

# CONTAINERS

---

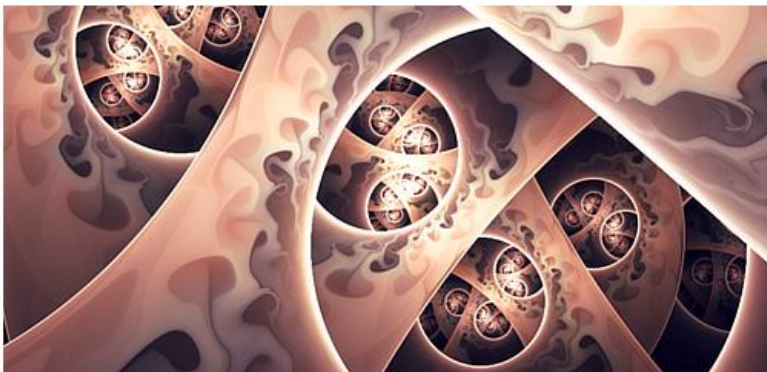
It was on my to do list for a while, and finally i moved my ass to write it.

My previous Xaos / Weights tutorials:

- [Xaos Basics](#)
- [Linked Transforms](#)
- [Shared Linked Transforms](#)
- [Shared Linked Transforms - Examples](#)
- [Multiple Loops Fractals](#)

**Attention: this tutorial may require some previous knowledge of how weights / xaos work. I will not go deep into explaining the interface and so on, so you may need to check some of the tutorials linked above.**

Below, you can see a few examples of what you can achieve using container transforms (a bit of shameless self promotion):



# Framework Review

---

In this section, I will list the frameworks which will be used in examples in this tutorial, with links to tutorials.

## Plastic

Spherical + linear or spherical + eyefish filled with bubble or hemisphere. **Tip: for structured works, spherical + eyefish is preferred as it creates less overlaps. For same reason, we will always use hemisphere instead of bubble.**

- [Spherical Gems Tutorial](#)
- [Sphrical Bubbles](#)

## Elliptic Splits

Elliptic and splits transforms combo:

- [Elliptic Splits - 2 transforms](#)
- [Elliptic Splits \(for Chaotica\)](#)
- [Elliptic Splits - A more sophisticated setup](#)

## Hypertile

Hypertile1 or hypertile2 (there is no visual difference between those two, although hypertile2 is a bit more efficient).

- [Hypertile Basics](#)
- [Hypertile Uncovered](#)
- [The world of hypertile](#)

## Foci / Unpolar

Foci or unpolar applied to a tile:

- [Foci](#)

## Glynnsim3

Glynnsim3 transform:

- [Circular Flame](#)
- [Filling Glynnsim3](#)

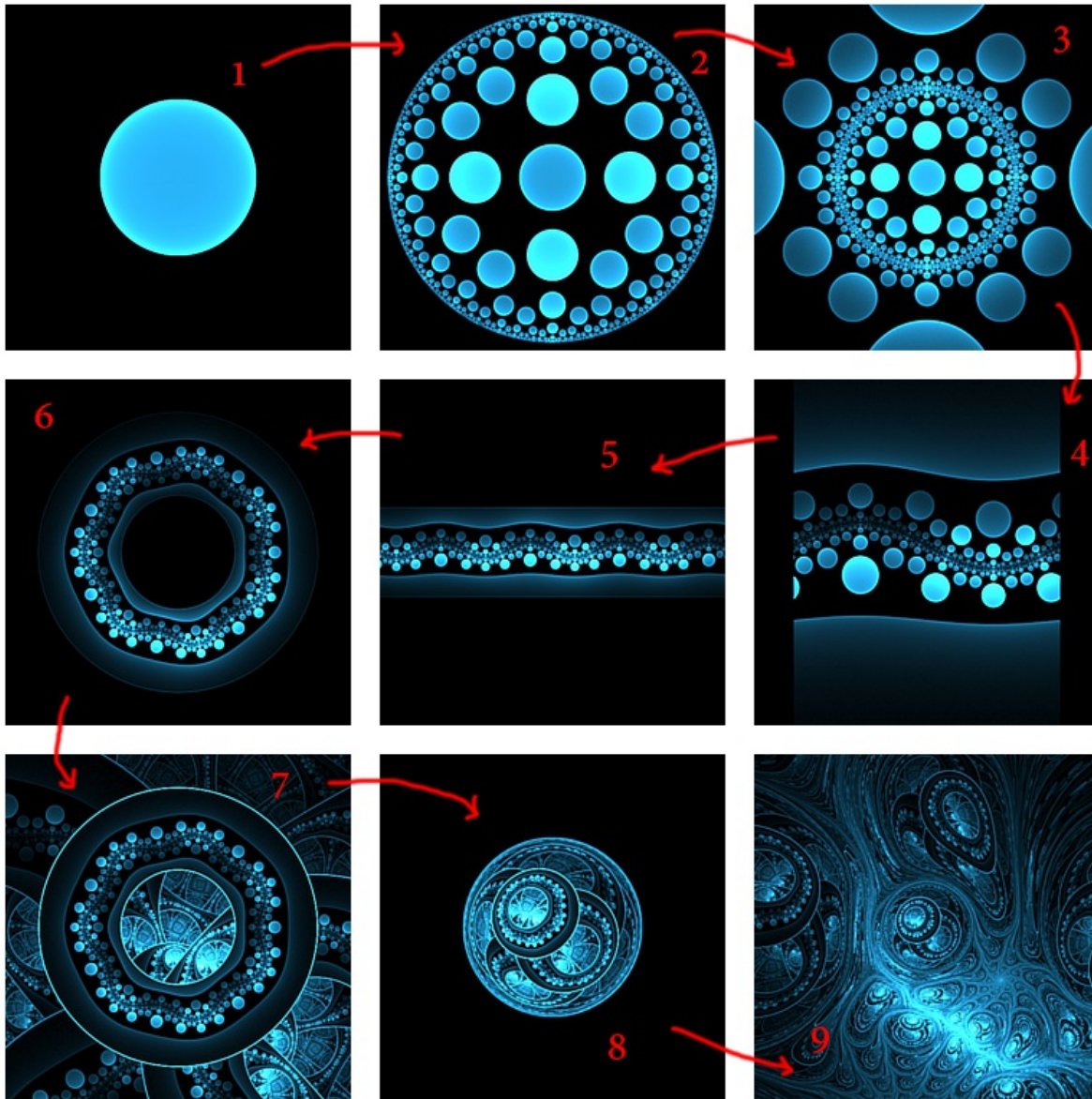
## The Basic Idea Behind Containers

Think of building something. You start with many small pieces - the transforms - which you can use to assemble bigger elements - individual frameworks - which, in turn, you can use to build the final work.

**OBS: a framework is a transform or a group of transforms that create a reasonably structured shape. Some examples of frameworks: elliptic-splits (2 transforms), plastic (1 transform with 2 functions), glynnsim3 (1 transform), hypertile (1 transform), and so on.**

Lets see an example of what can be achieved with this technique (note that this example uses same steps we will see in this tutorial, but it is a bit more advanced than the practical examples we will do step by step in next sections).

On the picture below, try tracking each framework through the transformations:



So, what happens is:

1. Sineblur
2. It goes into hypertile
3. We use two sphericals to mirror it to the area outside the circle
4. Then, apply polar2 to both sphericals at once
5. Crop and tile it
6. Here, we use unpolar to convert the stripe into a ring
7. And we use the ring to fill a glynnsim
8. Which, in turn, we put into a hemisphere
9. Finally, we fill a sherical + eyefish (aka plastic) with the hemisphere

Jumping a bit ahead, here I used container transforms twice, and linked transforms 6 times.



While you may worry about how to do it in Apo / Chaotica / JWF, its way more important to understand how the things are assembled together to make the final artwork. To use xaos / weights successfully, you must have a clear idea of what you want to achieve. Think of it as building a house: if you don't have a clear plan of everything before you start, you most likely won't get anywhere.

When making works featuring complex weights setups, I suggest you follow those steps:

**Step 1.** Decide the overall plan. What do you want to achieve? Which frameworks you will put together? In what order?

This is vital. If you just want to mess around with transforms and throw in some weights magic, don't waste your time. As you get more experience with it, you will need to do less planning.

