FOCI FOR XAOSFOBIC

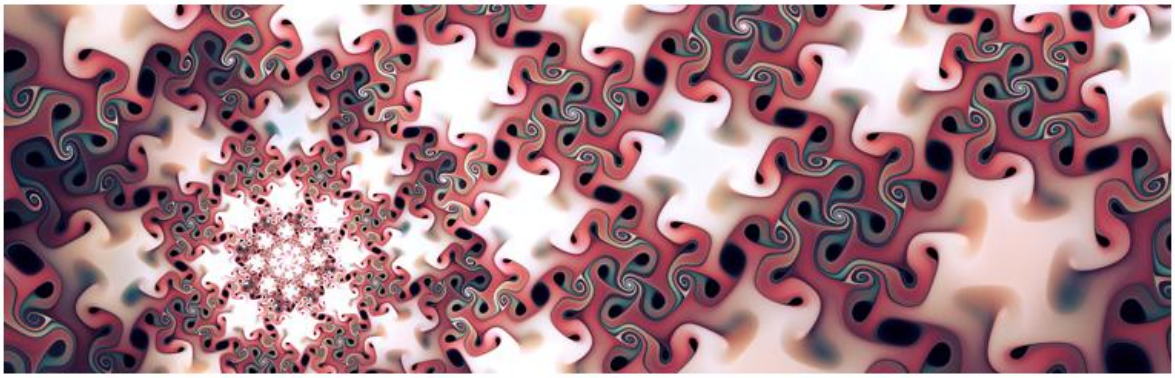
A small foci trick for xaosfobic and lazy people in general. I've introduced it first in my Using Chaotica Transforms to the Max tutorial, but I feel it needs further exploration, due to possibilities it offers. Its a rather fun technique, and I would love to see it used more and expanded to other transforms as well.

In this tutorial, you may need the following transforms:

- tile_hlp by Zy0rg
- bTransform from bSeries pack by Michael Faber

They are **not necessary** to complete this tutorial, but will contribute to obtain better results.

The three works below are examples of what you can achieve with the technique described in this tutorial.

