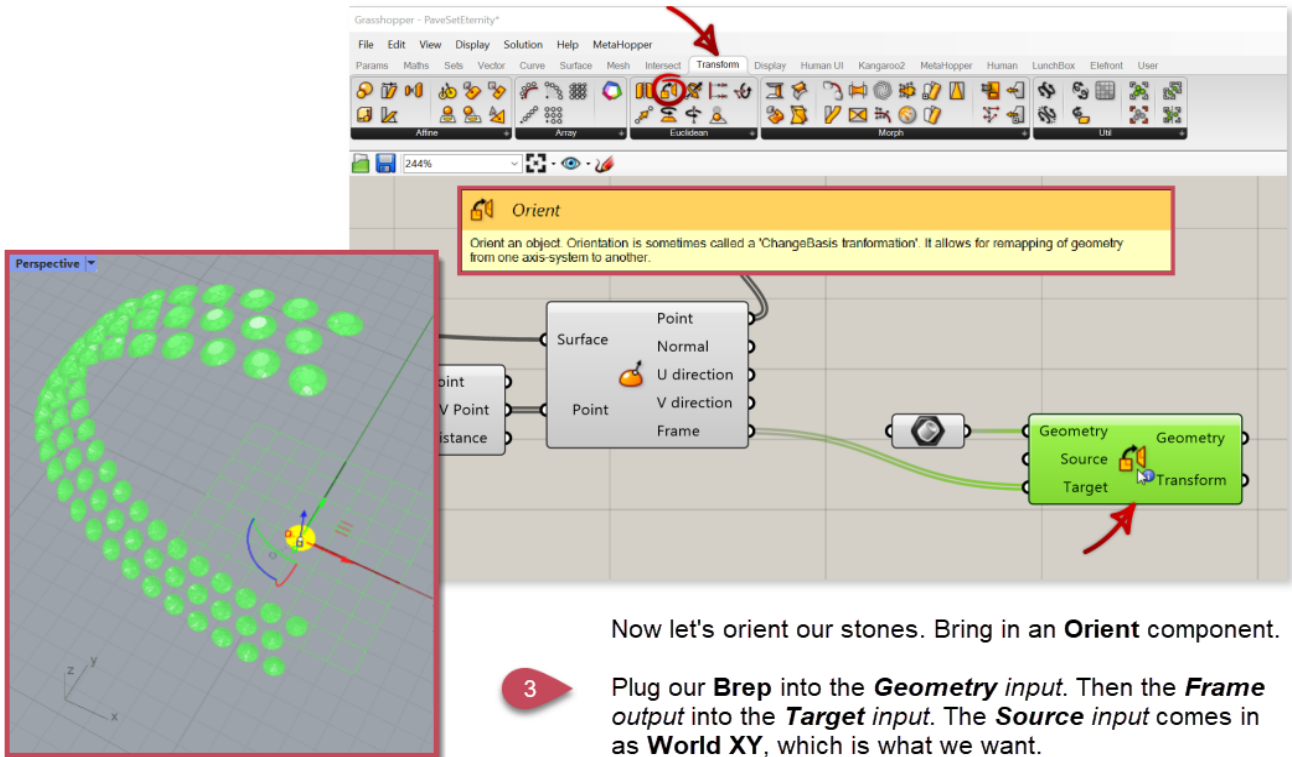
...we insert a **Surface Closest Point** component to get our *UV coordinates*. The **Surface** input is yet again the one from (step 3; pg 21). Plug our *merged point Result* into the **Point** input.

Now take the **UV Point** output and plug into the **Point** input of the **Evaluate Surface** component. Use a **Point List** to check that we are on track.

Arranging the Stones



- 1 We use Rhino to bring in or create the *round stone source, prongs, and cutters*. Start by *importing your favorite 1mm round stone*. Bring it into our **Source Diamond Layer**.
- 2 With our stone *selected* we are going to use the **Brep** component and right-click to **Set one Brep**.



Now let's orient our stones. Bring in an **Orient** component.

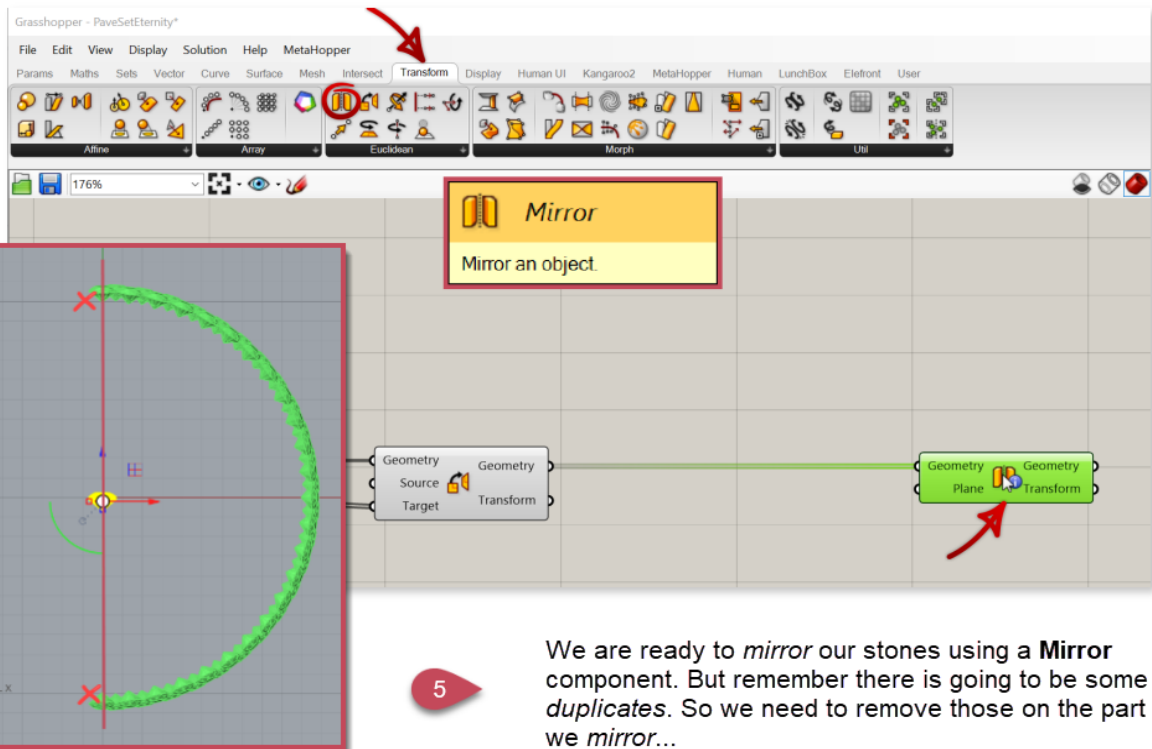
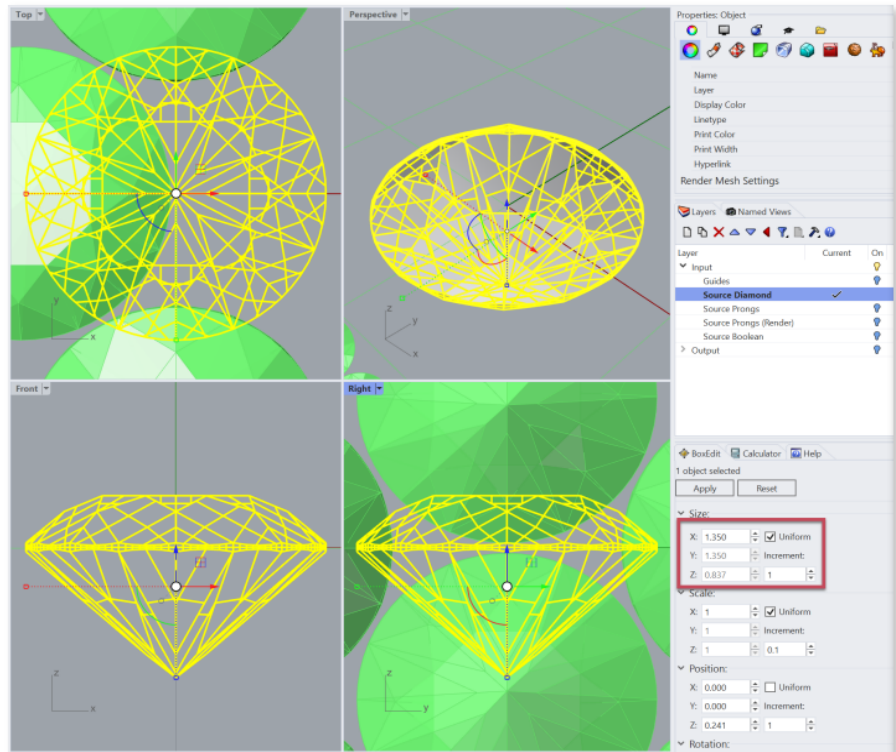
- 3 Plug our **Brep** into the **Geometry** input. Then the **Frame** output into the **Target** input. The **Source** input comes in as **World XY**, which is what we want.

Arranging the Stones

Remember we are using 1.35mm stones so go ahead and change the size and do your best to keep its *girdle* on the axis.

4

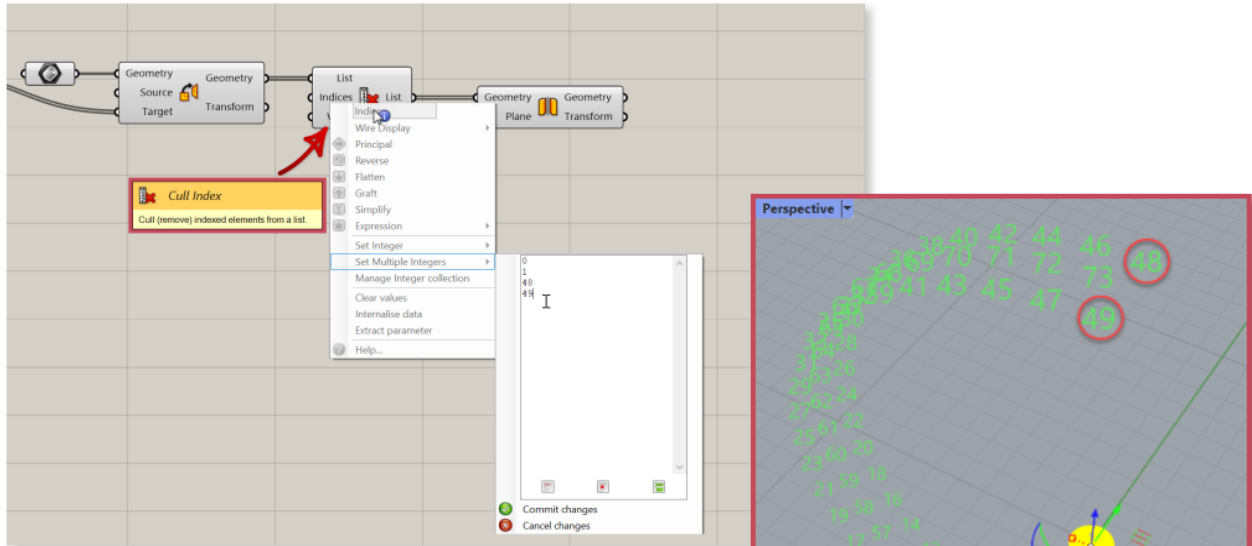
I like to use the **BoxEdit** panel for this. Notice we can make changes in Rhino (to our geometry that was assigned as a *Brep* in **GH**) and the *preview* will update. This comes in very handy as we arrange our stones.



5

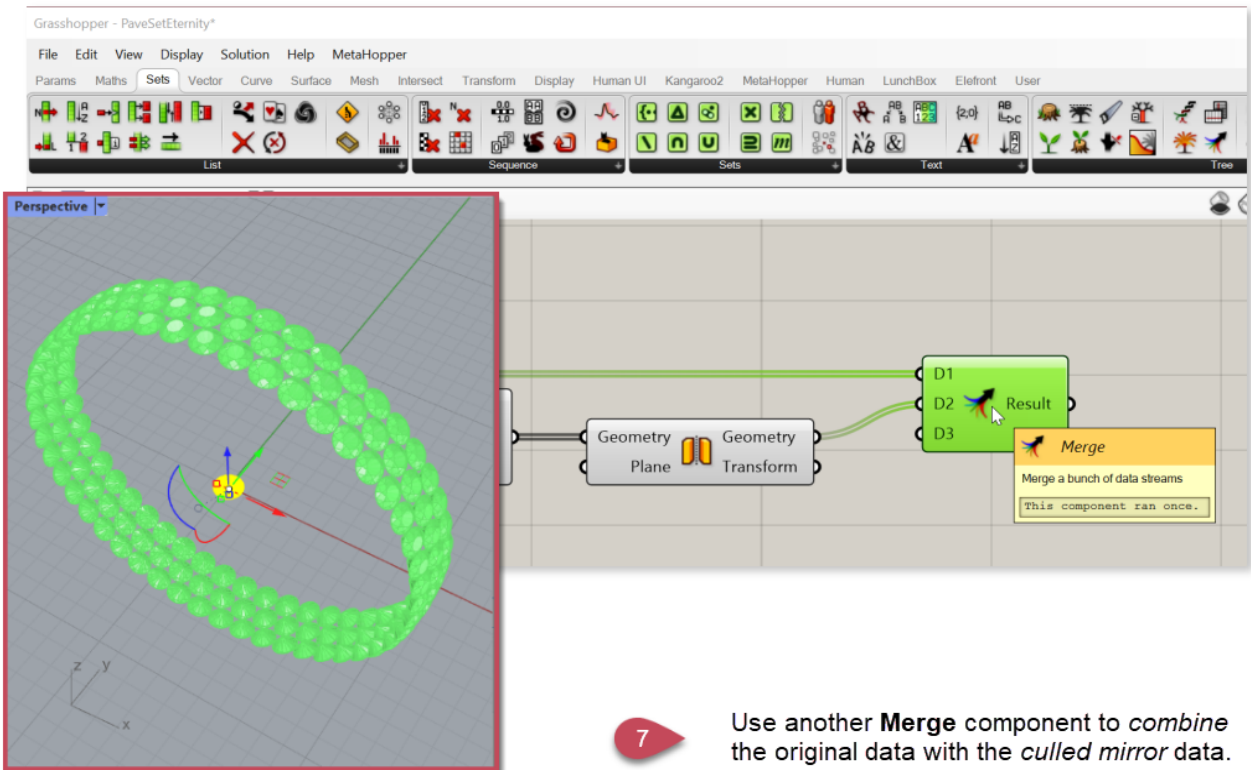
We are ready to *mirror* our stones using a **Mirror** component. But remember there is going to be some *duplicates*. So we need to remove those on the part we *mirror*...