**Step 2.** Work out the details. Are you familiar enough with the frameworks you are going to use?

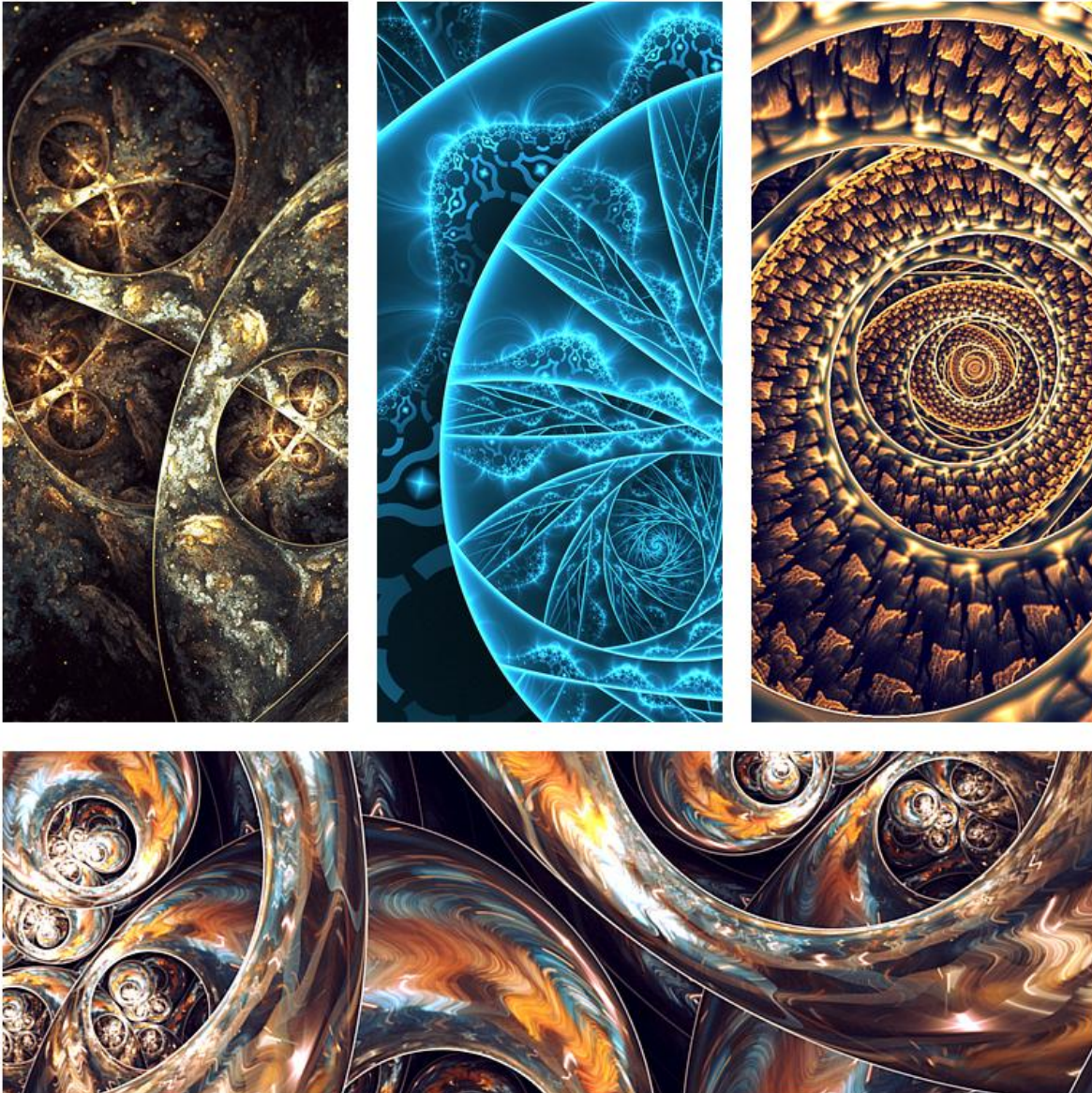
Sometimes, its useful to first make some simple flames featuring each of the frameworks: for example, if you want to fill a plastic with hypertile, you may first make a simple hypertile and a simple plastic to make sure you wont have major troubles with them once you start putting everything together. Its way easier to figure out all the sketchy points when you have 2-3 transforms and no weights in the mix. Think of how you will connect those transforms.

In the example above, I fill Glynnsim with hypertile. In order to do this, I basically follow the <http://pillemaster.deviantart.com/art/Circular-Flame-Tutorial-or-Unpolar-GlynnSim-271965226>. First, I make hypertile into square (steps 3 and 4), then tile it (5), and finally use unpolar (6). I recommend you check out the Circular Flame Tutorial to envision how all this worked.

In a few words, it is very important to understand well how each framework works and what every transform in it does when you create structured. As you want to fill one framework with another, you need to know how to put them together. Think of building again: if you build a house, you must somehow match a doorway and the door you want to use. You can fit your doorway to a door you like, you can find a door that fits the doorway you built. But if you build a hobbit house with round doorways, you would need to see at some point that regular rectangular doors won't fit.

**Step 3.** Once you have it planned, time to make it!

And just a few examples of works where I filled Glynnsim using container transforms:



## Xaos / Weights Review

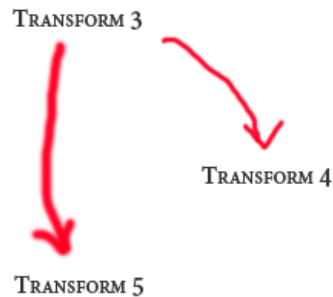
I strongly recommend you check out [this awesome explanation](#) by FarDareisMai.

In short, IFS fractals are created using "chaos game" algorithm. It works like this: we start by picking a random point, then, iteratively:

1. Pick a random transform
2. Apply this transform to the point
3. Optionally, plot the point

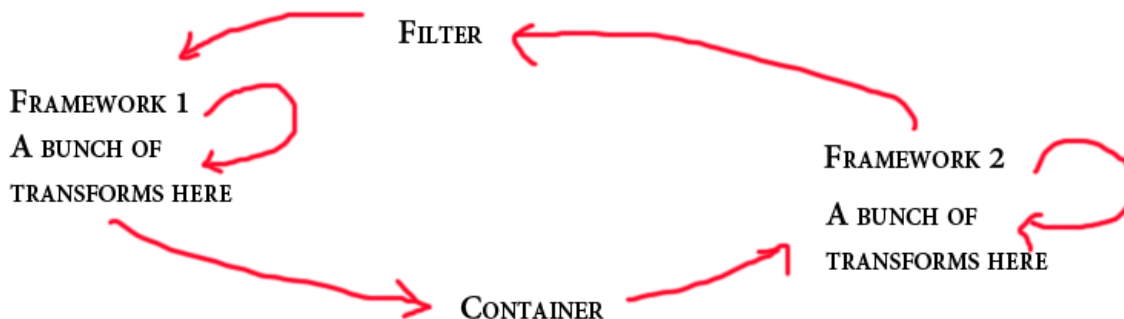
Xaos, or controlled chaos - also called weights in Chaotica - allows us to control the probabilities of selecting a certain transform in (1). Lets check a simple example:

<b>Transform:</b> 3		
Name:		
Weight: 0.5		
Triangle	Transform	Colors
Variations	Variables	Xaos
Path	Weight modifier	
to 1	0	
to 2	0	
to 3	0	
to 4	1	
to 5	2	
<input checked="" type="radio"/> View as "to"		
<input type="radio"/> View as "from"		



What does the setup above means? Transforms 1-3 won't be applied after transform 3 is applied in "chaos game". In 2 out of 3 times, transform 5 will be applied after transform 3. In 1 out of 3 times, transform 4 will be applied after transform 3.

Most relevant: using xaos / weights, you have the ability of setting the order the transforms will be applied, by setting some xaos / weight values to 0 and not allowing a set of transforms to be applied after a certain transform or group of transforms.



The fluxogram above shows the 4 basic elements of a fractal combining two frameworks. Red arrows correspond to positive weight / xaos values (everything else is null).

**The framework 1** acts as a "filler" to framework 2 (it is used as an element in **framework 2**, for example instead of hemisphere in a plastic). Each framework has some transforms and possibly even a weight structure.

The **container** is the only link between both frameworks. In xaos / weights terms, the only possible order of applying transforms is:

1. A transform from framework 1
2. Container
3. A transform from framework 2

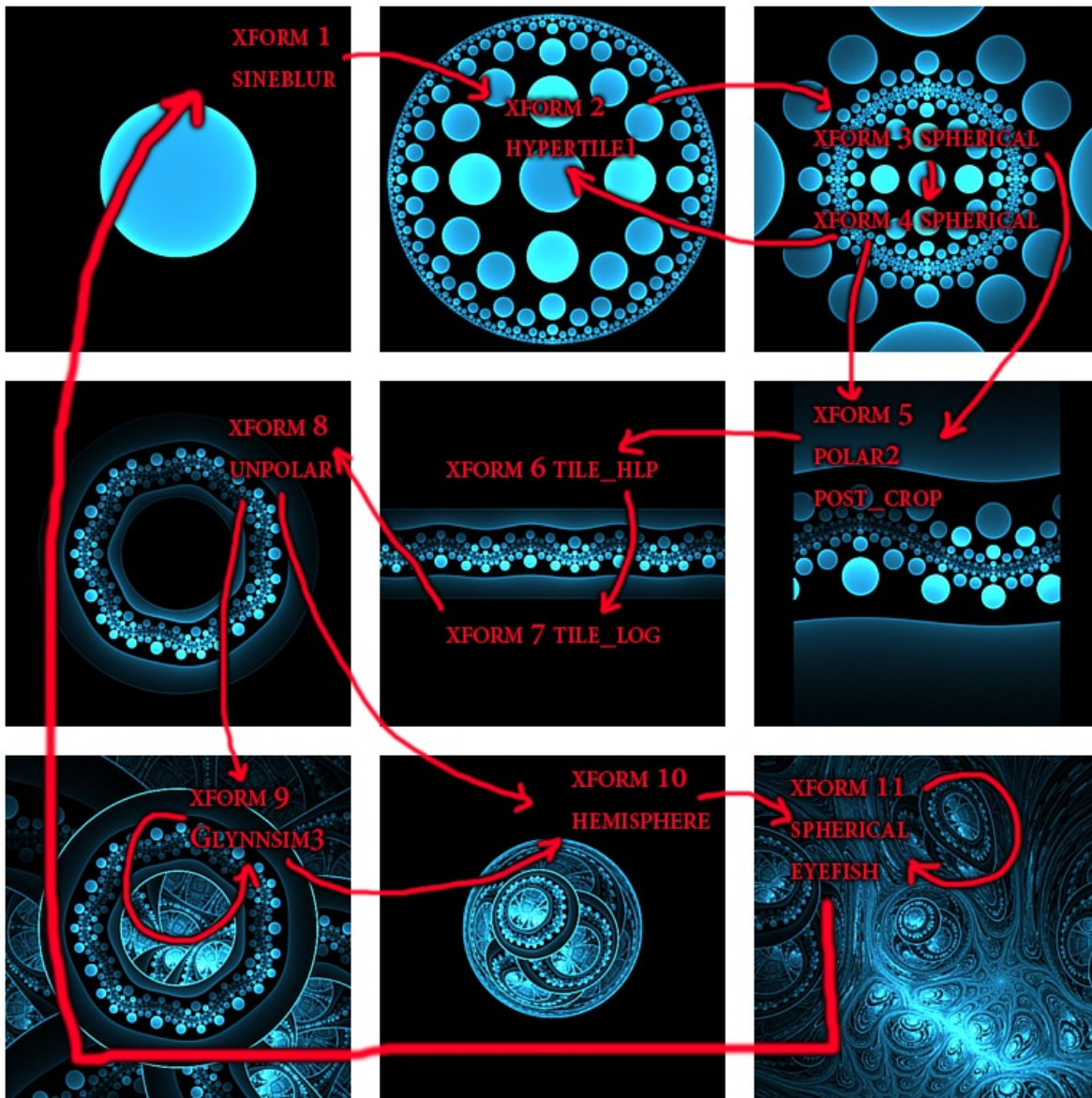
For all transforms in framework 1, the xaos / weight values for transforms from framework 2 are 0.