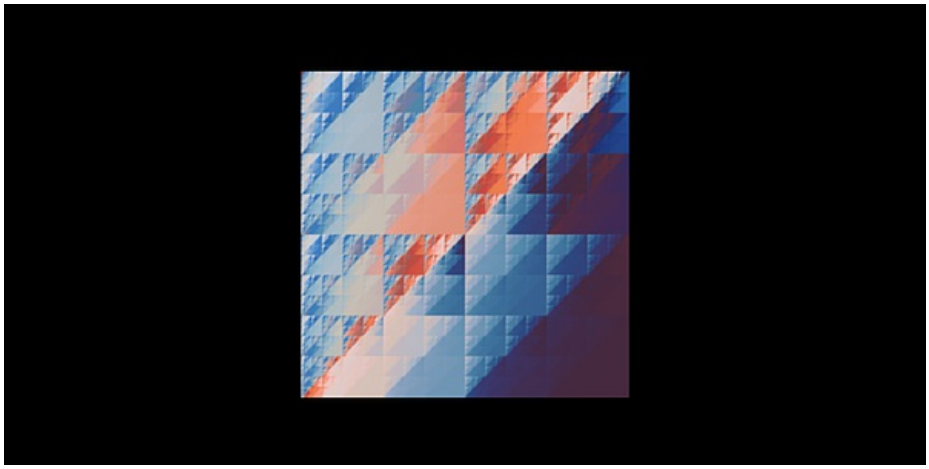


Example 1: Simple Tiled Square

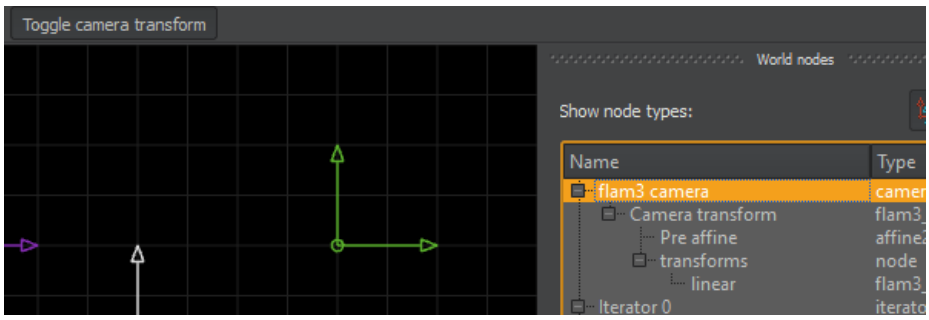
For this technique to work, we need something square to start with. In this example, we will work with a square linear tile. If you don't know how to make one:

- Linear tile tutorial
- Parameters

We start with a square:

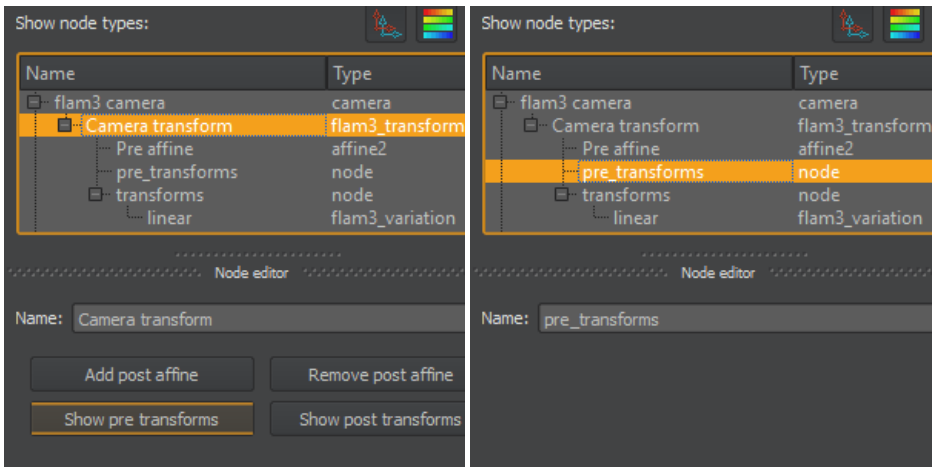


Go to the **World Editor** (ctrl + e) and **toggle camera transform**.

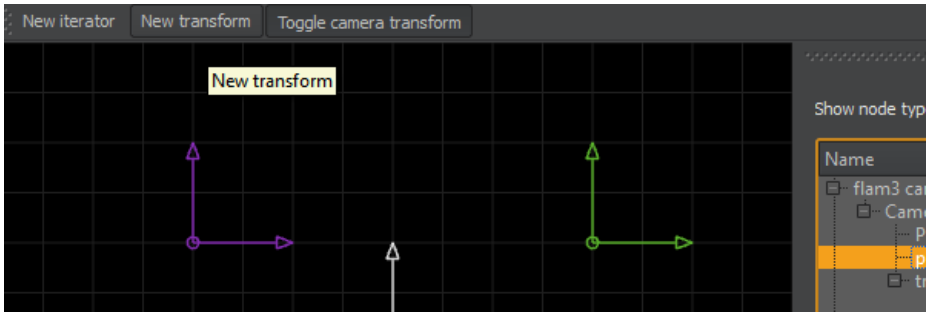


First, we will work only with **pre-transforms**:

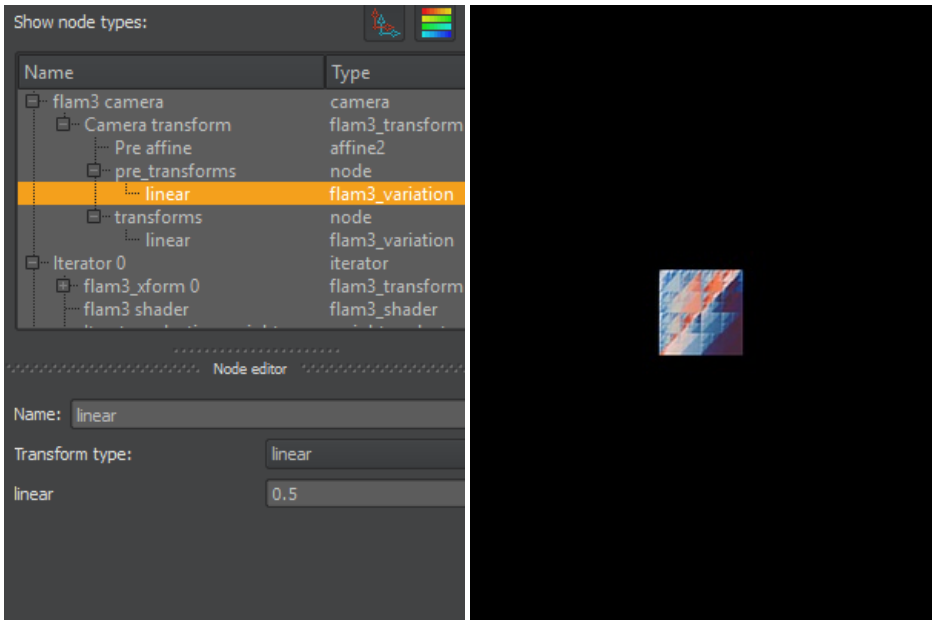
- Click on **Camera Transform** node of the Camera transform.
- In the node editor, click on **Show pre transforms**.
- When the pre transform node shows up, click on it.



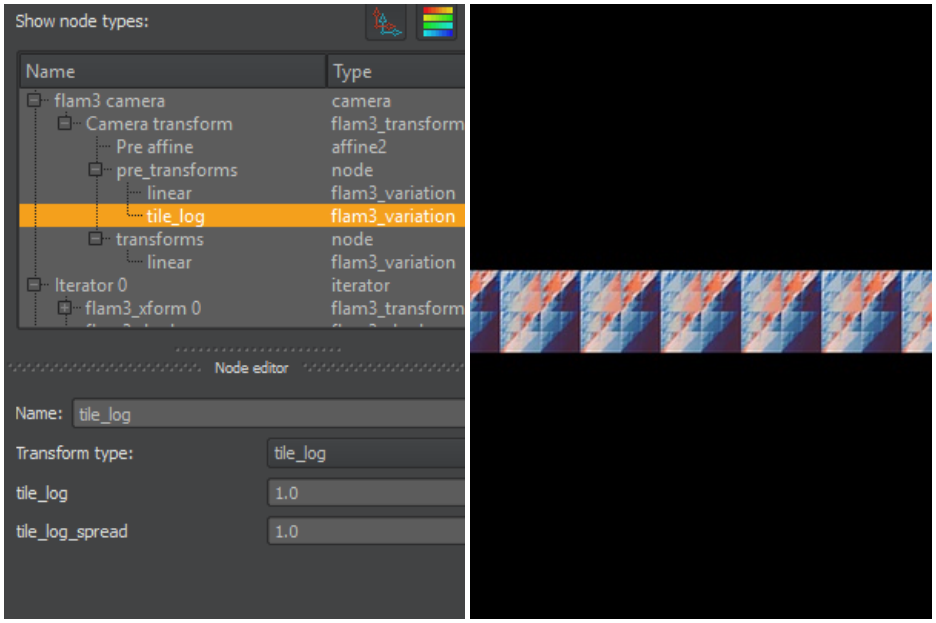
Now, we will start adding transforms to it. To add a new transform, click on the **New transform** button, or use the ctrl + t shortcut.



Step 1: add a linear pre transform. This will be used to scale the pattern. For the linear tiles we are using, you will most likely need 1, 0.5 or 0.25 and so on. This is used to scale down the tile, and I'll show a few examples of it later on. If you use the parameters I provided, you will need 0.25.