Arranging the Stones



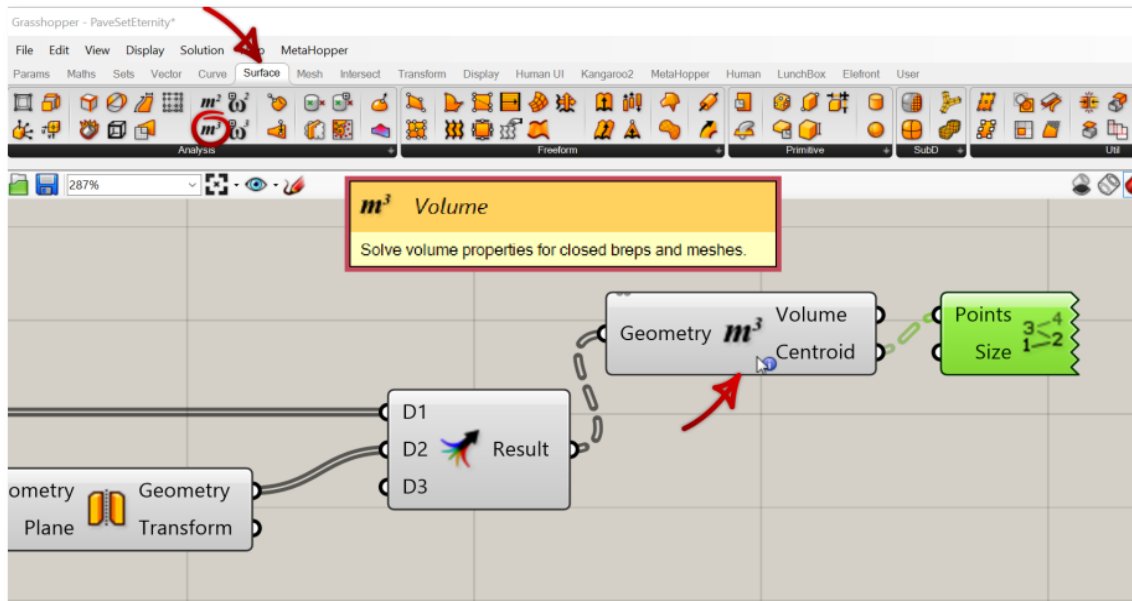
...using a **Cull Index** component again. This time we will remove *multiple* data points.

- 6 Using the **Point List** we set up previously we can target the points to remove (*cull*). Type in **0**, **1**, **48**, and **49** on separate lines after you right-click to access **Set Multiple Integers**.



- 7 Use another **Merge** component to *combine* the original data with the *culled mirror* data.

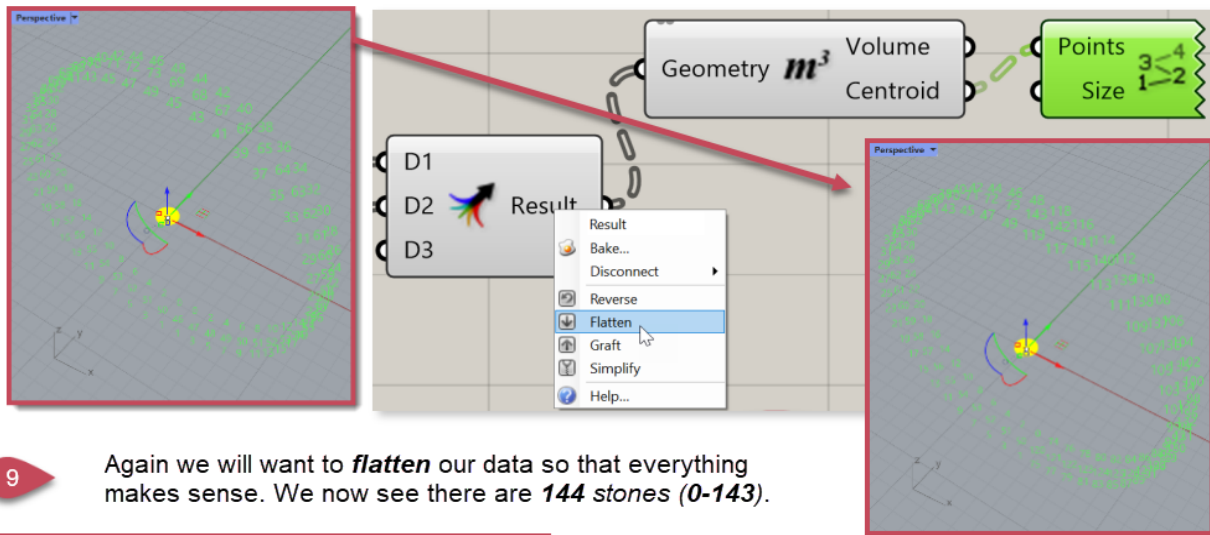
Arranging the Stones



Now, say you want to check your stone count. At this point, we have solid geometry and not the points we had been accustomed to. So how can we use our favorite **Point List** component?

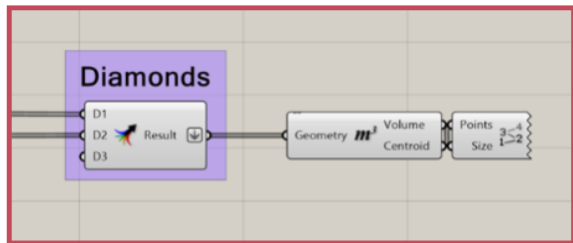
8

I like to use a component called **Volume** that, besides its *volume* information, will give us a **Centroid**. These are the *volume center points* that can now be referenced by the **Point List**.



9

Again we will want to **flatten** our data so that everything makes sense. We now see there are **144 stones (0-143)**.



10

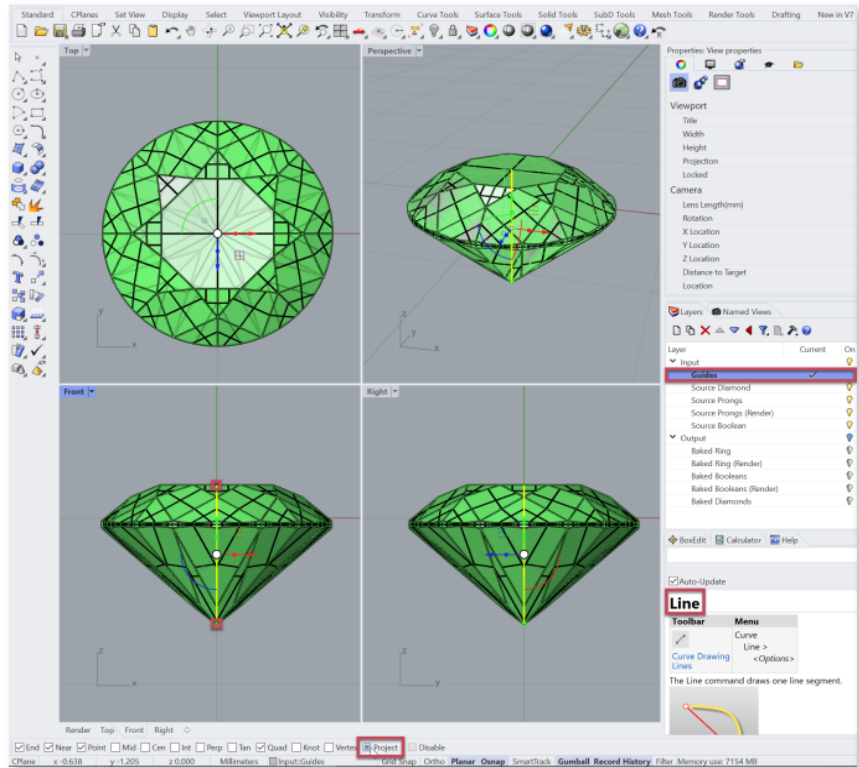
I like to highlight my *bake-able* components so I can quickly find them. So I use the **Scribble** component for a *label* and *group* with the **Merge** component.

Now, on to the next section...

Creating the Prongs

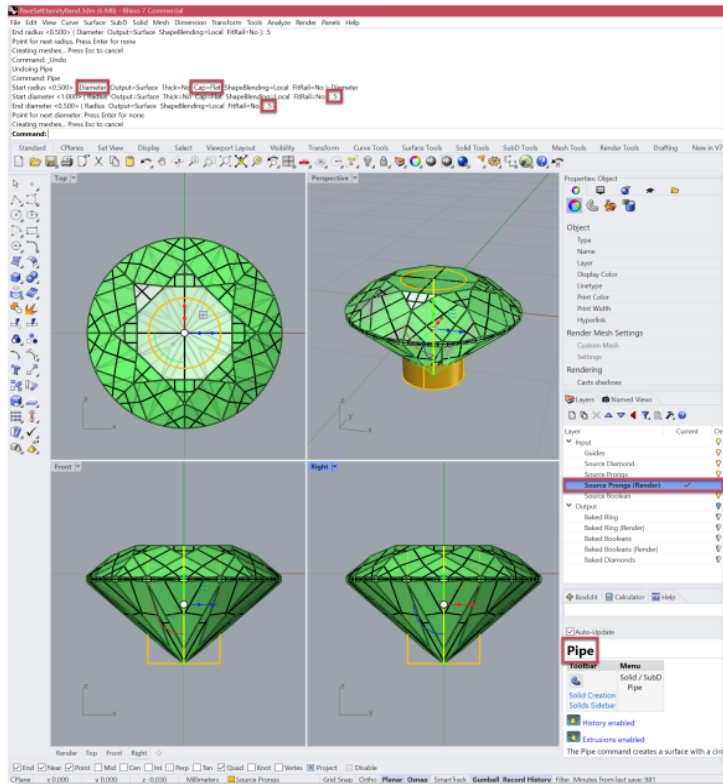
1

Back in Rhino, we will use **Osnaps** with **Project** enabled. Make the **Guides** Layer the **current** layer. Create a **Line** from our stones table to culet.



2

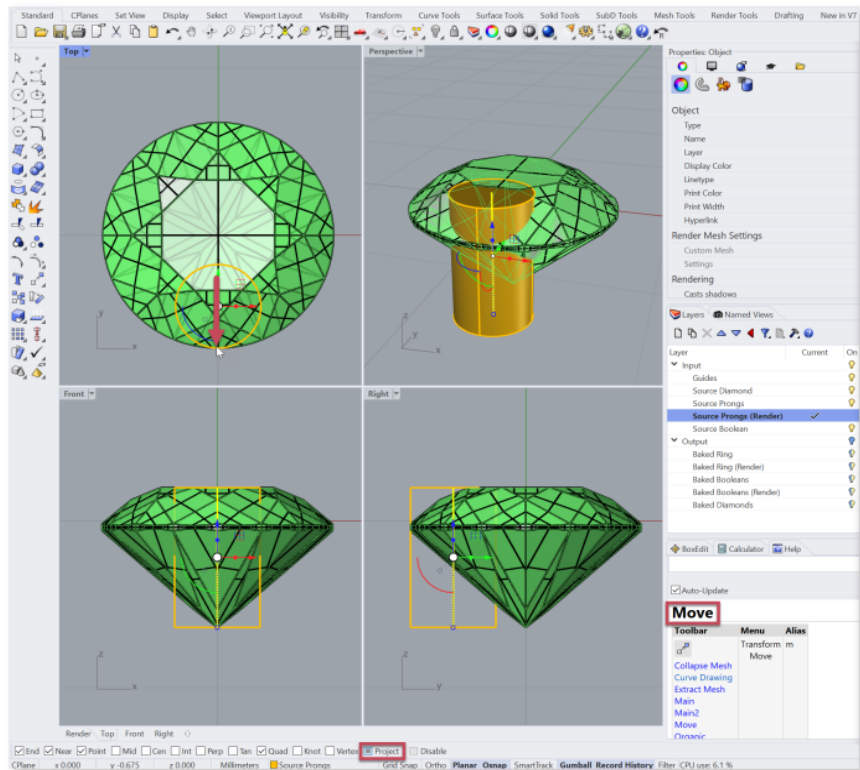
Make **Source Prongs (Render)** Layer **current**. Use the **Pipe** command with **Cap=Flat** and a **.5mm Diameter for Start and End**.



Creating the Prongs

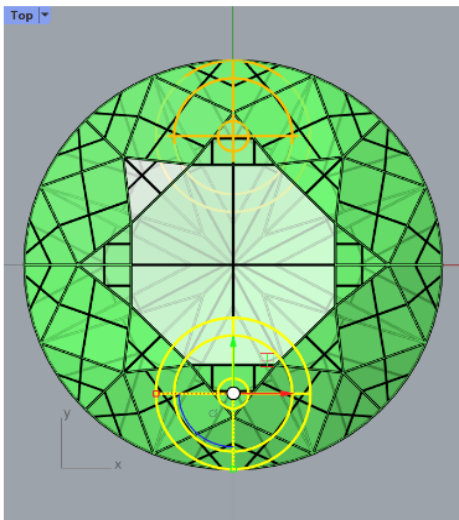
3

Move the prong forward (-Y direction) so that it snaps to the front of our stone's girdle. Use **Projected Osnaps Quad and End**.



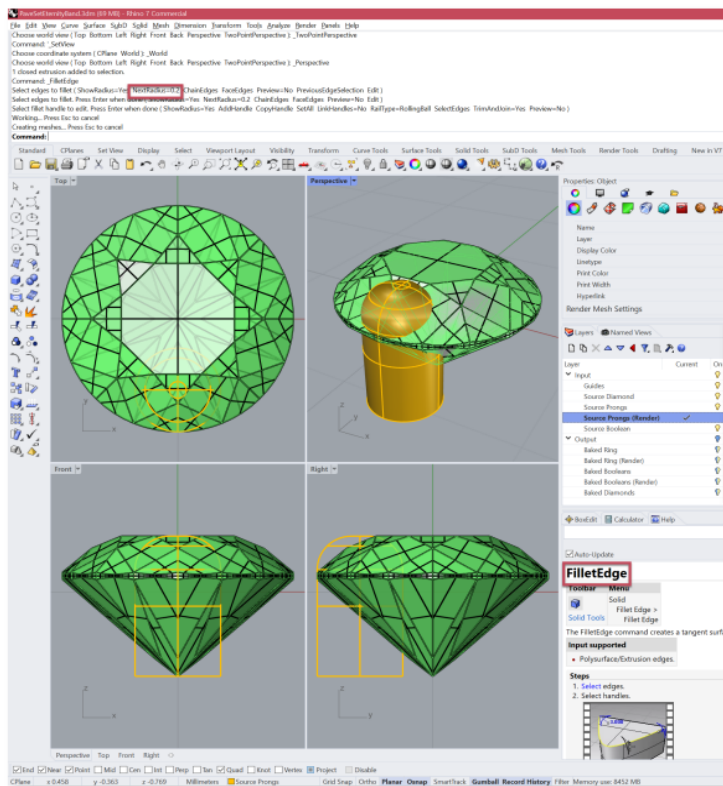
4

Fillet the **top edge** with a **Radius of 0.2** with the **FilletEdge** command.