Finally, in "chaos game", we always must be able to select the next transform to be applied. So we cannot have a "dead end" in the fluxogram, a transform with xaos / weight values 0 to all other transforms. We need to somehow be able to go from framework 2 back to framework 1.

Sometimes, you can just use framework 2 as filler for framework 1, which is the case when we don't need a **filter**. Most commonly, framework 2 does not fit in framework 1, either because it is ugly, or technically very complex, or just cannot be done, so we use a blur or, less commonly, an empty transform.



Just so you get a proper taste of it, lets take a look at the fluxogram of the Hypertile-Glynnsim-Plastic i showed in the previous section. Red lines correspond to xaos 1, everything else is 0.



Here, we have 3 framework connected by 2 containers and 1 filter.

Here, **framework 1** is hypertile + 2 sphericals (transforms 2, 3 and 4), filled with sineblur.

Then, we have a polar2 **container** (transform 5) and a chain of linked transforms (6, 7 and 8).

The **framework 2** is Glynnsim3 (transform 9), filled with unpolar. The second **container** is hemisphere (transform 10).

Sherical + eyefish (transform 11) is the **third framework**.

Finally, sineblur (transform 1) is the **filter**. We heed a blur here, as hypertile requires something with symmetry to go into it, and a plastic does not satisfy this.

## Coloring Review

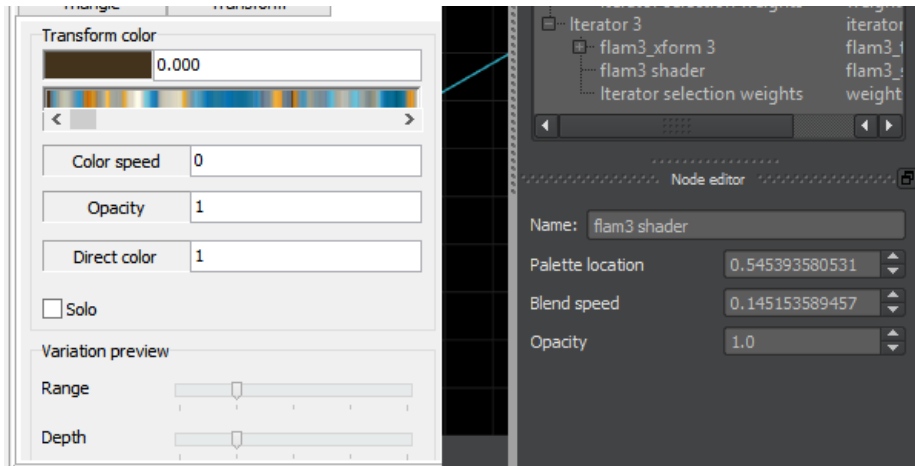
---

When using container transforms, you will deal with more transforms than usual. This means that you will have to have some extra care with the coloring to make it work.

In this section, we will review a few coloring tools which will be used later on.

### Opacity

Opacity of a transform is how visible it is. Opacity 0 means the transform is 100% transparent, and opacity 1 means its 100% opaque. Below, opacity setting in Apophysis (left) and Chaotica (right)

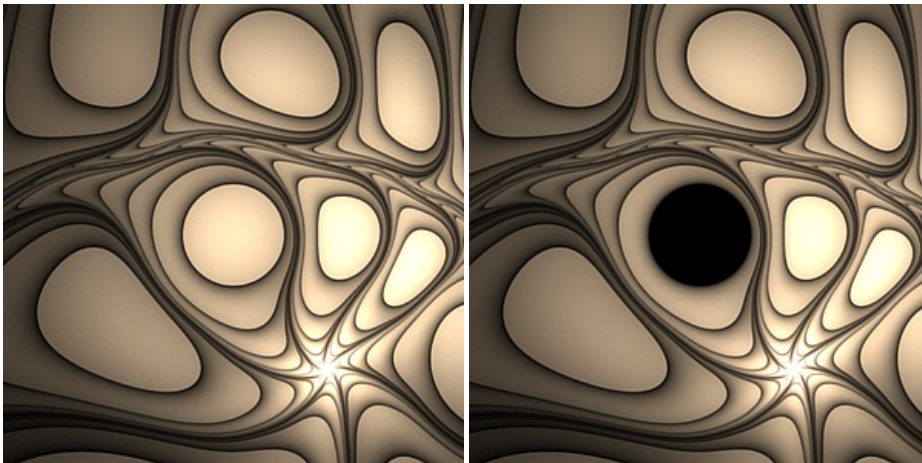


While you can use all range of values from 0 to 1 (and even above 1 in Chaotica, leading to interesting results), we are mostly interested in states 0 - transform not visible at all - and 1 - transform entirely visible.

Lets take a look at one simple example, spherical bubble framework with 2 transforms:

- tranform 1: spherical + eyefish
- transform 2: sineblur (I used blur here to have less interference)

Below, both transforms with opacity = 1 (left) and sineblur with opacity = 0 (right)

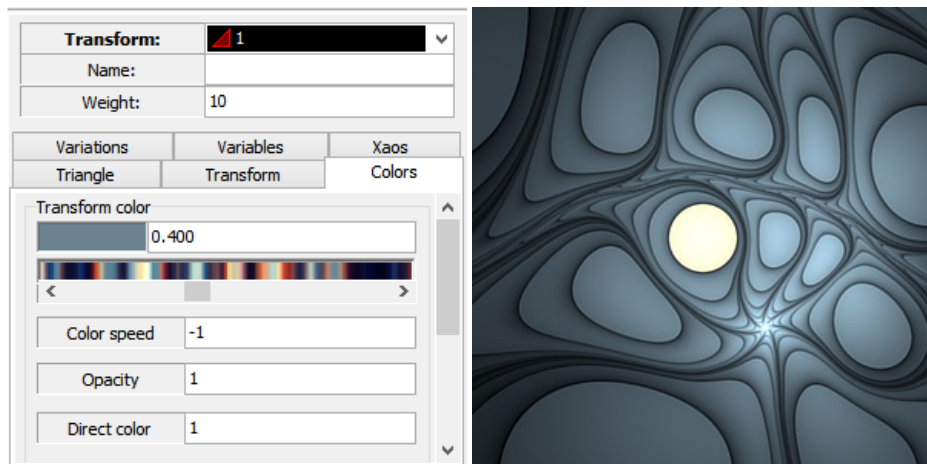


### Speed

Speed is called **color speed** in Apo and **blend speed** in Chaotica. This variable defines how much of the transform color (aka palette location) will be mixed in on each iteration.

We are interested, in particular, in 3 settings. Once more, we will go through them using the spherical + sineblur parameters.

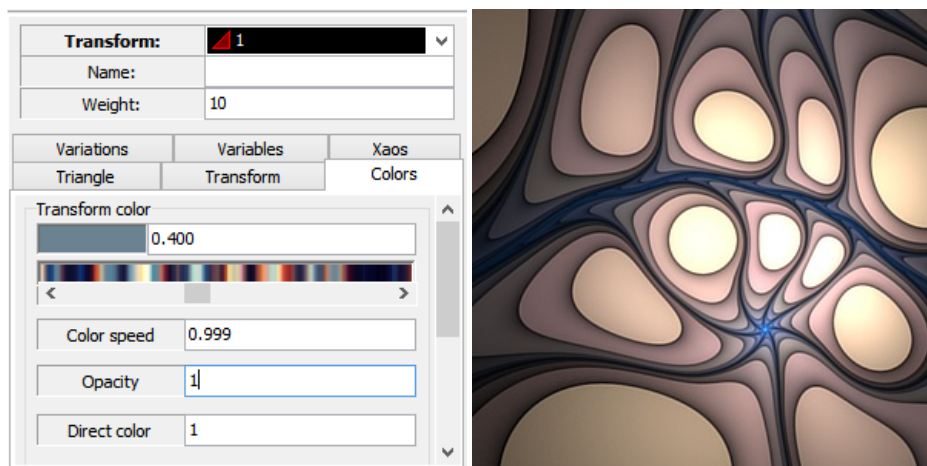
**Setting 1:** solid color. To make a certain transform solid color, set color speed = -1 (blend speed = 1 in Chaotica).