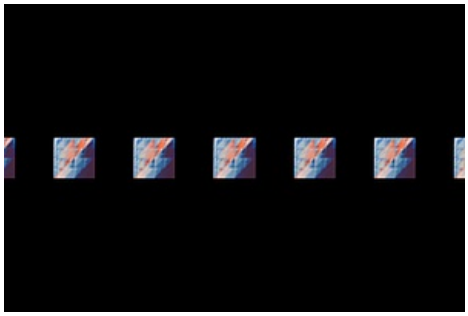


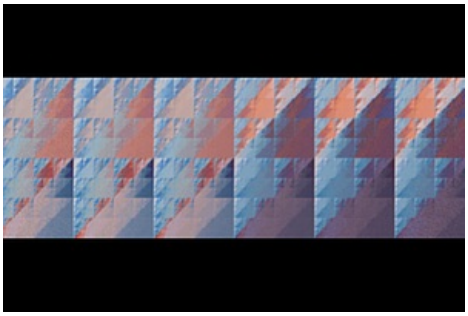
Step 2: add a tile_log pre transform. This transform tiles the square horizontally, as you can see on the preview.



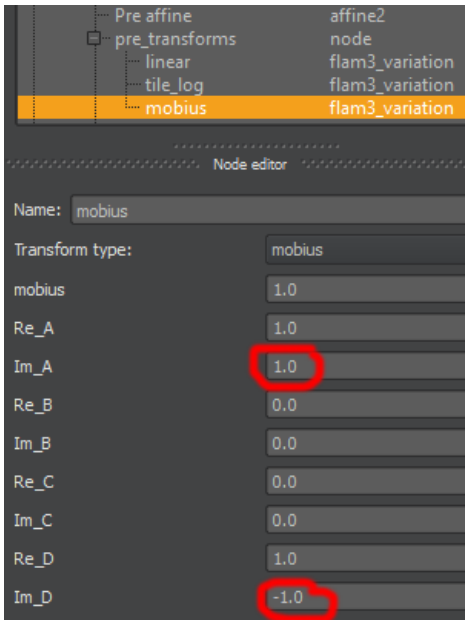
For this transform to work properly, your square must have side = 1 unit. This is why we use the linear from step 1, to scale down (or up) the square to fit into this tile.

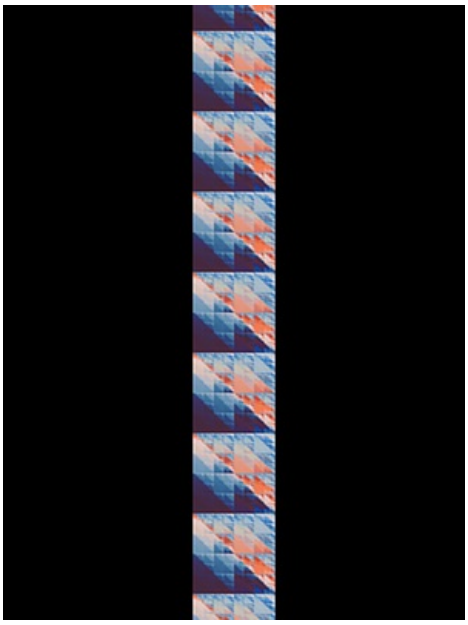
Below, two examples of wrong scale. On the left, the square is too small. Go back to the linear and double its value (if it was 0.25, set it to 0.5 and so on). On the right, the square is too big. Go back to the linear and reduce its value by 50% (if it was 0.5, change it to 0.25).