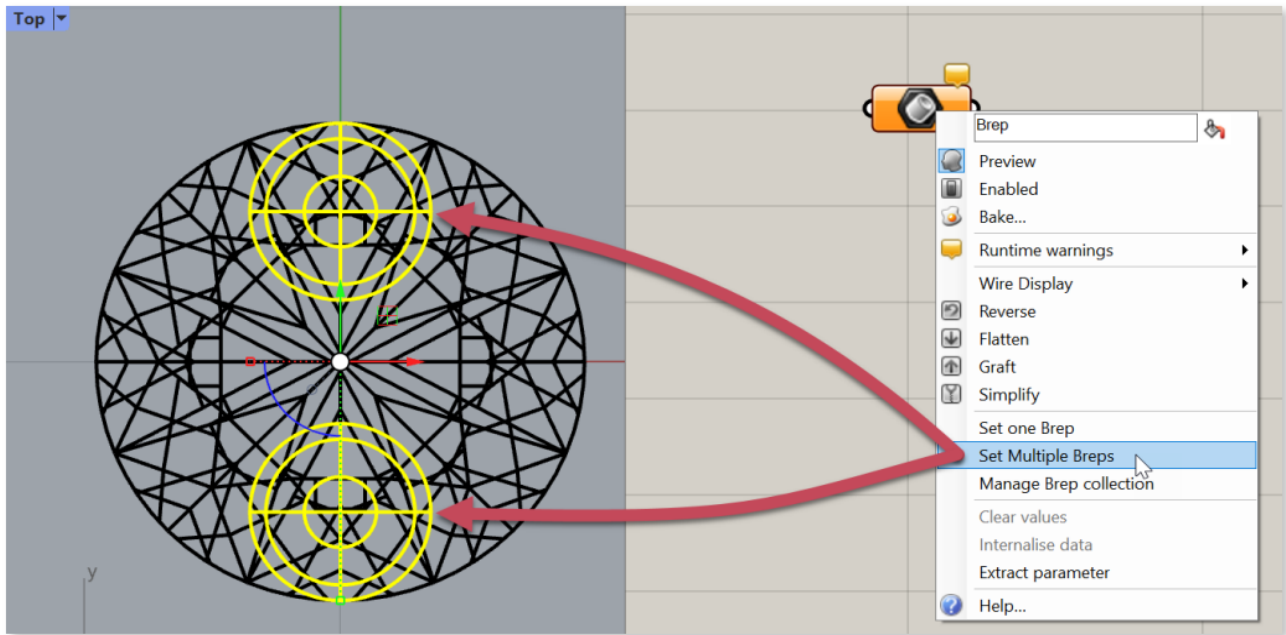


5

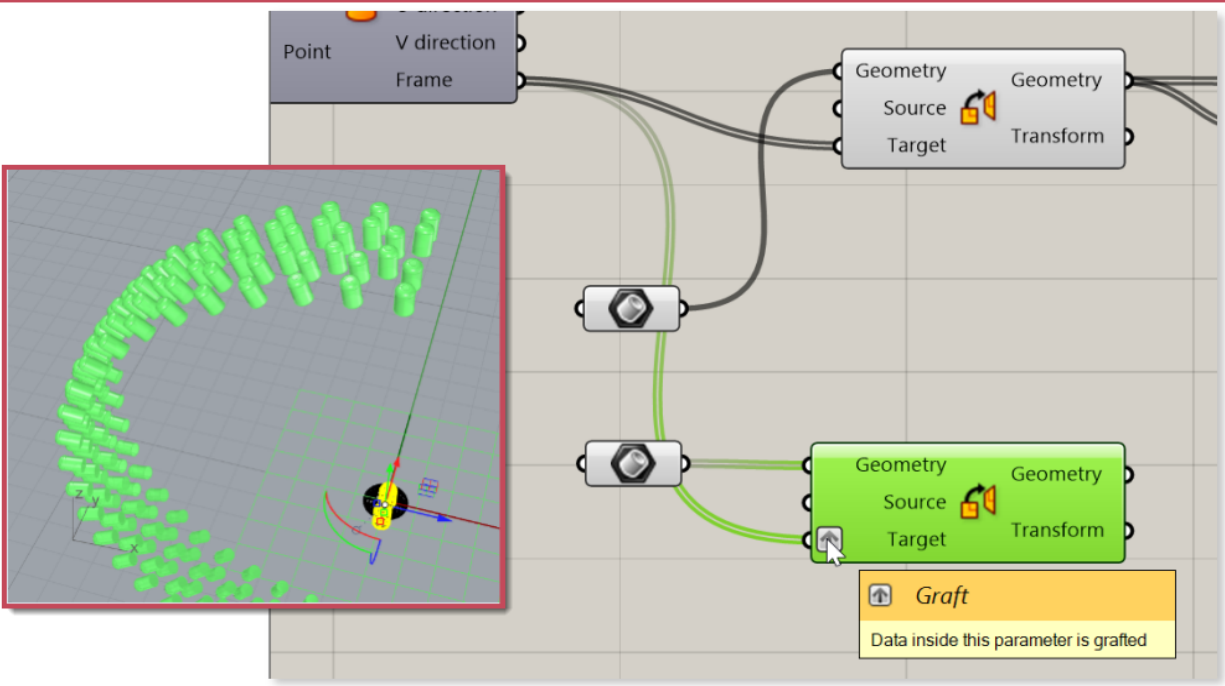
Mirror over the **X-Axis** with **Record History** enabled.



Creating the Prongs

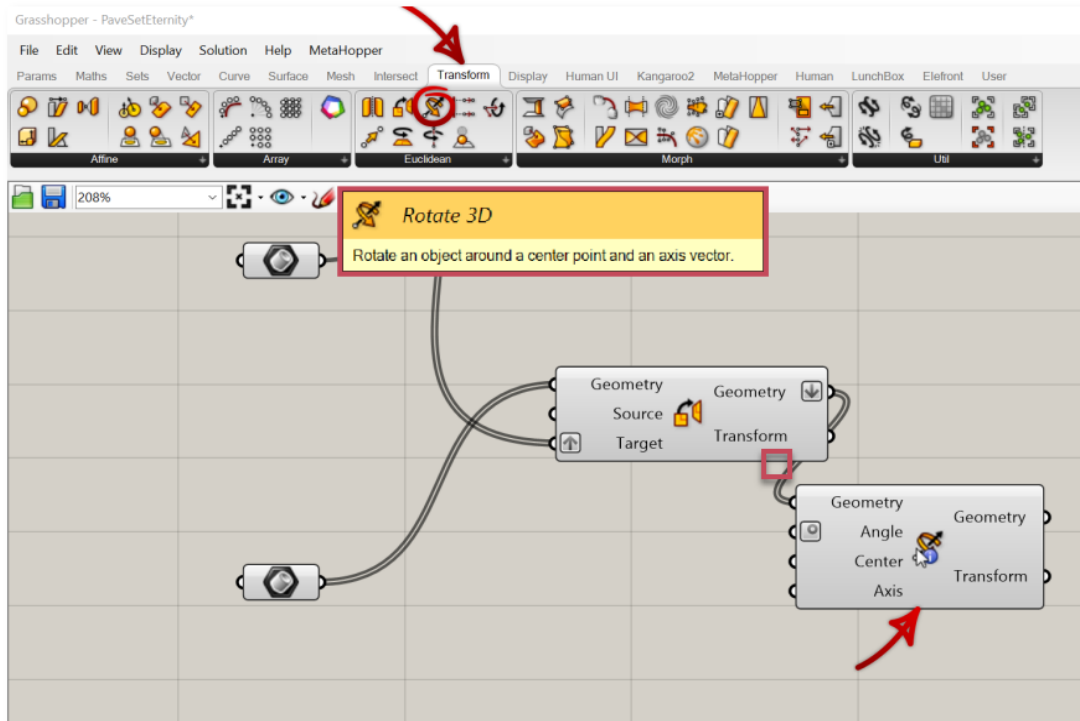


6 In GH use another **Brep** component and this time select **Set Multiple Breps** by right-clicking.



7 Copy the **Orient** component from your *Diamond Brep*. Plug in the *Prongs Brep*. Because we have **Multiple Breps** we will want to **Graft** the incoming **Target** data. This should give us a result like the above illustration.

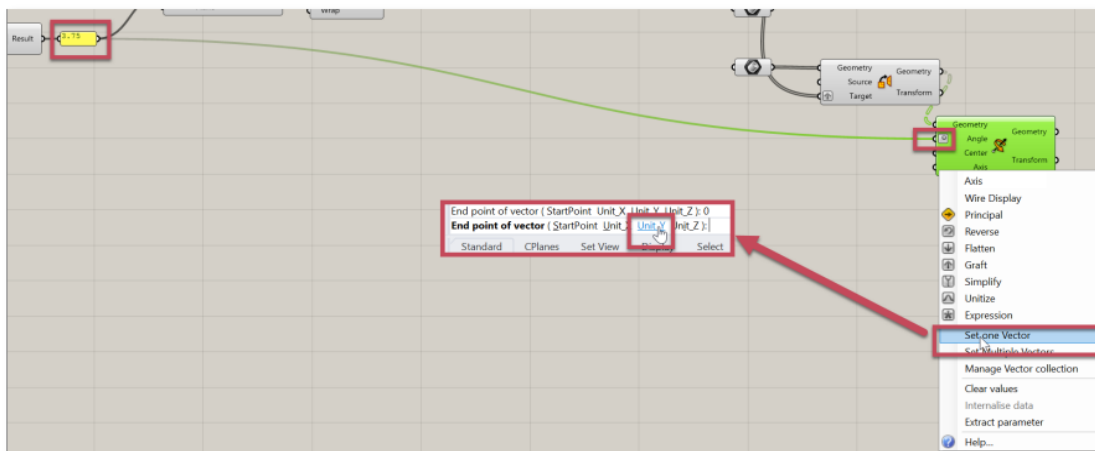
Creating the Prongs



8

Now we need to shift our prongs over so that they surround our diamonds. To avoid distortion we should do this **GH** and not Rhino. Like our stone, we built them centered on the **Y-Axis** so when we rotate them in **GH** they should always remain properly aligned with our *surface normals*.

Use the **Rotate 3D** component and set your **Angle input** to **Degrees**.



9

Right-click on the **Axis** input to **Set one Vector**. This time rather than drawing a line, use the short cut and select **Unit Y** in the Rhino *command line*.

We want to *rotate* our prongs the same amount as our middle row of diamonds. So feed the previous **Result** into our **Angle** input. Again, be sure **Degrees** is set by right-clicking the **Angle** input.

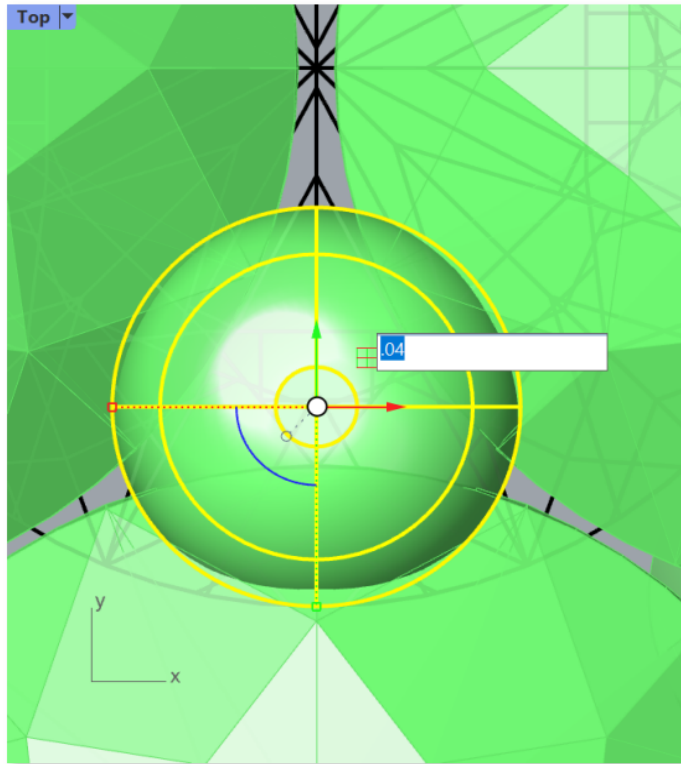
Creating the Prongs

10

Preview both the **Rotate 3D** component and the **Diamond Merge** component.

Now we can use the **Top Viewport** in Rhino to center our prongs. Here I moved the original (*parent*) prong **.04mm** toward the *origin*.

Its *child* should update (also in toward the *origin*). As we make further adjustments we may adjust their location again.

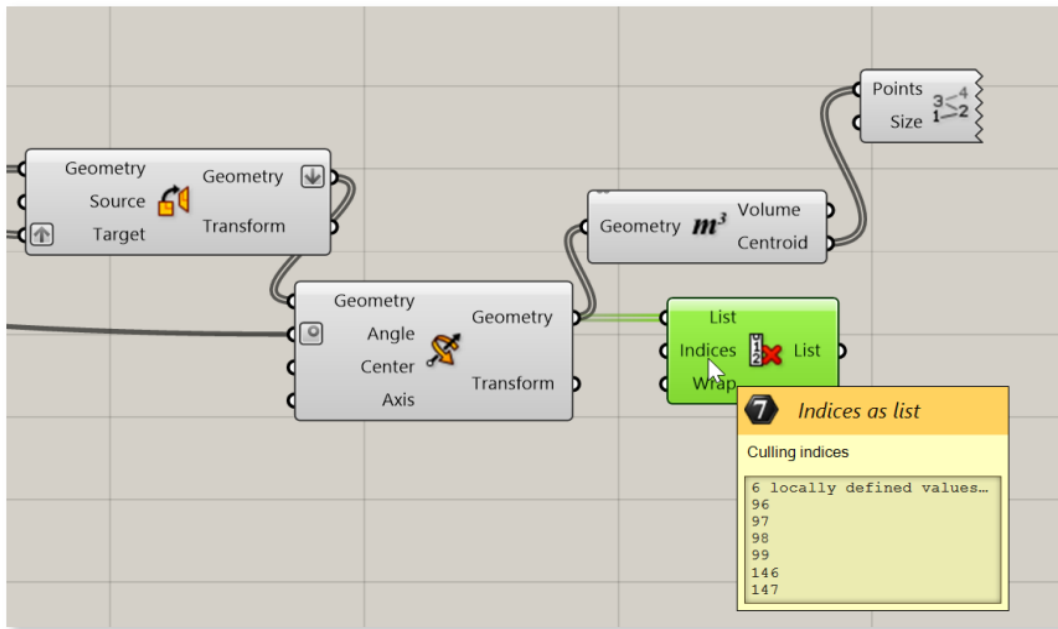


11

As we did with the diamonds we need to see which prongs will be repeated when mirrored. Then we will *cull* those as we did before. We have *twice* as many items and it would help to see a **Point List**.