Notice how all the area that corresponds to transform 1 became blue, without any color transitions or gradient.

**Setting 2:** very subtle transition. This is almost opposite of solid color: the color is added in very small amounts on each iteration, creating a very smooth and subtle colour change.

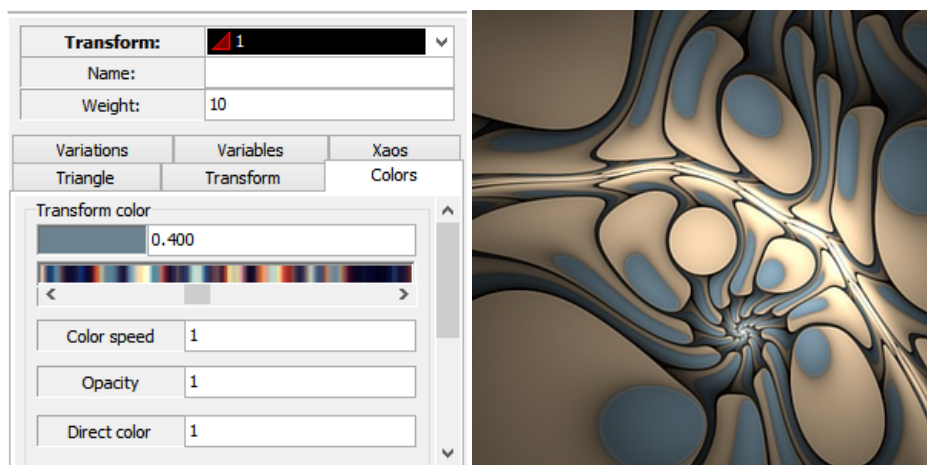
For this effect, we usually use color speed around 0.99 or something like that (blend speed = 0.005 in Chaotica).



Observe how the color slowly fades to blue.

**Setting 3:** no color change at all. When using container transforms, we usually end up having huge chain of transforms. If we allow each of them affect the coloring, the final result will be, in best case, hard to control - and in worst, very close to solid color version. This is why we just use color speed 1 (blend speed = 0 in Chaotica).

OBS: I modified the setup, adding a second sineblur.

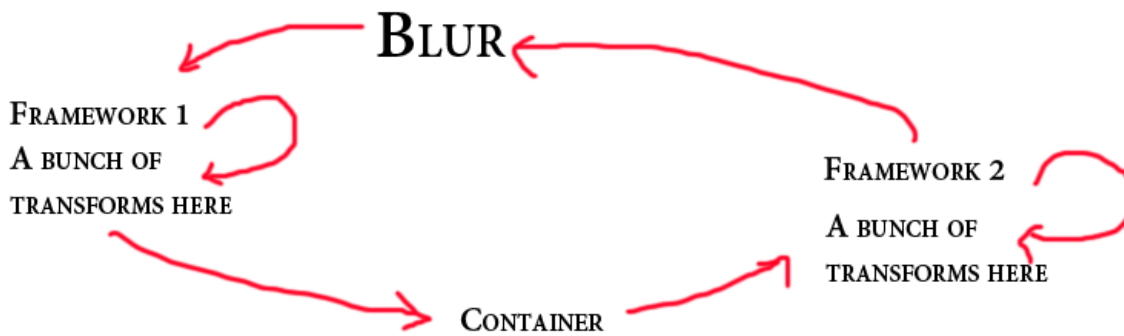


Observe how the color combination is preserved through the pattern.



## Container Transforms With Blur Filter

This is the most simple container transform setup.



What do we have on the fluxogram above?

The **Blur filter** is actually a part of the **framework 1**. For example, framework 1 is something like a spherical bubble with blurred bubble (such as in the example from previous chapter, where I used sineblur instead of bubble / hemisphere).

The **framework 2** does not affect framework 1, because of the blur link (the blur blurs everything out, so it is not visible).

On the other hand, framework 1 is part of framework 2.

And now, lets finally try something out.

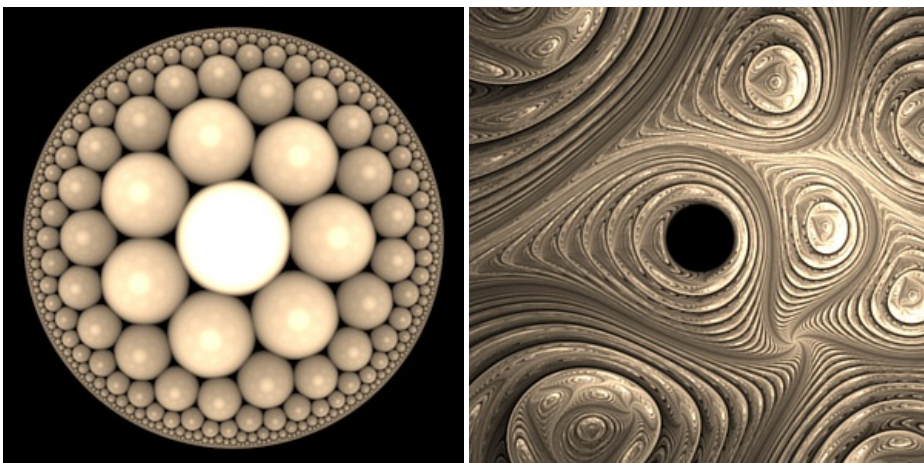
### Example 1: Hypertile Inside a Plastic

In this example, we will use a hypertile as framework 1, and a plastic as framework 2.

First, lets take a look at each framework by itself. You may use the parameters below, but I strongly recommend that you try creating those setups yourself.

- [Hypertile parameters](#)
- [Plastic parameters](#)

Lets look at previews of both frameworks. On the plastic preview, I set opacity of hemisphere transform to 0.



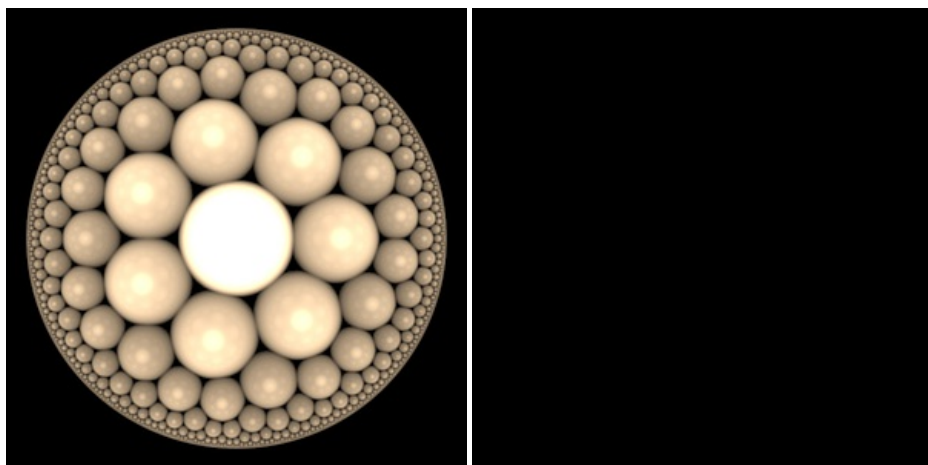
How to connect those 2? If you could just scale down the hypertile and use it instead the hemisphere, that would do the job. Both are round, its a fair replacement. **Notice that we don't need any additional transformation for hypertile to replace hemisphere - this means we can use a linear container.**

Follow the steps below to create your first container transform (and use it):

**Step 1:** Start with a hypertile. In my example, transform 1 is hypertile1, and transform 2 is hemisphere with pre\_blur.

**Step 2:** Now, set the opacity of those two transforms to 0. As we want the hypertile to be part of something else, we don't want the original visible.

So far, we have the following results: step 1 (left) and step 2 (right):



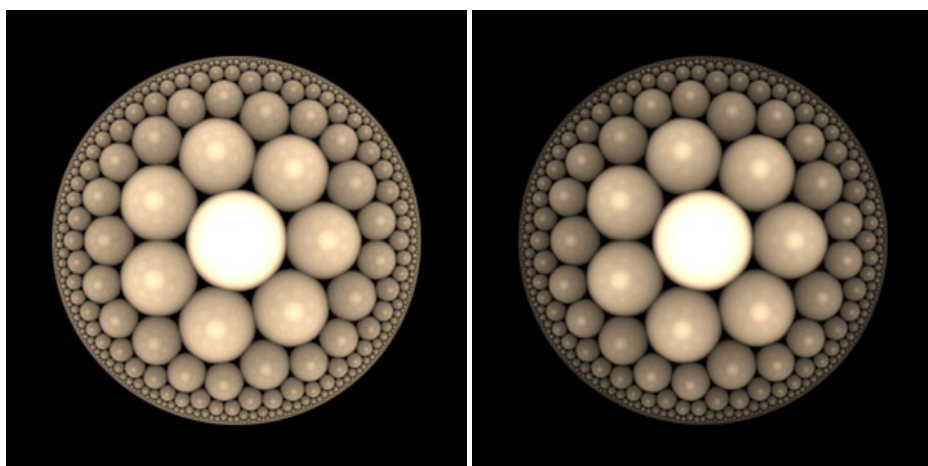
**Step 3:** Add a new transform, with linear only.

**Step 4:** Finally, we can set up the weights:

- transform 1: 1 to itself, 1 to transform 2 and 1 to transform 3 (linear)
- transform 2: 1 to transform 1, 1 to itself and 1 to transform 3 (linear)
- transform 3: 0 to transform 1, 1 to transform 2 and 0 to itself

And we got our linear container transform!