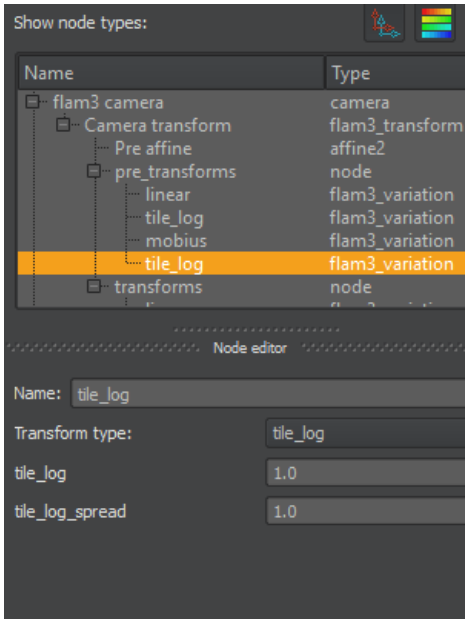


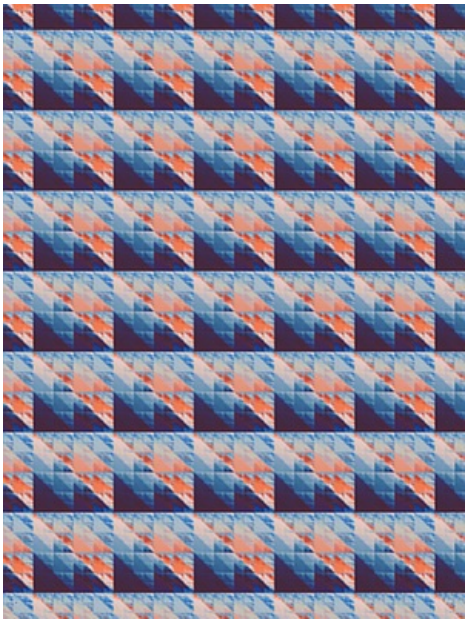
Step 3: add a mobius pre transform. Set Im A variable to 1 and Im D variable to -1 as shown below. This will rotate your pattern by 90 degrees.





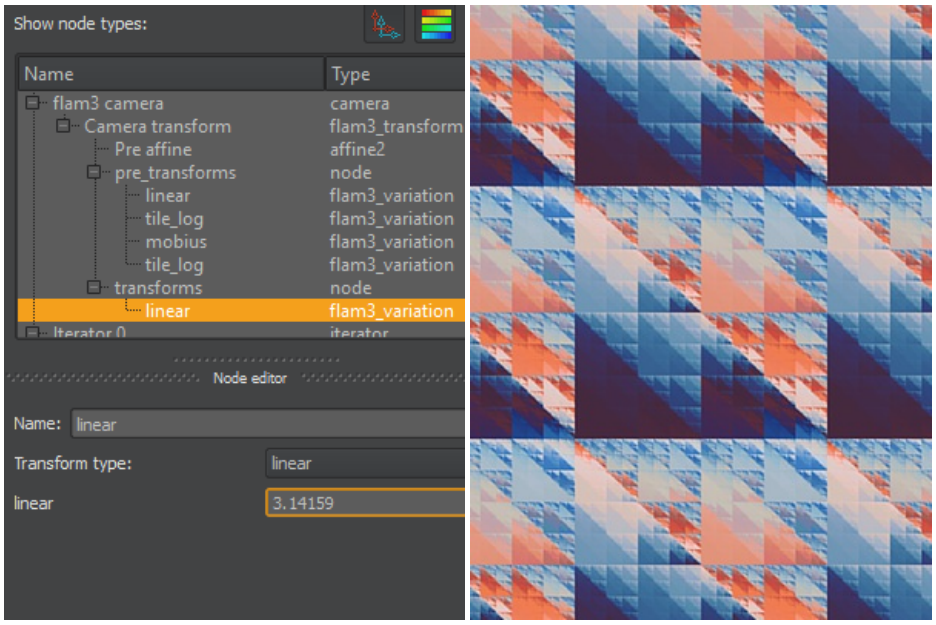
Step 4: add another tile_log transform.