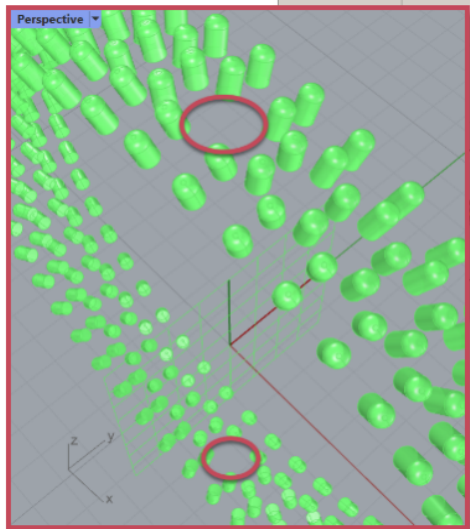
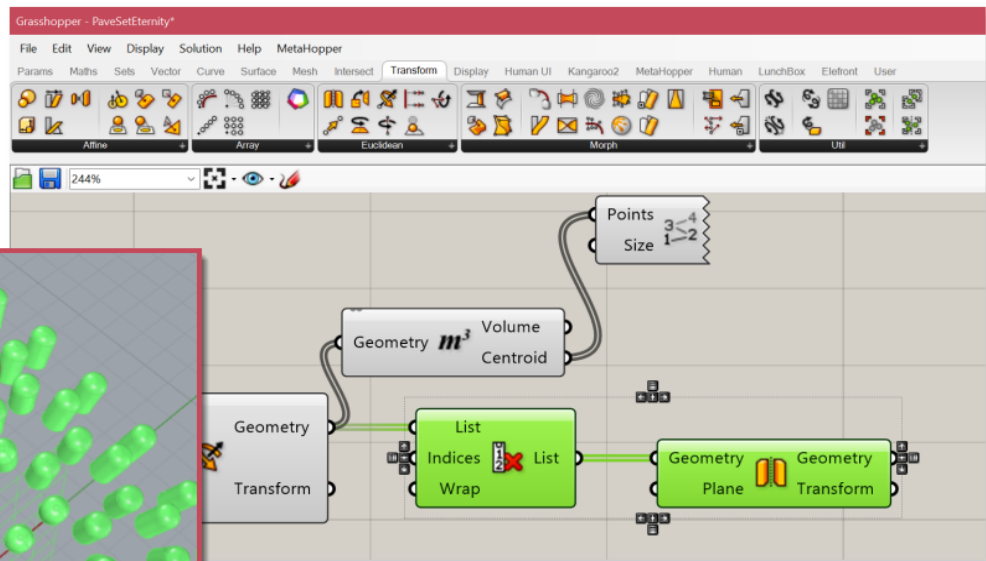
Use the **Centroid** output from a **Volume** component to do this. I see I have six beads that I do not want to mirror...

Creating the Prongs



12

...so use a **Cull Index** component and set the **Indices** input accordingly. Remove items 96, 97, 98, 99, 146, and 147.



13

Plug in our **List** to the **Geometry** input of another **Mirror** component (**YZ Plane**).

With our last two components *previewed* we see that we are missing four prongs. Two (146,147) we just removed. The two on the bottom will be a *mirror* image of those.

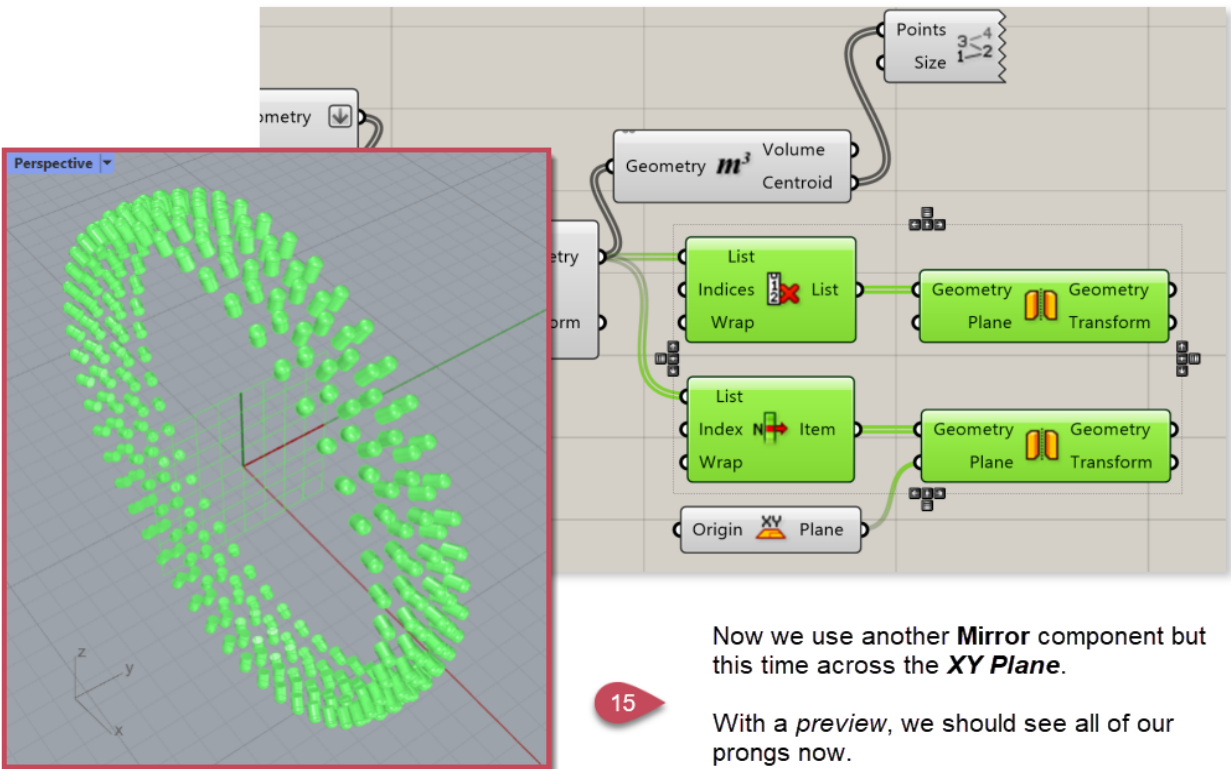
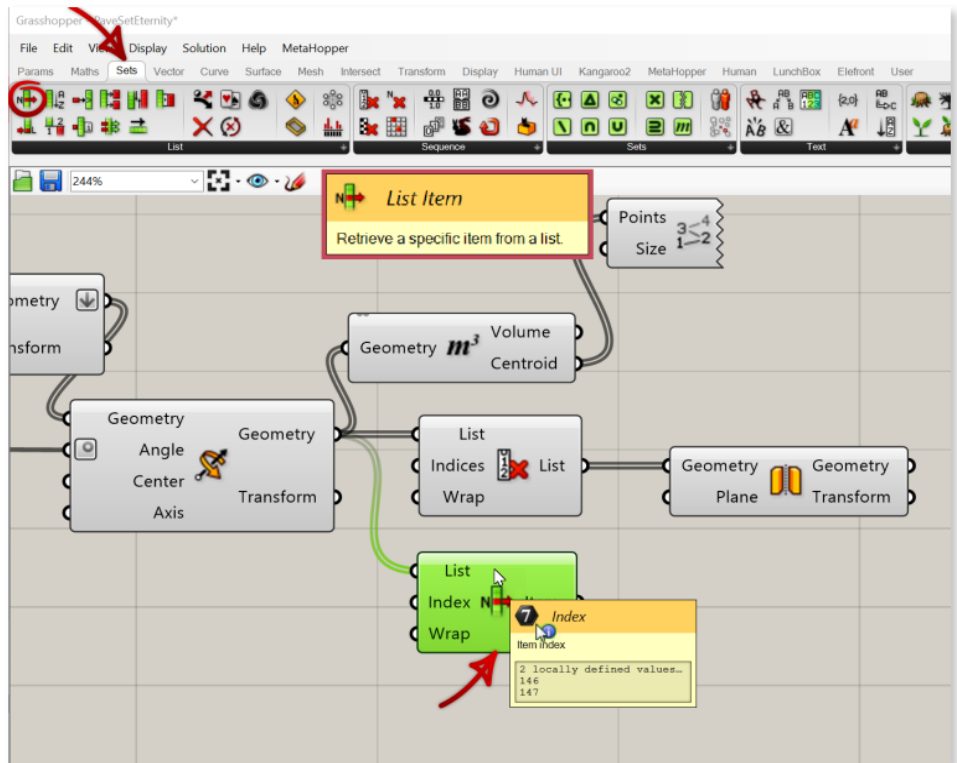
Creating the Prongs

We use a **List Item** component to retrieve certain items.

Here we want items 146 and 147.

14

Right-click on the **Index** input and enter those values in the **Set Multiple Integers** field.

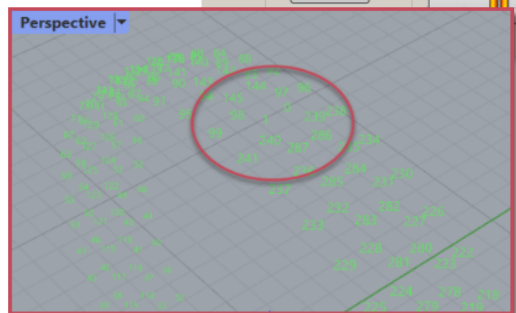
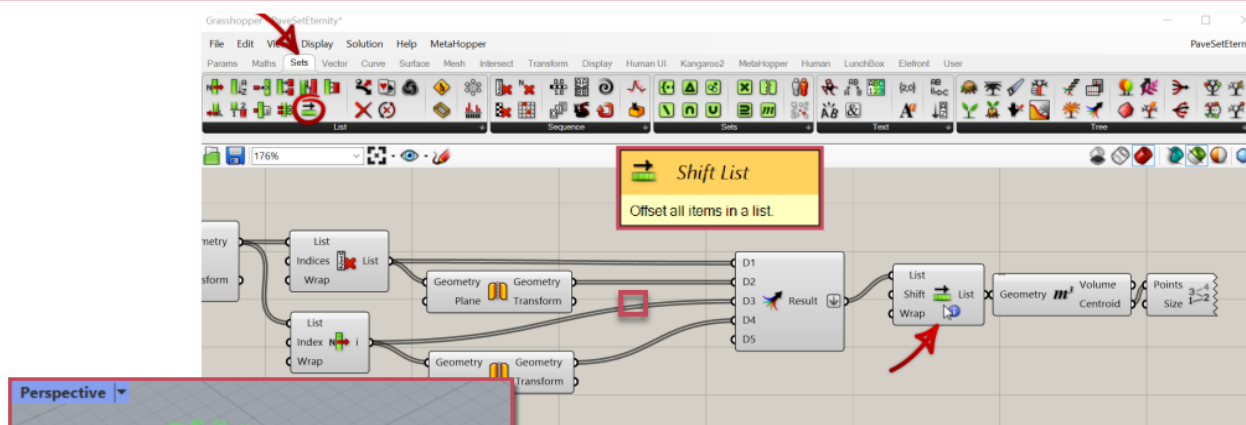


15

Now we use another **Mirror** component but this time across the **XY Plane**.

With a *preview*, we should see all of our prongs now.

Creating the Prongs

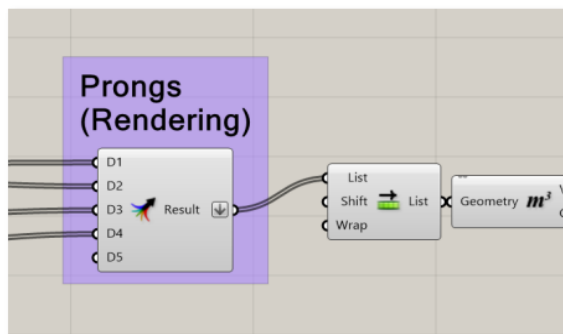
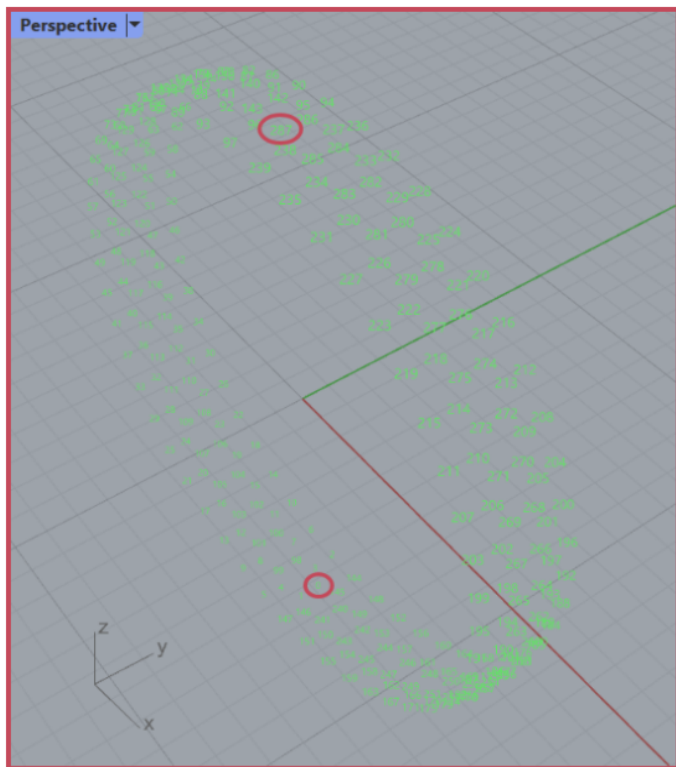


- 16 Next we **Merge** four data streams together. Our **Culled List** goes into **D1**. Its mirror goes into **D2**. The **two prongs** from the **List Item** component are for **D3**. Finally, their **mirror** goes into **D4**. **Flatten** the **Result**.
- 17 Once again, use a **Centroid** output from the **Volume** component to get a **Point List**.

You may notice that the preview has 0 and 1 out of sequence on the list. This is easily fixed with **Shift List** component. Enter a value of 2 for the **Shift** input.