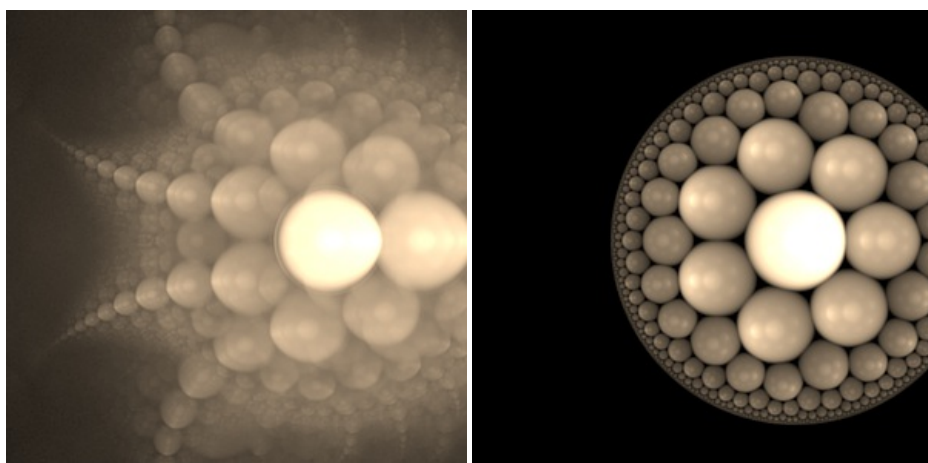
After steps 3 and 4, your hypertile should look this (step 3 on the left, step 4 on the right):



Yes, you went through 10 pages just for this ;)

Observe that, in this case, adding a linear transform seems to not affect the pattern. But try moving the linear around before and after setting up the weights as described in step 4.

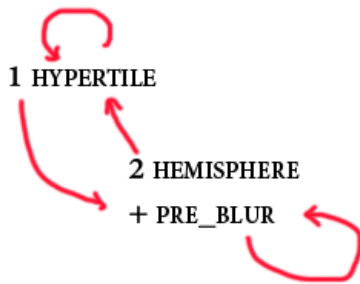
Below, I moved transform 3 by 0.5 right on setup of step 3 (left) and setup of step 4 (right).



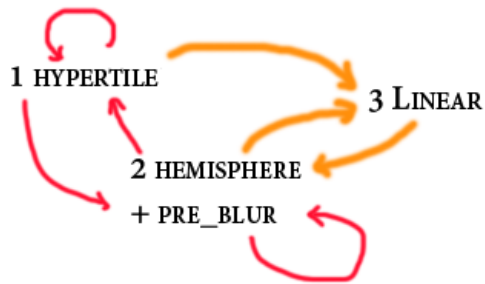
Why does this happen and how does it work?

Lets go back and review the weights structures we implemented in the 4 steps above.

### STEP 1 SETUP (THE REGULAR HYPERTILE)



### STEP 4 SETUP (WITH LINEAR CONTAINER)



Above, we have the original weights, and the ones you should have after step 4.

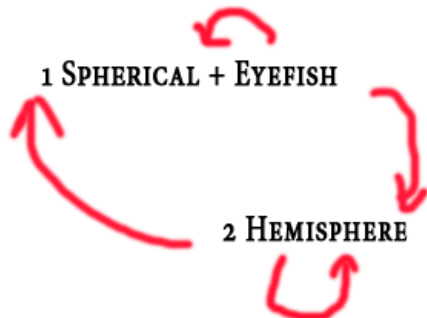
Observe that the weight / chaos relations between transforms 1 and 2 remained: this is necessary to preserve the hypertile pattern we are interested in.

Then, both hypertile and hemisphere send information to the container. Lets try to picture it in terms of "chaos game" algorithm. Instead of being plotted on screen, those transforms now send points to the linear - which acts as some sort of "virtual" screen, containing the image of the hypertile pattern (and this is why it is called **container**).

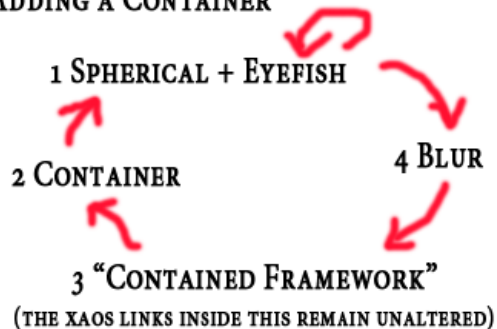
On the other hand, the linear container sends points only to transform 2, which has **blur**. This way, it will not be visible in the hypertile (because it will be blurred).

Now that we created a container transform, lets put it into a spherical + eyefish pattern. The first step is to figure out the Xaos structure.

### NORMAL SPHERICAL FRAMEWORK



### ADDING A CONTAINER



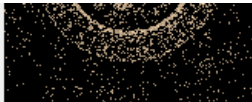
Take a look at the chart above. We already have:

- "contained framework": hypertile and hemisphere + pre\_blur (transforms 1 and 2)
- container: linear transform 3 we added above
- blur filter: while this is pictured as a separate transform on the fluxogram, pre\_blur acts as such, so we don't need an additional blur transform

The only transform we need to add is **spherical + eyefish**. So:

1. add a new transform
2. replace linear with spherical = 1 and eyefish = 0.5
3. As shown on the fluxogram above, this transform should only recieve points from the container transform (transform 3) and itself (transform 4). So set the weights as shown below (left image).
4. Also, it sends points only to itself (transform 4) and the blur transform (hemisphere + blur, transform 2). Set up the weights accordingly (right image).



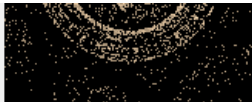


**Transform:** 4

Name:

Weight: 0.5

Triangle	Transform	Colors
Variations	Variables	Xaos
Path	Weight modifier	
from 1	0	
from 2	1	
from 3	1	
from 4	1	



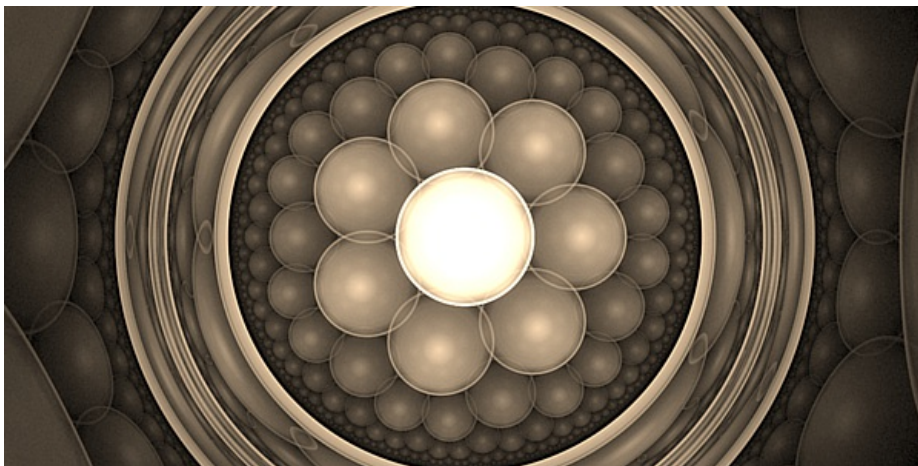
**Transform:** 4

Name:

Weight: 0.5

Triangle	Transform	Colors
Variations	Variables	Xaos
Path	Weight modifier	
to 1	0	
to 2	1	
to 3	1	
to 4	1	

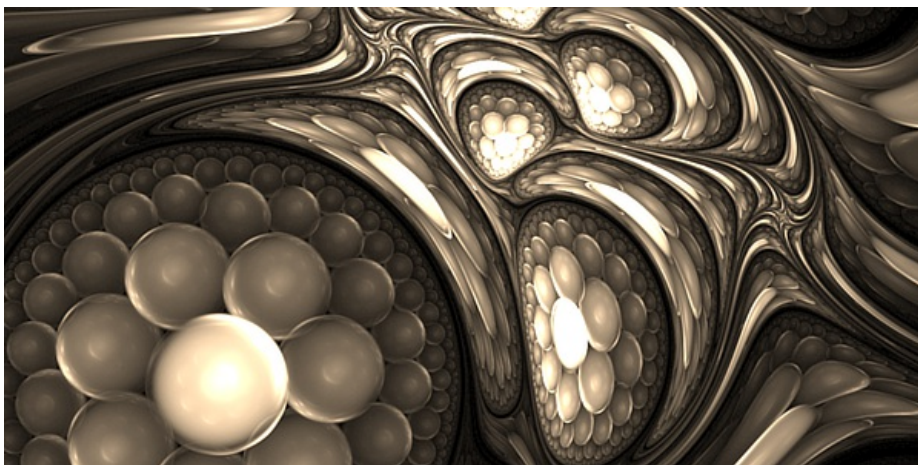
Your result should look somewhat like this:



Some tweaking tips:

1. Increase the weight of transform 4 (spherical+eyefish).
2. Scale transform 4 down, and move and rotate it.

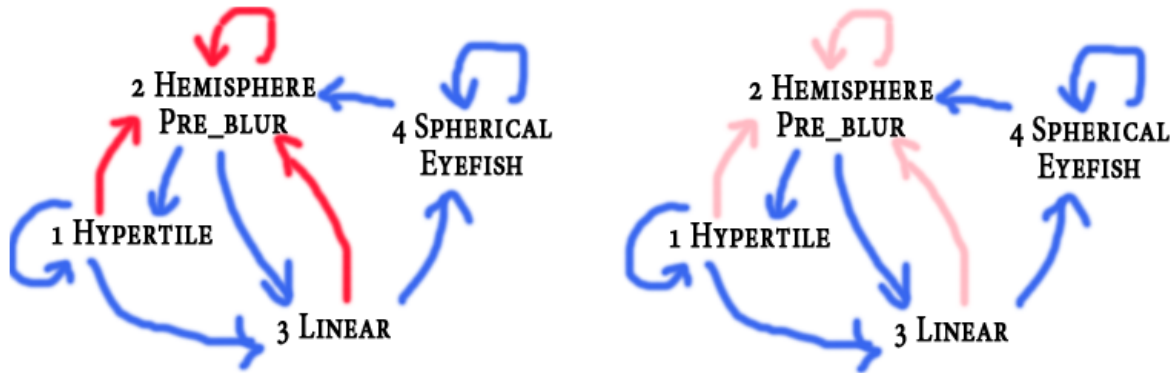
And you will get something like this:



If you need, you may also check the parameters for my final version here: <http://sta.sh/03omz9atr17>

## Tip: Cleaning Up Weights

Still working on parameters from **Example 1**, take a closer look at the xaos structure we created:



The xaos links highlighted in red:

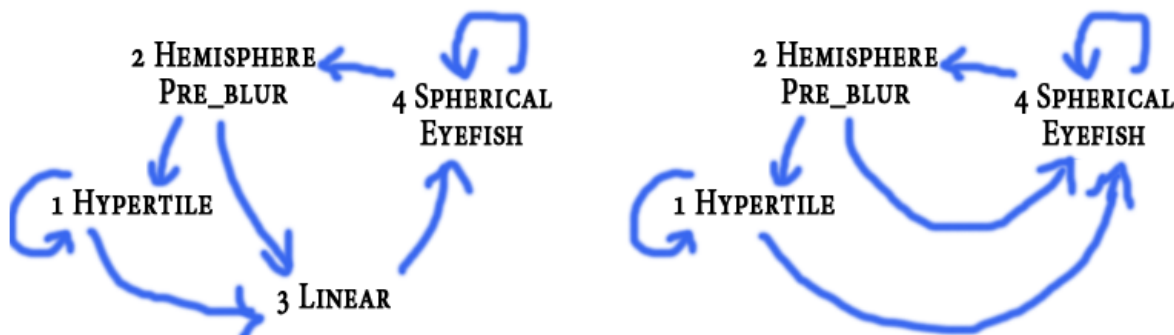
- add nothing to the visual (since transform 2 has pre\_blur, whatever goes into this transform will be blurred, and we will not see it).
- are not necessary to create a "closed" flow of points through all transforms. See that you still go from a transform to any other transform following the blue arrows without getting stuck in a dead end.

Removing those links will result in a faster and cleaner render, although you may need to do some tweaks to keep the original look.

## Was The Container Transform Actually Necessary In This Example?

As some of you may have noticed, the only function of the linear transform (transform 3) is to work as a link between hypertile and the plastic frameworks. This transform doesn't add any additional distortion or effect.

Which means we could remove it if we want to.



On the picture above, the left fluxogram shows the original xaos setup. The right one shows same setup, but without the linear transform: observe how, instead of going through linear, hypertile (transform 1) and hemisphere (transform 2) connect directly to spherical + eyefish transform.

## Example 2: Elliptic Splits Inside Glynnsim3

Now, we will use elliptic splits as framework 1, and glynnsim3 as framework 2.

Again, the first step is to study the frameworks individually. You may use the parameters below, but I strongly recommend that you try creating those setups yourself.

- [Elliptic Splits parameters](#)
- [Glynnsim parameters](#)

Lets look at previews of both frameworks.



First of all, observe that we have blur (inside cylinders of the Elliptic Splits pattern).

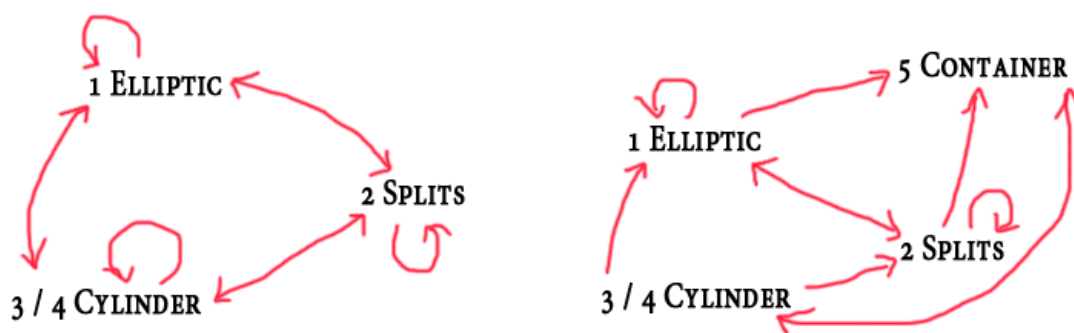
Unlike in example 1, this time we do not have a "natural" fit: we will need to use some sort of transformation to fit elliptic split into Glynnsim. There are 2 techniques that allow such fit:

- <http://pillemaster.deviantart.com/art/Circular-Flame-Tutorial-or-Unpolar-GlynnSim-271965226>
- <http://tatasz.deviantart.com/journal/Filling-Glynnsim3-596233151>

In this example, we will use the second one, which means a spherical + eyefish container. Specifically, we will use a chain of 3 transforms as container:

1. julian
2. spherical + eyefish
3. hemisphere

The first step it to put elliptic splits into a container transform. The xaos setup is described on the fluxogram below:



The step by step is:

1. Set opacities of all transforms to 0
2. Add a new transform (transform 5)
3. Set color speed of transform 5 to 1
4. Still on transform 5, set the "to" weights to transforms 1, 2 and 5 to 0 (leaving 1 value only to transforms 3 and 4, which have cylinder and pre\_blur)
5. On transforms 3 and 4, set all "from weights" to 0, except the ones from transform 5.
6. Now, go back to transform 5 and add a linked transform (transform 6)
7. Go to transform 6 and add another linked transform (transform 7)

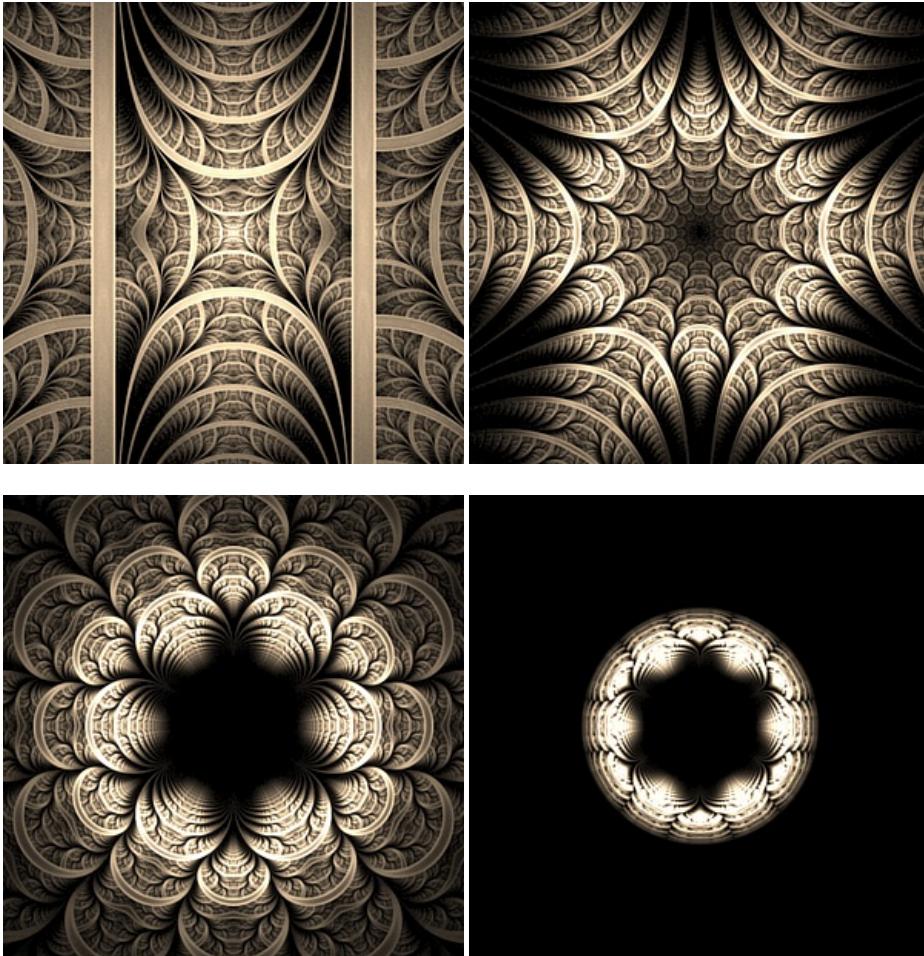
If your fractal looks weird, tweak the weights of transforms 1-4. At this point, you should have something like this: <http://sta.sh/0icfuepplvz>.



Now, time to change those linear containers to achieve the shape we want:

- On transform 5, replace linear with julian (I used power = 3 in this example)
- On transform 6, replace linear with spherical = 1 and eyefish = 0.25
- On transform 7, replace linear with hemisphere

Don't worry too much with details at this point, it's easier to tweak it later. This is the evolution:



At this point, you can see how this container will fill the Glynnsim3 pattern. And it's time to add the second framework.