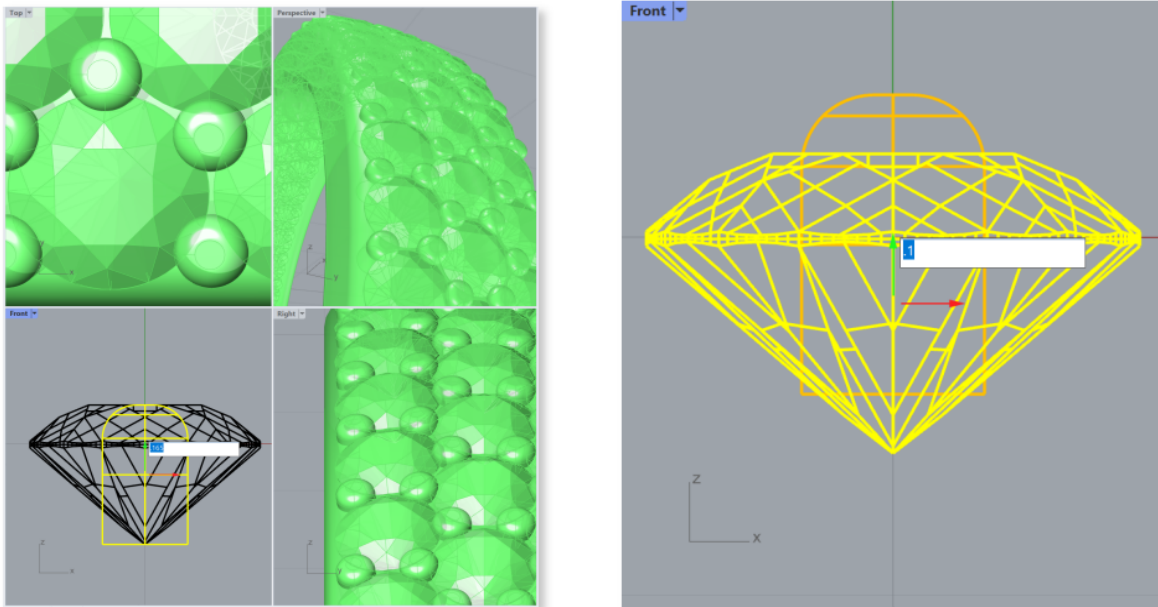
Now our list of numbers looks *correct*.

While this step was not necessary for this exercise it is good practice and might make a difference should we want to reference it later.



- 18 Let's clean up similarly to before with **Scribble** and **group**.

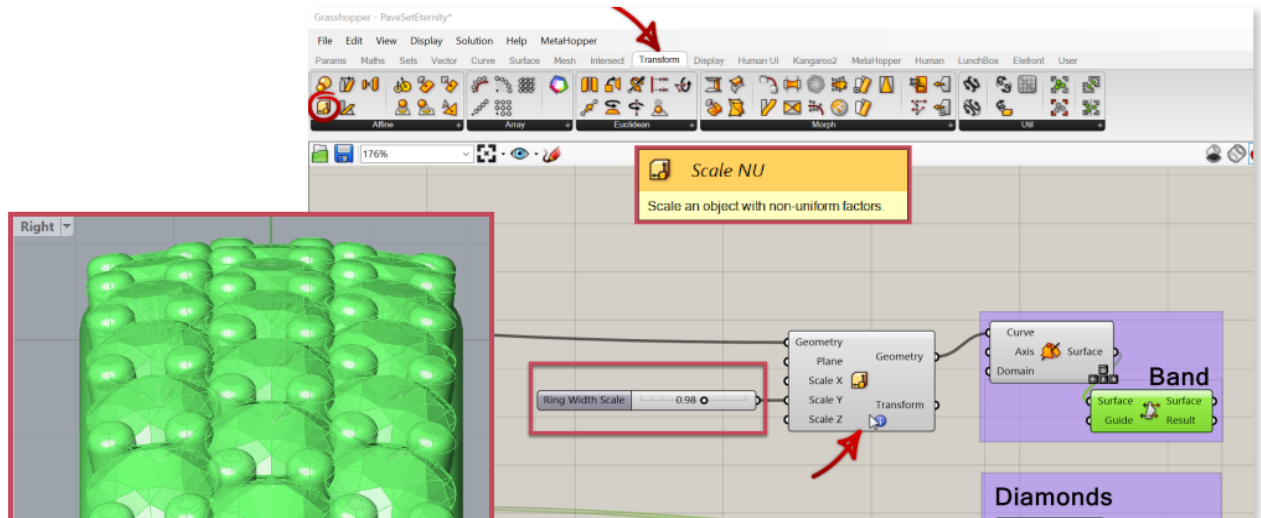
Adjusting the Parameters



Let's turn on our *preview* for the *Band*, *Diamonds*, and *Prongs*. Currently both the *diamonds* and the *prongs* are a bit low and need to come out of the *band's* surface further. This is easily accomplished in Rhino.

1

Move the *parent* prong up **.165mm**. Next move the stone up **.1mm**.



2

We can slim down our *band*, however we do not want to mess with our initial curves because that would affect other elements. So we can use a **Scale NU** component on the *profile* just before we create the *surface*.

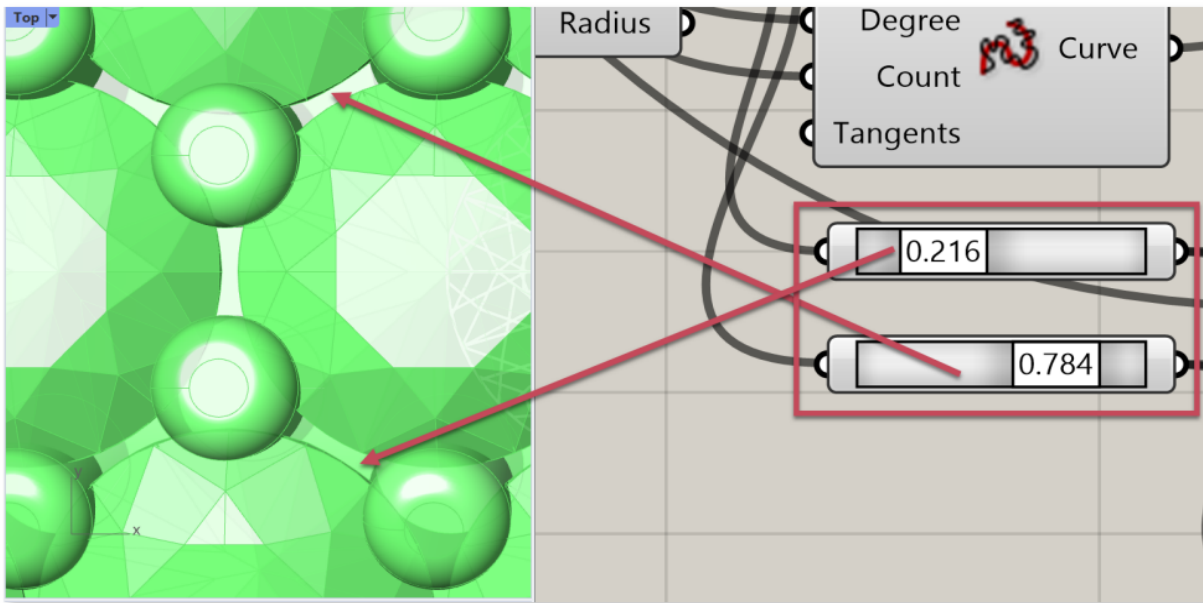
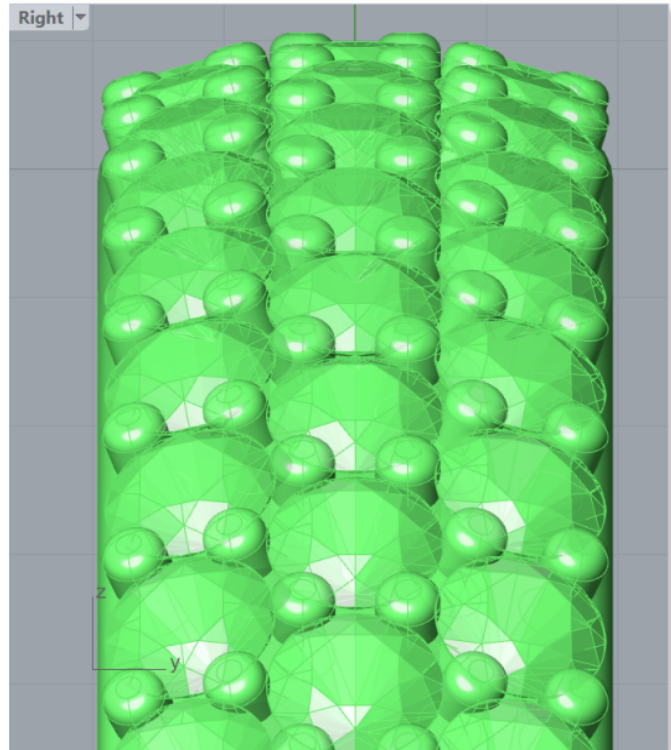
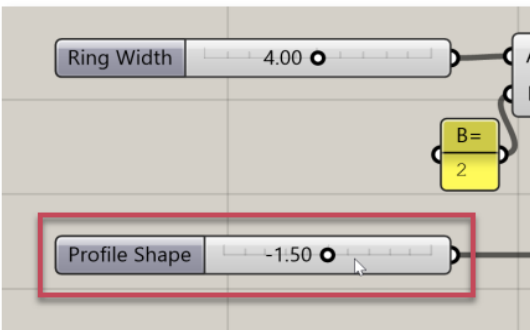
We only want to change the **Scale Y** input. I created an adjustable **Number Slider** with a domain from **.95**

Adjusting the Parameters

If we use our slider **Profile Shape** (found at the left side of our canvas) we can make the profile more rounded.

3 This will also affect the *normals* of our diamonds (changing their direction on the outside rows).

Here I change the value to **-1.50** and the result is more to my liking.



4 Revisit our last two **Point On Curve** components (previously set at .210 and .790). We can adjust the *gap* between the *middle row* and *outside rows*.

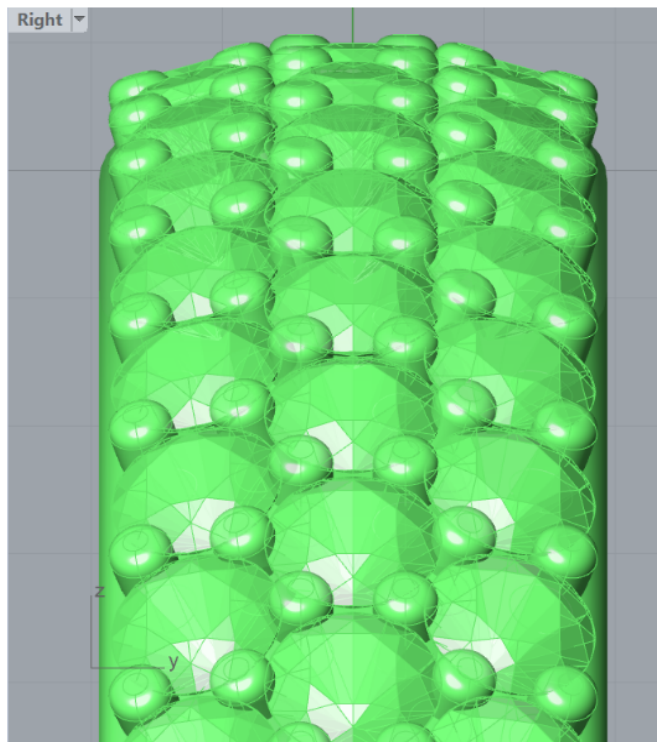
We want to get all the *gaps* as even as possible. Here I adjusted to **.216** and **.784** and they look much better.

Adjusting the Parameters

5

After changing the profile shape and adjusting the *gaps* we revisit the *band's width* again.

I tweaked it slightly - down to **0.97**.

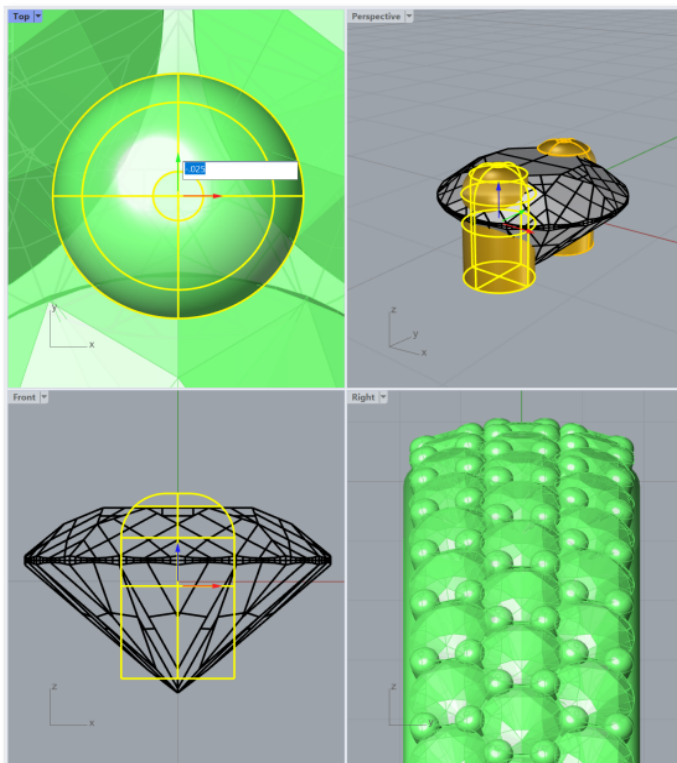


6

Now after those adjustments I need to do a final adjustment to our *prong* locations.

I move the *parent* prong in the *Y-direction* again.

Here I move it away from *origin* **-.025mm**. Be sure the *child* updates in the *opposite* direction.

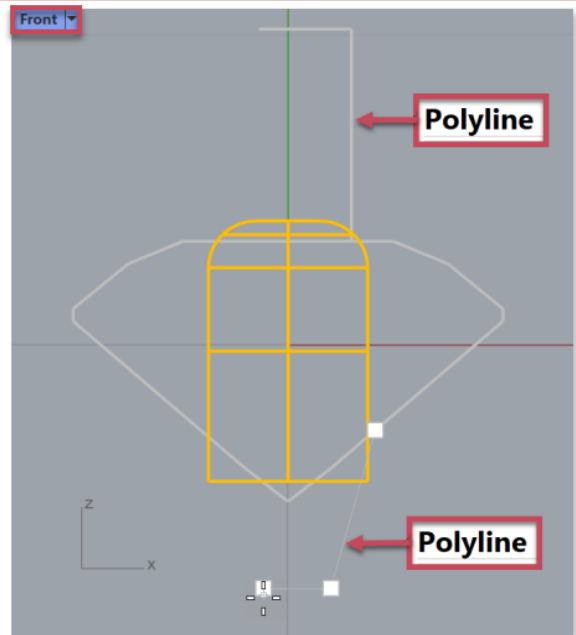
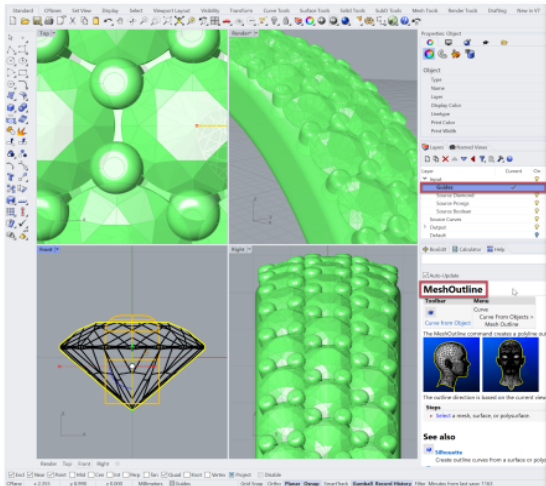


Creating Gem Cutters (for Rendering)

Next we will create some **Gem Cutters** to use for the purpose of *rendering* our model.

1

With the **Guides Layer current**, use the **MeshOutline** command and select our *stone* in the **Front Viewport**. This will create a *closed polyline* around the stone

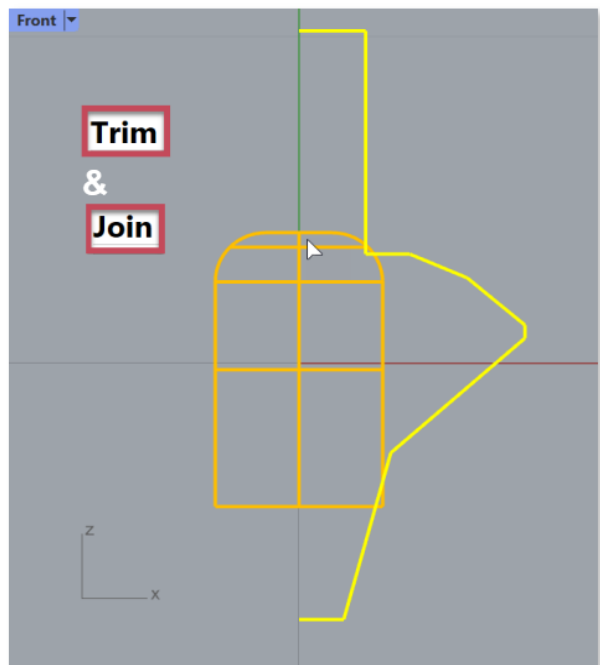
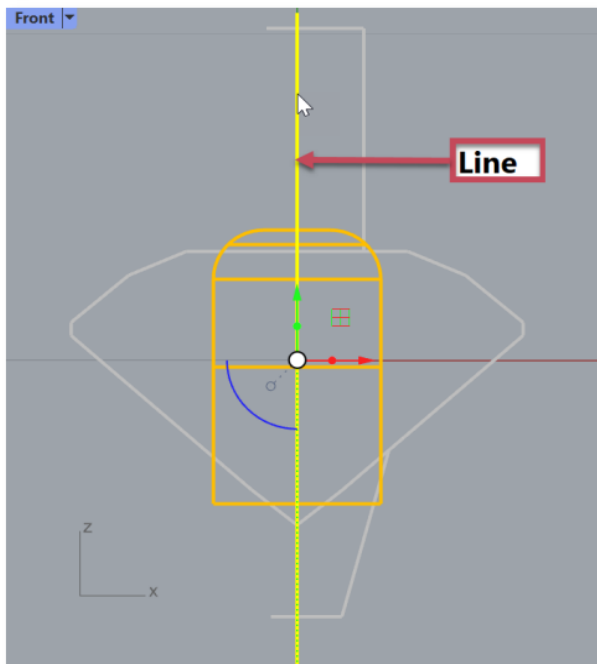


2

Next draw two *open Polylines* (also on the *CPlane* in the **Front Viewport**) as depicted above.

3

Create a "trim" **Line** at center.



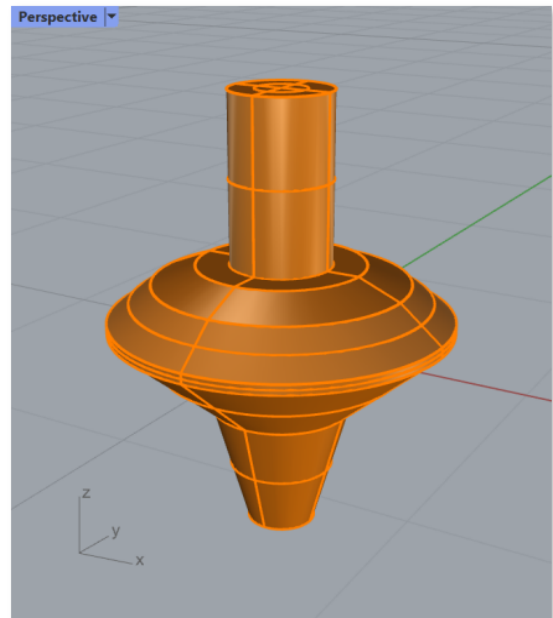
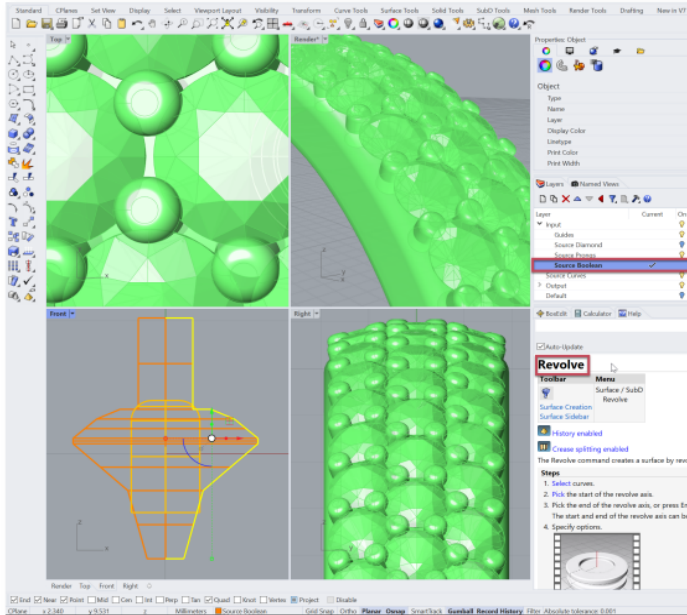
4

Then **Trim** and **Join** into one *open polyline*.

Creating Gem Cutters (for Rendering)

5

With **Source Boolean** Layer current, use the **Revolve** command to create a surface from the **polyline** we just created.

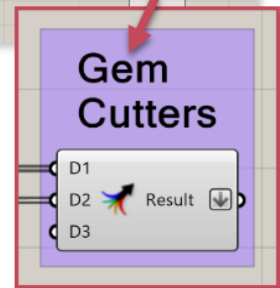
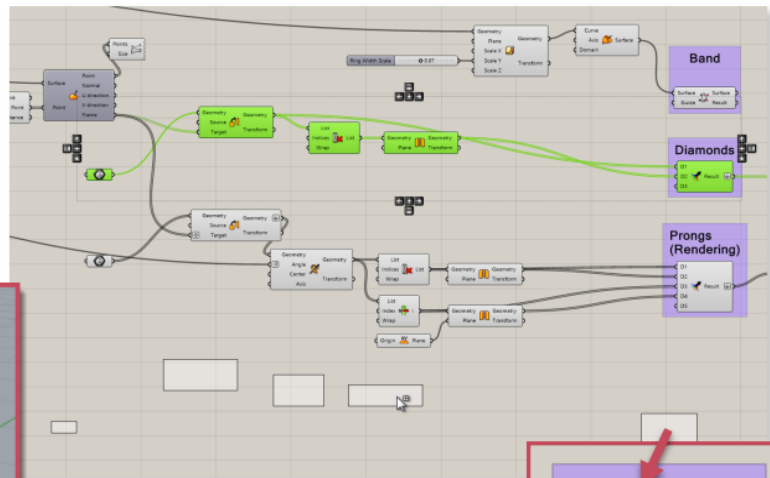
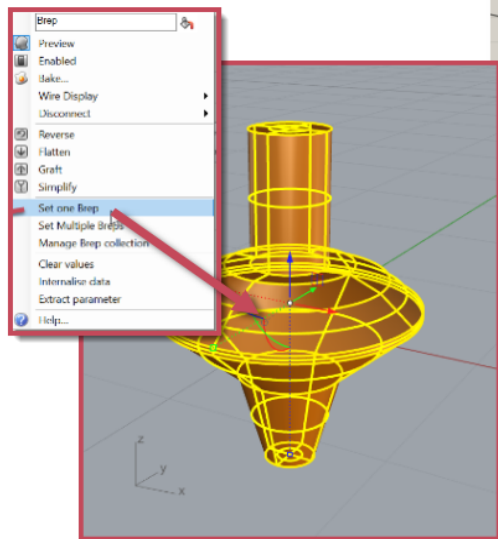


6

Copy the entire **Diamond** section (*highlighted in green*).

7

Next **Set one Brep** (*right-click*) as the **Revolved closed solid polysurface**.



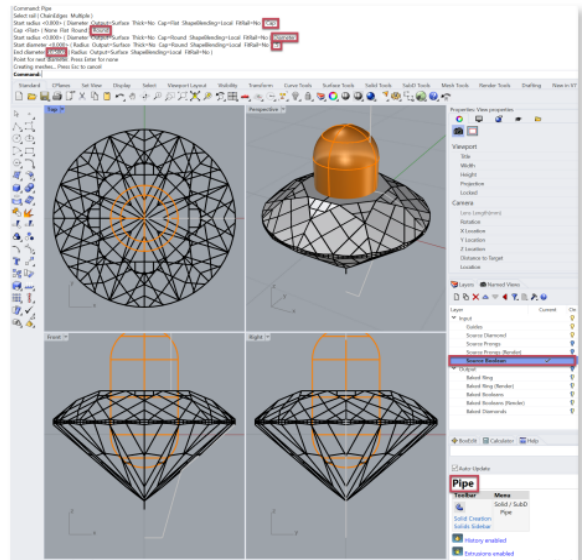
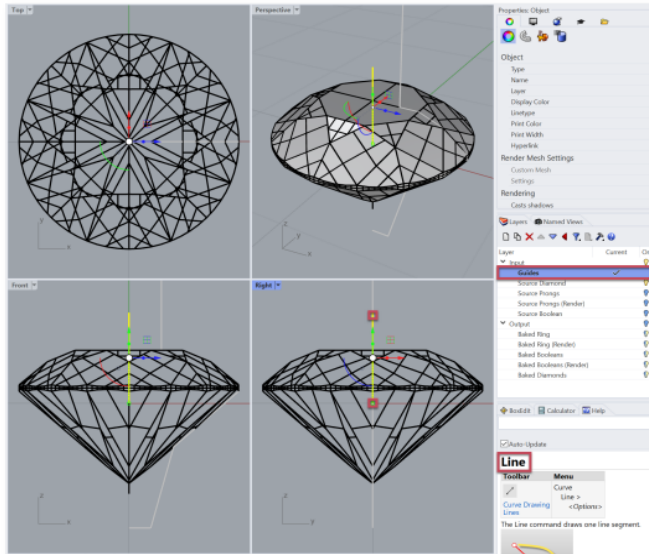
8

Group with a **Scribble** called **Gem Cutters**.

Creating Pilot Holes (for Production)

Our boolean cutters will be different for our production model. So let's go ahead and create those now. In Rhino (**Guides Layer**) create a *projected Line* starting from 0 to anywhere above our stone's table.

1



2

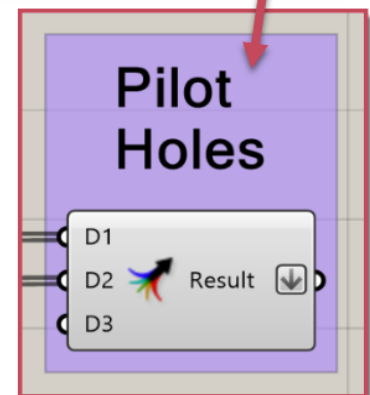
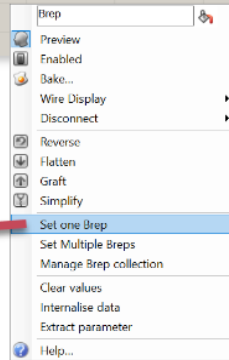
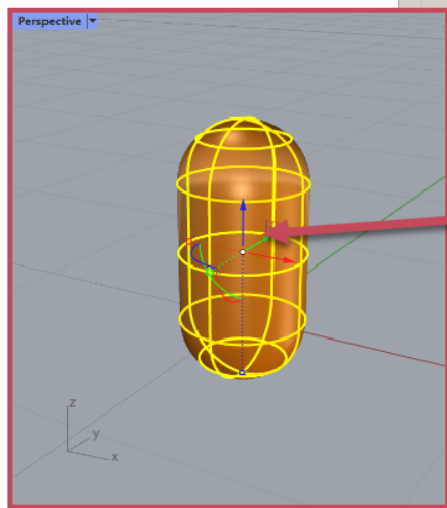
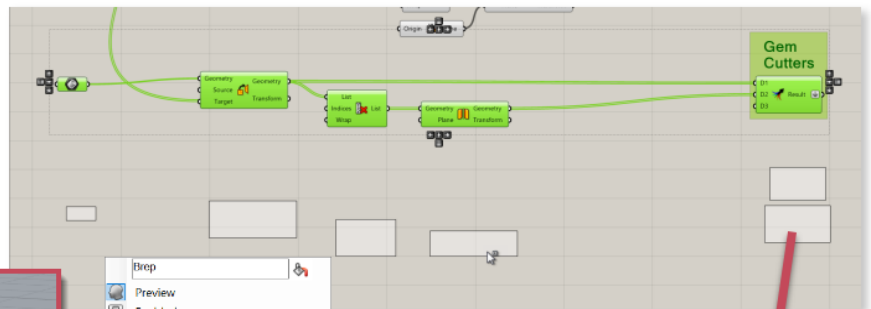
Next, in the **Source Boolean Layer**, we will run a **Pipe** command with a **Rounded Cap** and a **.5mm diameter** (at start and end).

Copy (*drag and tap Alt*) the entire **Gem Cutters** section.

3

Next **Set one Brep** (*right-click*) as the **Piped closed solid polysurface**.