This is the fluxogram for it:

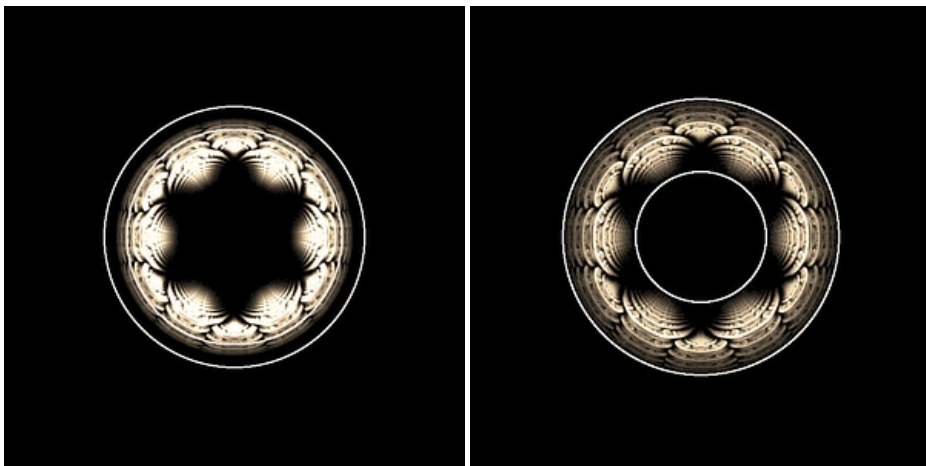


And, if needed, the step by step to setup the chaos:

1. Add a new transform (transform 8)
2. Replace linear with Glynnsim3 (set contrast to 0.8 and power to 0)
3. Still on transform 8, set all "from" weights to 0, except ones from transform 7 and transform 8
4. Set all "to" weights to 0, except ones to transforms 3 and 4
5. Now, on transforms 3 and 4, set them to receive points only from transform 8

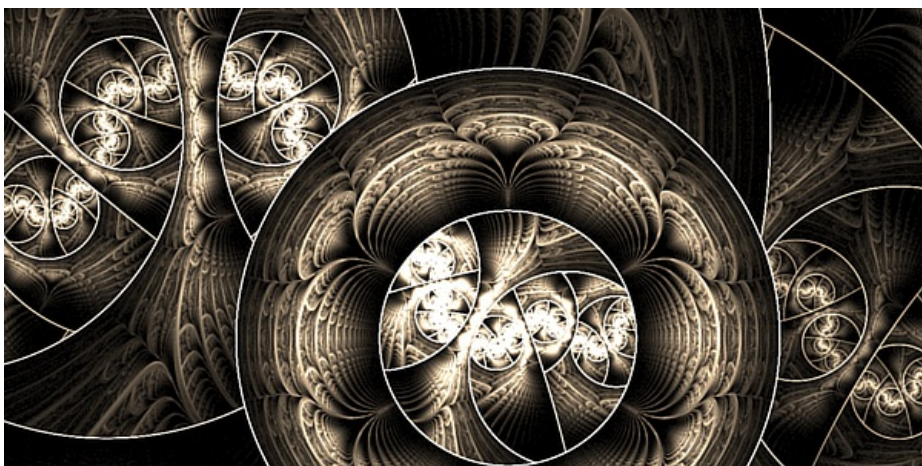
Notice that the elliptic splits "ring" doesn't fit Glynnsim3 perfectly. To fix that, increase Glynnsim3 thickness (to something like 0.45) and, on transform 7, scale up the post transform (a scale factor of 145% will work in this example).

Below, the original (left) and the fitted ring (right):



My parameters in case you had troubles: <http://sta.sh/01vbqop69sxp>

Now, all you have to do is move, rotate and scale the GlynnSim3 transform to get something cool like this:



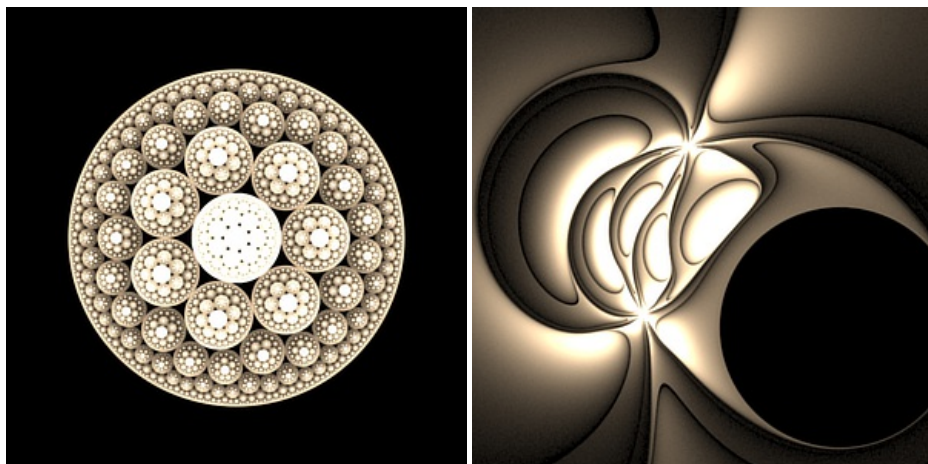
Or, with a few more tweaking, like this:





## Container Transforms With Empty Transform Filter

And we are back to our first example - hypertile and plastic. Unlike in first example, this time hypertile features no blur:



We cannot just replicate the setup of the first example: since we have no blur, it will turn into a bit of a mess. Look closely at the bubbles - the pattern is not very clear (because plastic is not symmetrical and hypertile requires symmetry):



To sort this out, we will use the technique first described in this awesome tutorial by piethein21: <http://piethein21.deviantart.com/art/Hybrids-tutorial-328226439>.

First, we add the filter transform:

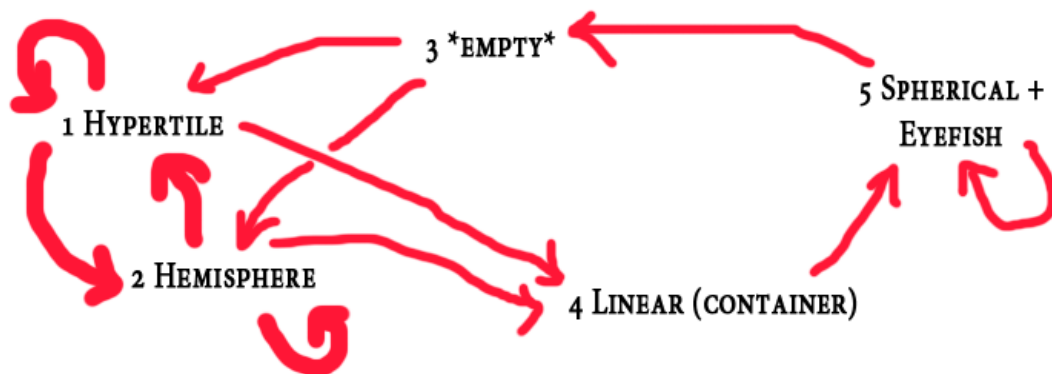
1. set all weights between existing transforms in hypertile to a high value, lets say 20.
2. add a new transform
3. set linear = 0 (and add no other variations)

Path	Weight modifier
to 1	20
to 2	20

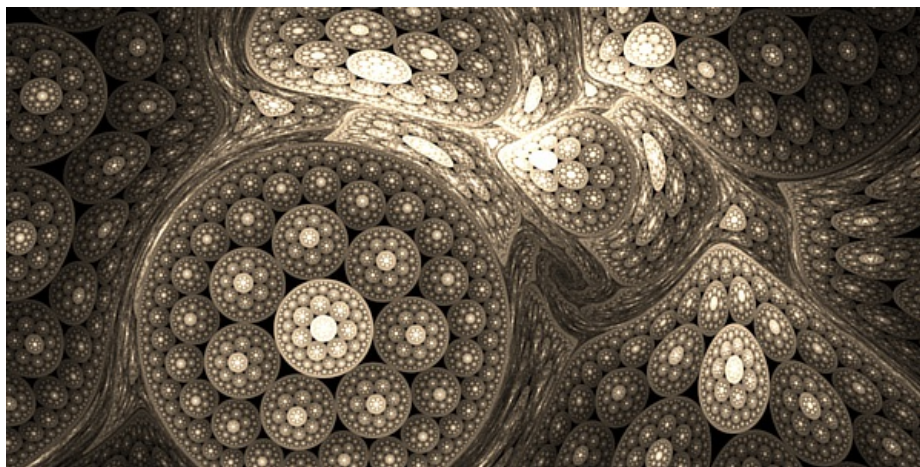
Name	Value
linear	20
flatten	0
sinusoidal	0
spherical	0
swirl	0
horseshoe	0
polar	0



And now, we have a filter and all we have to do is to set everything up (also don't forget to turn the opacities off):



The step by step is very similar to both previous examples, so I will show just the final result. You may also check out the parameters here: <http://sta.sh/025aslstqtf2>.



Notice how the hypertile does not have overlaps anymore.