4

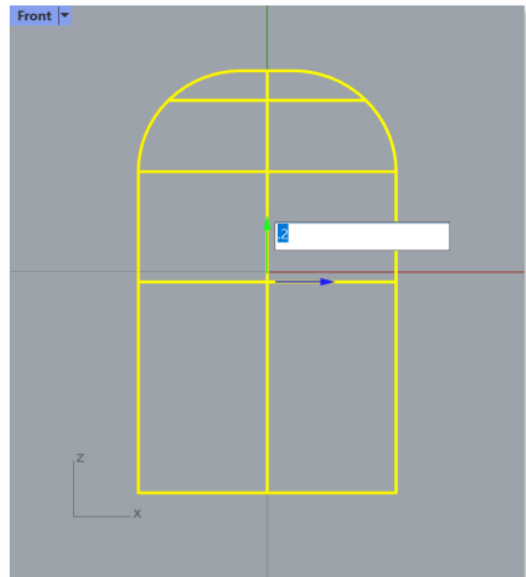
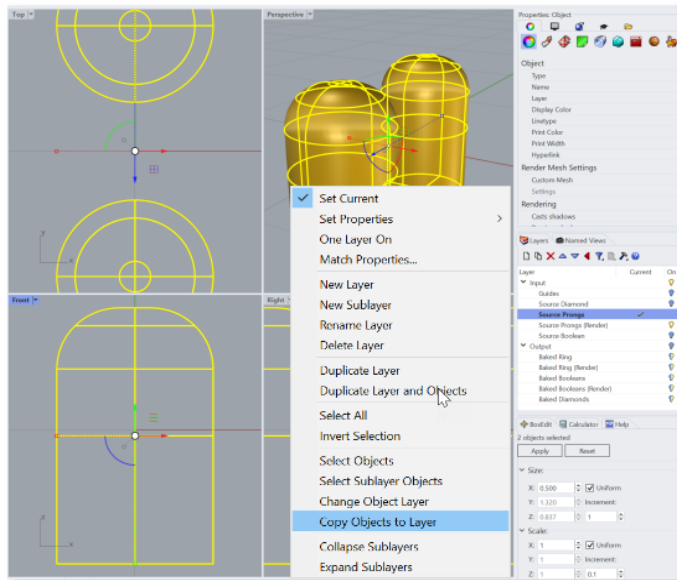


5

Group with a **Scribble** called **Pilot Holes**.

Creating Prongs (for Production)

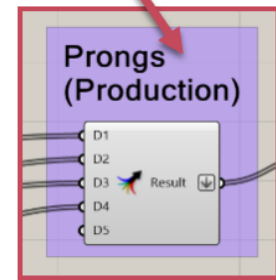
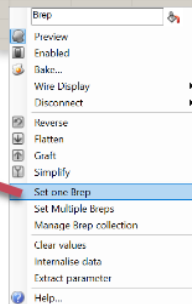
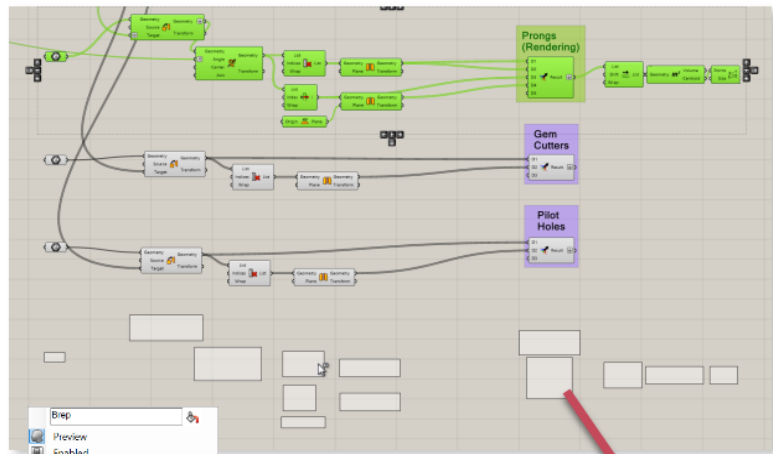
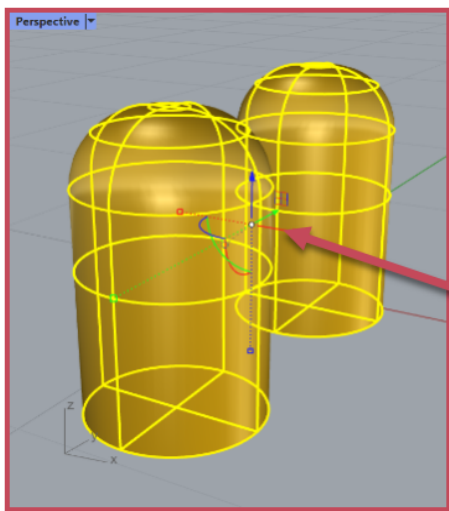
1 Select and copy both *Prong* objects into **Source Prongs** Layer.



2 These will be taller for production so move these copies *upward* .2mm.

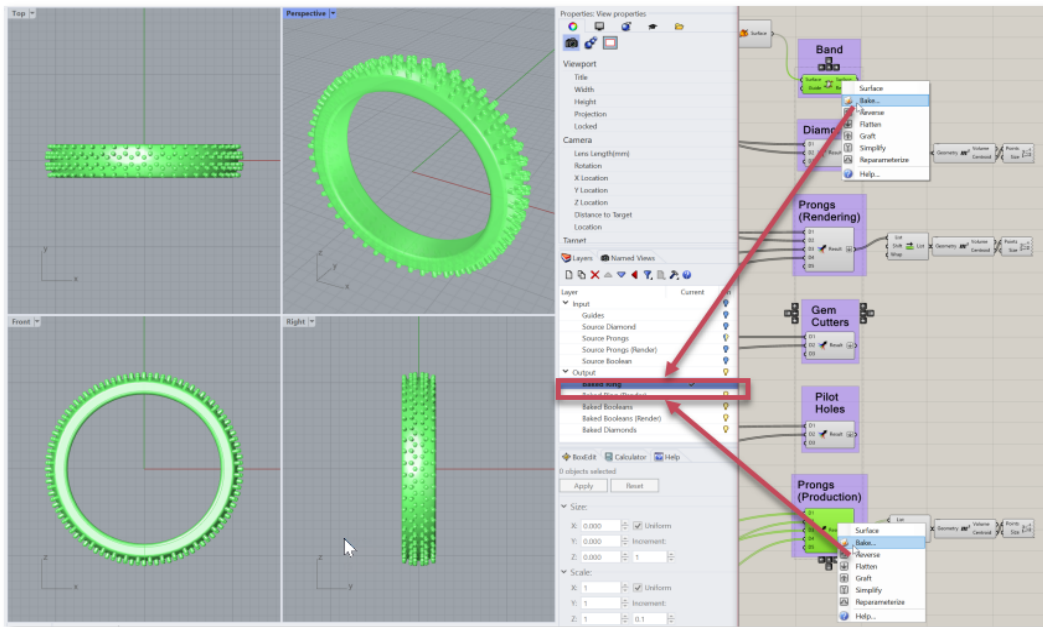
3 Copy (*drag* and *tap Alt*) the entire *Prongs* section (*highlighted in green above*).

4 Then **Set one Brep** (*right-click*) as the newly moved prongs in **Source Prongs** Layer.



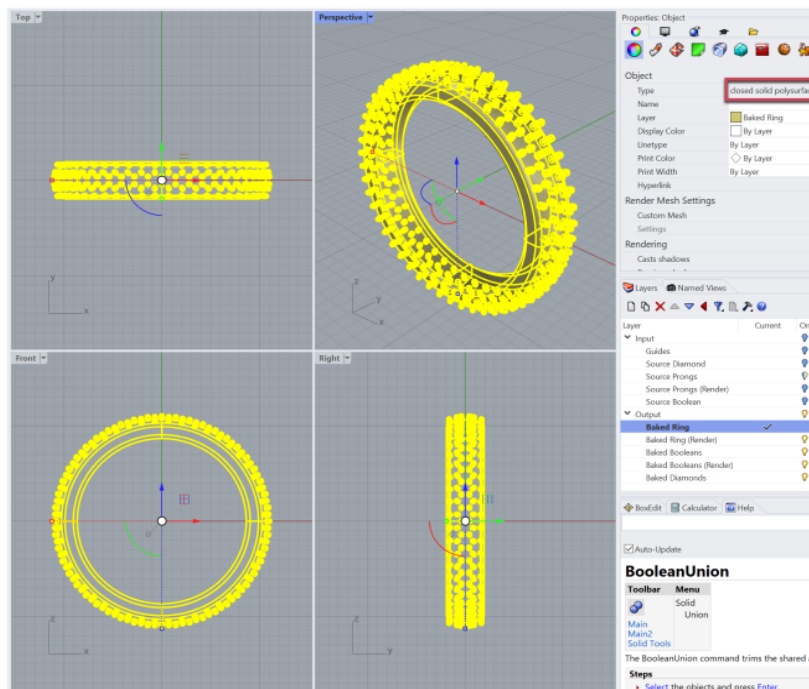
5 Group with a **Scribble** called **Prongs (Production)**.

Baking the Ring



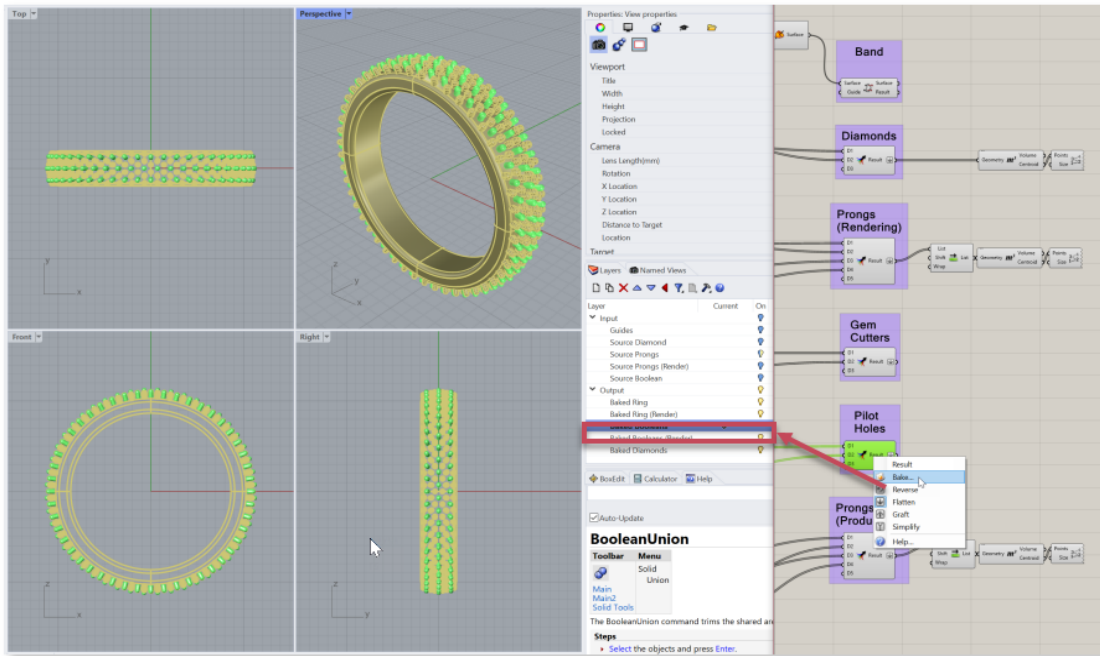
We have to **Bake** our GH results to create the geometry for Rhino. For a *production* model ring **Bake** the **BAND** and the **Prongs (Production)** by right-clicking each component separately.

Next, **GH** will ask which layer to *Bake* them into. Choose the **Baked Ring Layer** for both.



Select all the *objects* on the **Baked Ring Layer** and run the **BooleanUnion** command. After a few seconds, you should have a successful *closed solid polysurface*.

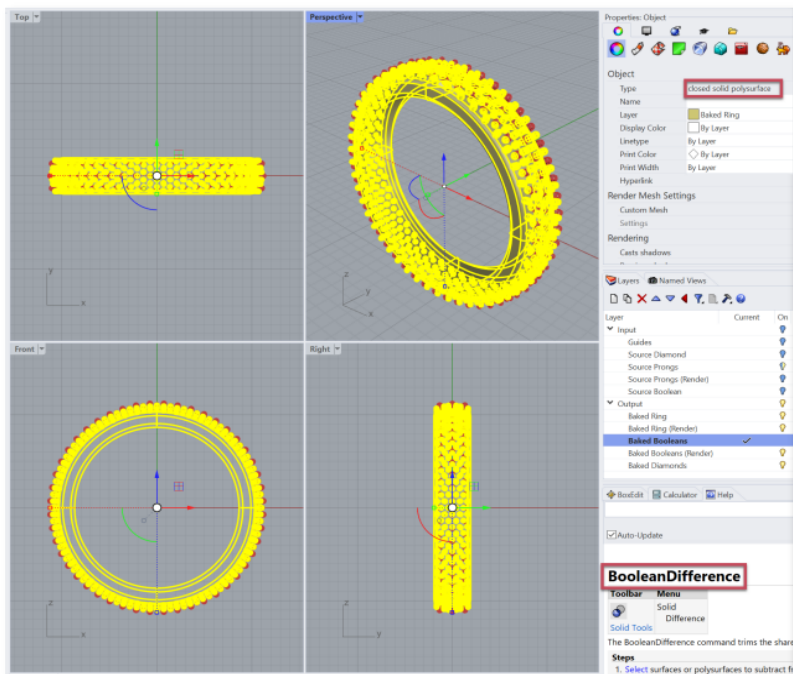
Baking the Ring



3

Again for a *production* model ring **Bake** the **Pilot Holes** by right-clicking and selecting **Bake**.

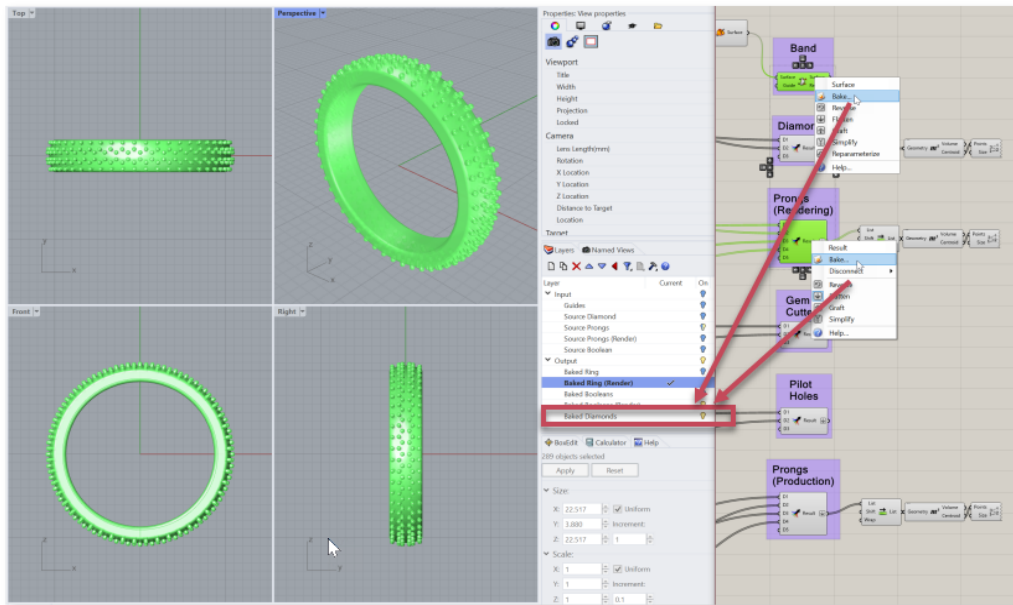
Next, **GH** will ask which layer to **Bake** them into. Choose the **Baked Booleans** Layer.



4

Select the *object* on the **Baked Ring** Layer and run the **BooleanDifference** command. Select the *objects* in **Baked Booleans** Layer for the *polysurfaces to subtract with*. After a few seconds, you should have a successful *closed solid polysurface*.

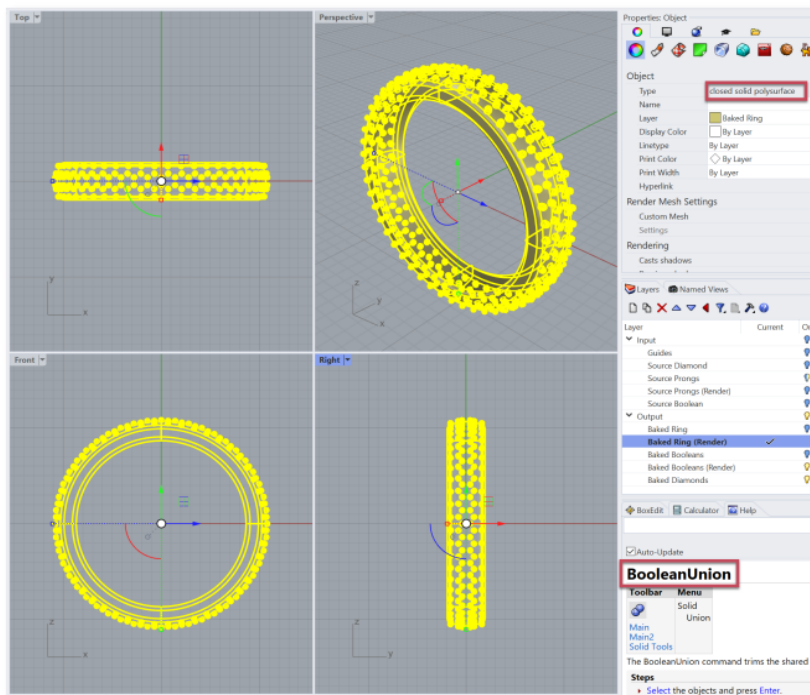
Baking the Ring



5

For a *render* model ring **Bake** the **Band** and the **Prongs (Rendering)** by right-clicking each component separately.

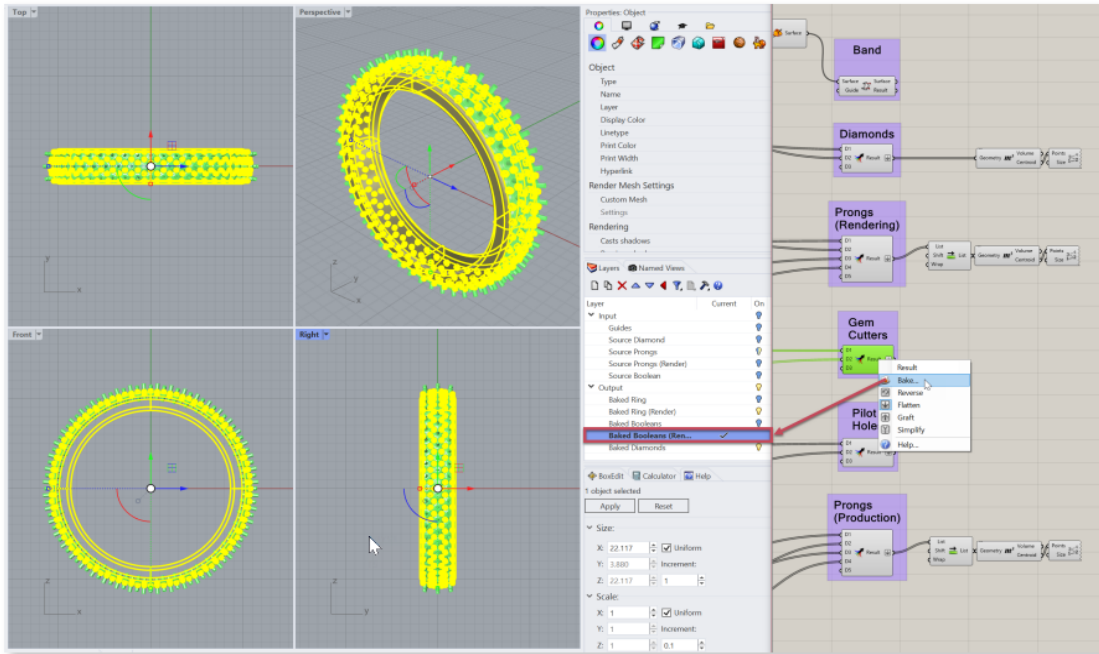
Next, **GH** will ask which layer to *Bake* them into. Choose the **Baked Ring (Render)** Layer for both.



6

Select the *object* on the **Baked Ring (Render)** Layer and run the **BooleanUnion** command. After a few seconds, you should have a successful *closed solid polysurface*.

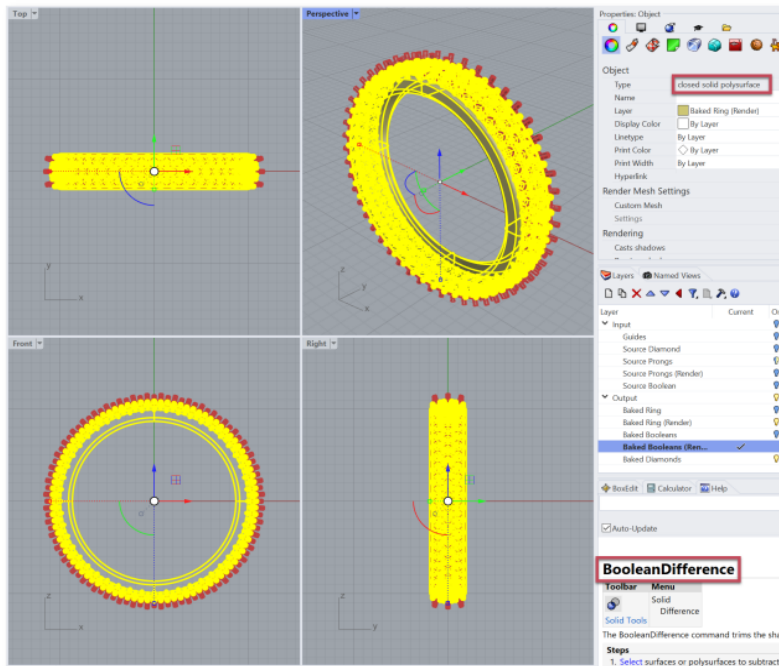
Baking the Ring



7

Again for a *render* model ring **Bake** the **Gem Cutters** by right-clicking and selecting **Bake**.

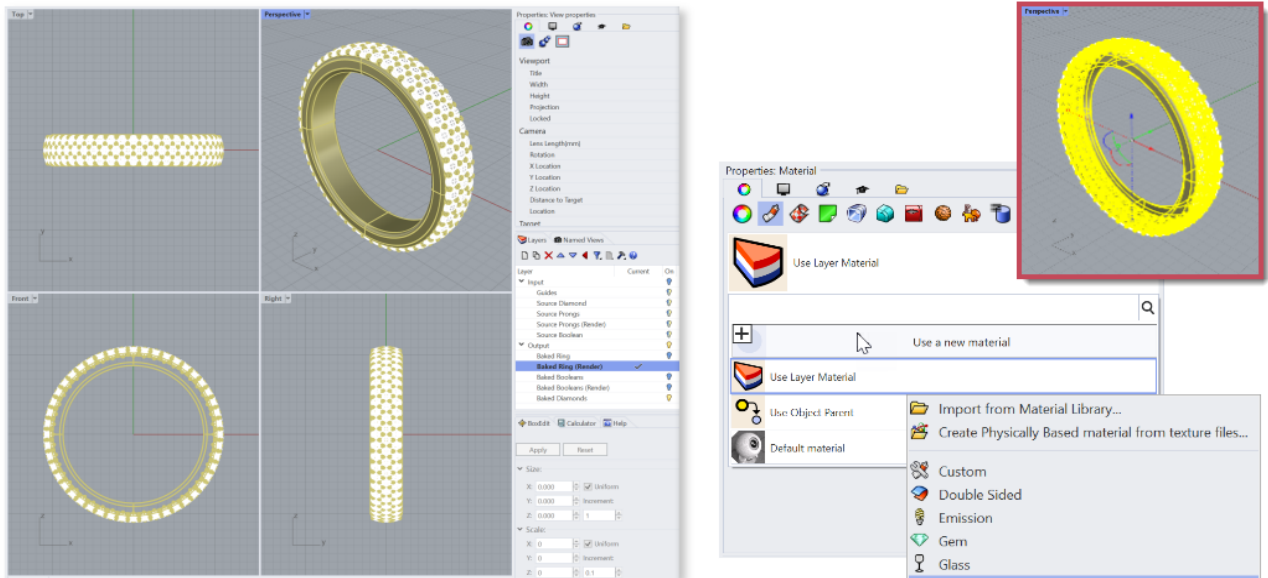
Next, **GH** will ask which layer to *Bake* them into. Choose the **Baked Booleans (Render)** Layer.



8

Select the *object* on the **Baked Ring (Render)** Layer and run the **BooleanDifference** command. Select the *objects* in **Baked Booleans (Render)** Layer for the *polysurfaces to subtract with*. After a few seconds, you should have a successful *closed solid polysurface*.

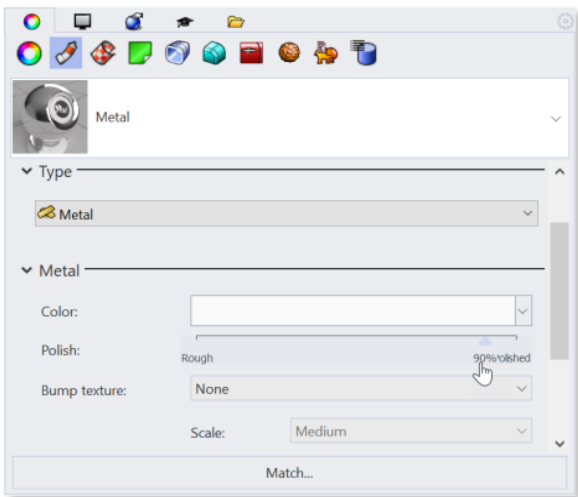
Rendering the Ring



1

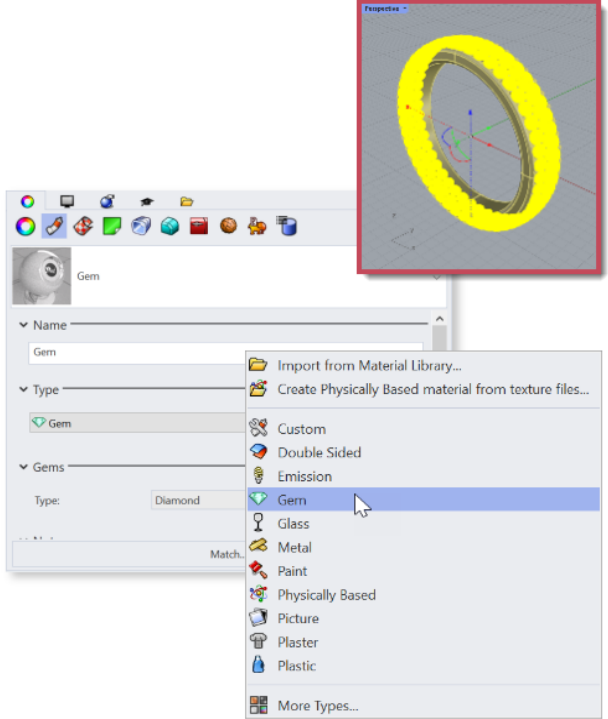
For rendering we have two Layers to work with. The ring on the **Baked Ring (Render)** Layer and the diamonds on the **Baked Diamonds** Layer.

Select the *ring* and in **Properties: Material** select **Use a new material**, then **Metal** from the sub-menu.



2

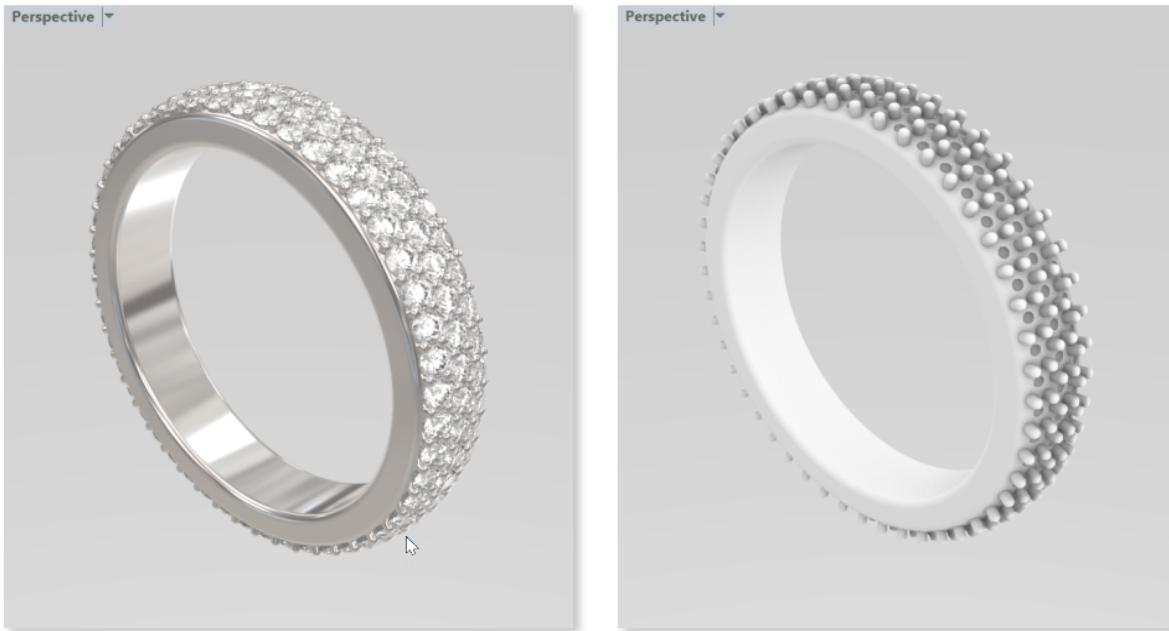
We'll keep the color **white** but raise the polish to **90%**.



3

Now select the *diamonds* and **Use a new material**. Select **Gem** and use a **Diamond** Type.

Conclusion



Congratulations! We have created a parametric Pave Set Eternity Band.

You should now be able to change ring size, width, thickness, and fillet; change stone size and count; change prong height and location. Everything we just created can be easily adapted without needing to start from scratch.

It may have seemed like a lot of work to set up but once you learn Grasshopper it really does not take any more time than modeling entirely in Rhino. My hope is that you have been *encouraged* to pursue learning this fantastic plug-in that will, over time, *save you a great deal of modeling time*.



= Awesome