## Container Transforms With No Filter

---

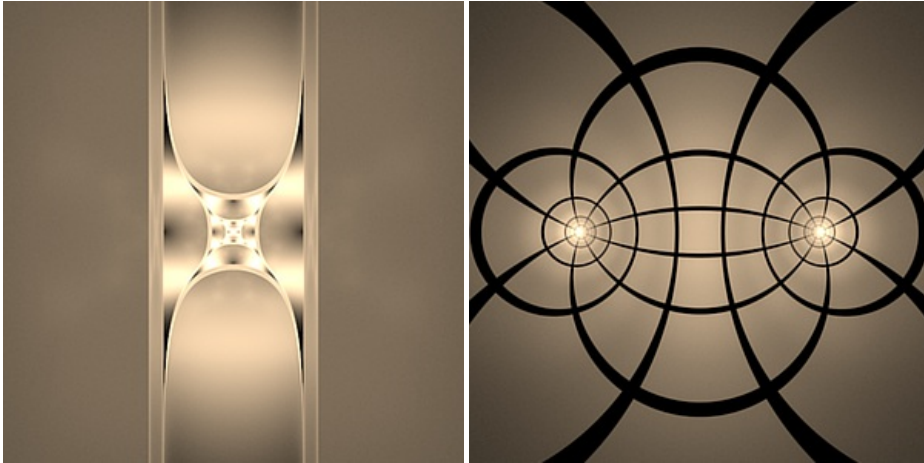
And this is the "king" of all container transforms. It produces the most stunning and also the most "fractal" results. In this setup, framework A fills framework B, but framework B also fills framework A.

For this example, you will need the following transforms by [Zy0rg](#):

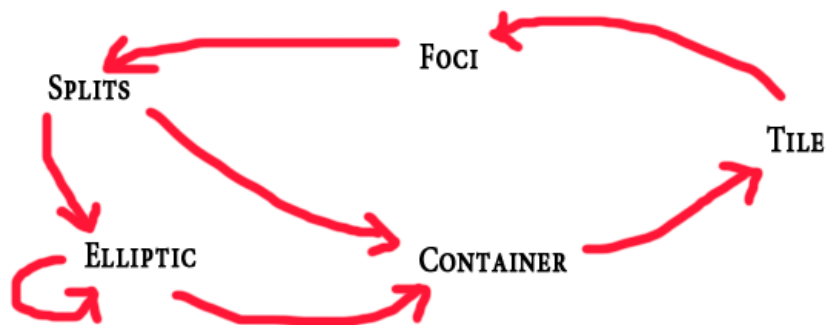
- [Tile\\_log](#)
- [Tile\\_hlp](#)

We will work with those 2 frameworks:

1. Elliptic - Splits
2. Foci



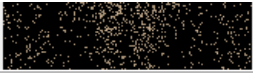
The planned xaos setup is roughly like this:



So, lets start with some basic elliptic splits: [Elliptic Splits parameters](#).

1. Set all opacities to 0.
2. Add a new transform (transform 5)
3. On transform 5, set all "to" weights to 0, except the one to splits (transform 2)
4. Observe that, on fluxogram above, splits will recieve points from the other framework only. So, for now, go to transform 2 and set up all "from" weights to 0, except from transform 5.

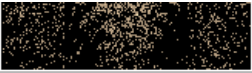
Below, step 3 (left) and 4 (right):



<b>Transform:</b>	5
Name:	
Weight:	0.5

Triangle	Transform	Colors
Variations	Variables	Xaos

Path	Weight modifier
to 1	0
to 2	1
to 3	0
to 4	0
to 5	0




<b>Transform:</b>	2
Name:	
Weight:	0.5

Triangle	Transform	Colors
Variations	Variables	Xaos

Path	Weight modifier
from 1	0
from 2	0
from 3	0
from 4	0
from 5	1

We got the container set up. Now, replace linear with bipolar + crop on transform 5:

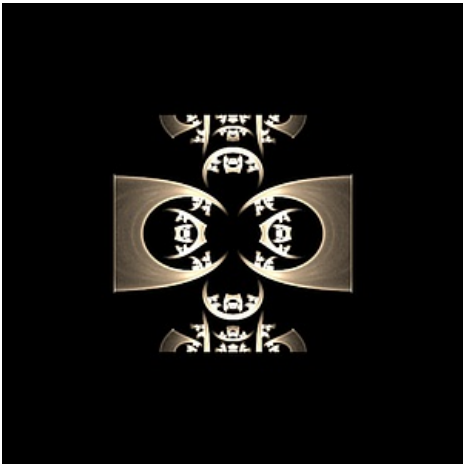


<b>Transform:</b>	5
Name:	
Weight:	0.5

Triangle	Transform	Colors
Variations	Variables	Xaos

Search:

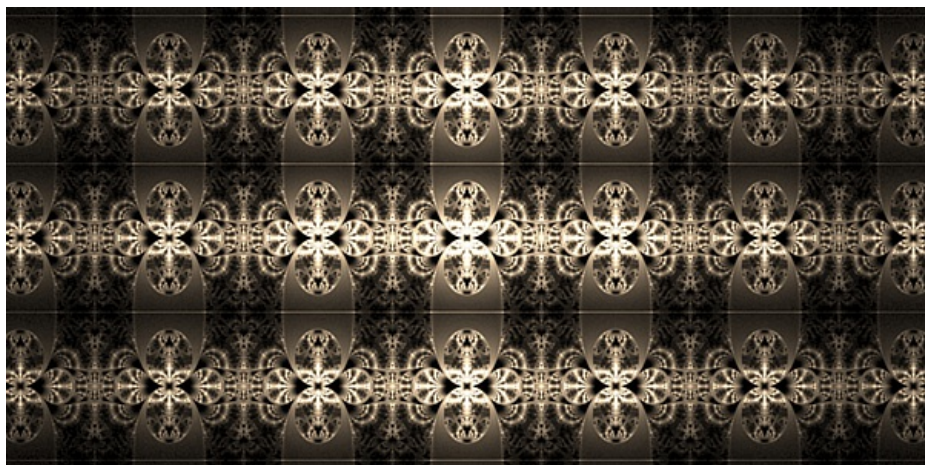
Name	Value
bipolar	1
post_crop	1



Now, its tile to tile and "focify" it.

1. On transform 5, add a new linked transform (transform 6), with tile\_hlp = 1
2. Scale transform 6 down by 50%
3. Add a new linked transform (transform 7), with tile\_log = 1
4. Add a new linked transform (transform 8), with tile\_hlp = 1
5. Rotate transform 8 by 90 degrees
6. Add a new linked transform (transform 9) with tile\_hlp = 1

Here we go, the tile is ready:

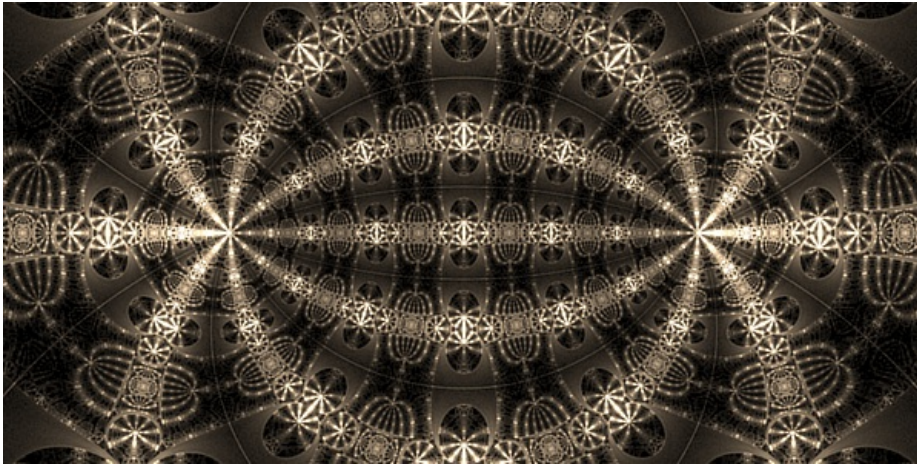


We are almost done now.

1. Add one last linked transform (transform 10) with foci = 1
2. Scale it up by 314.159%
3. Scale it down by 200% a few times

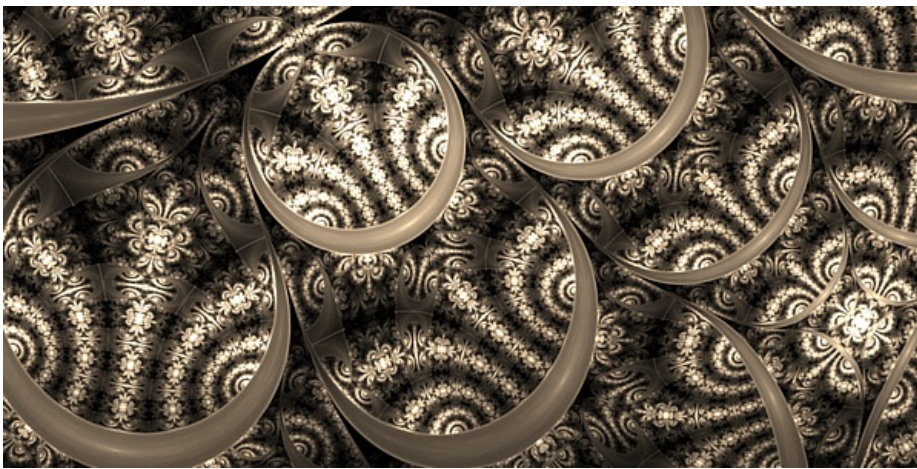


This is what you should have after the last step:



A final touch, set the opacity of transform 10 to 0. And then go back to transforms 1-4 and make them visible again.

And well, tweak until you get something cool. Here is my final result: <http://sta.sh/0gocsmf1jj8>



As exercise, I leave to you to make something that is not described on this tutorial. Because that is the point really, not just copy the examples (even if you tweak them).