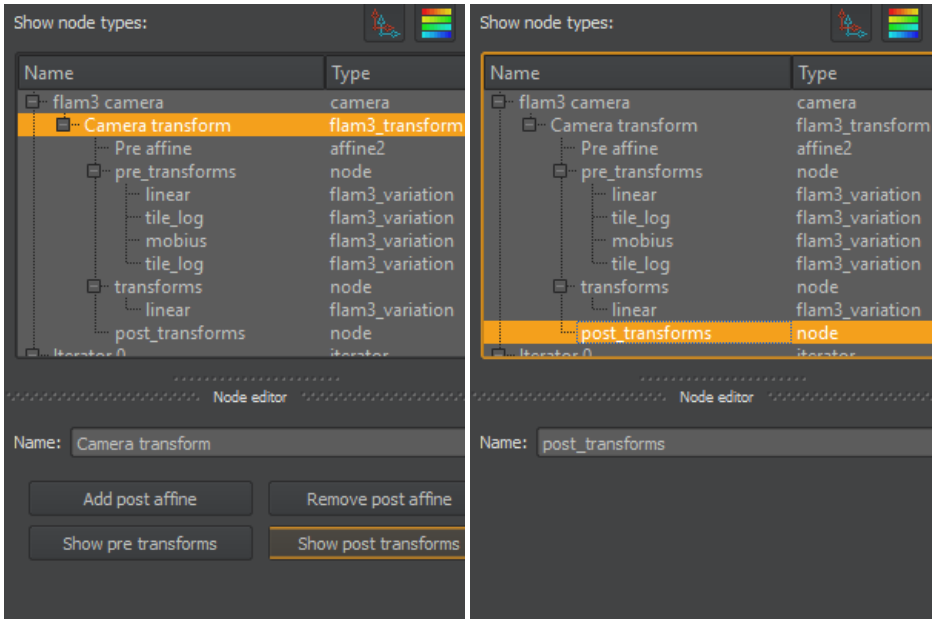
We are done with pre transforms, and now will edit the actual transform.

Step 5: change the linear transform value from 1 to 3.14159. For foci to work seamlessly here, we need a multiple or a fraction of PI. As Chaotica does not allow pasting numbers with many decimals, I usually break it down in two transforms: one linear value is just PI itself, and another works as multiplier.

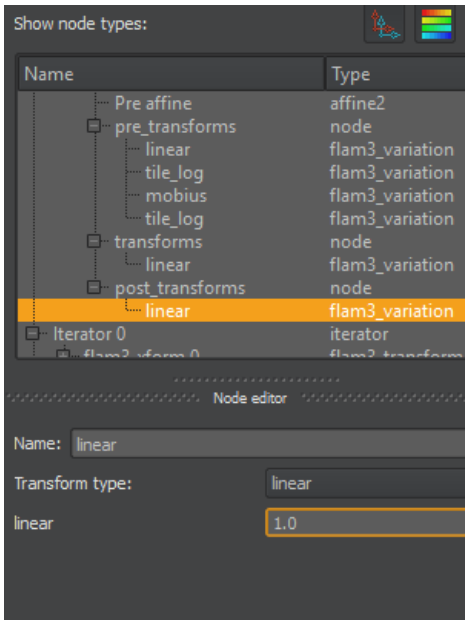


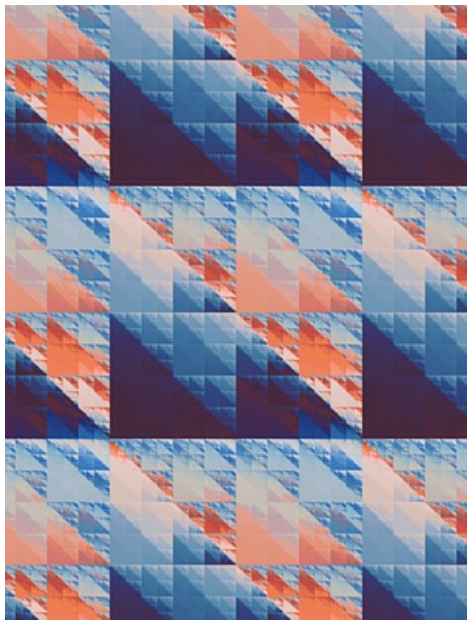
This was the only transform we will have. Lets now move to the post transforms.

- Click on **Camera Transform** node of the Camera transform.
- In the node editor, click on **Show post transforms**.
- When the pre transform node shows up, click on it.

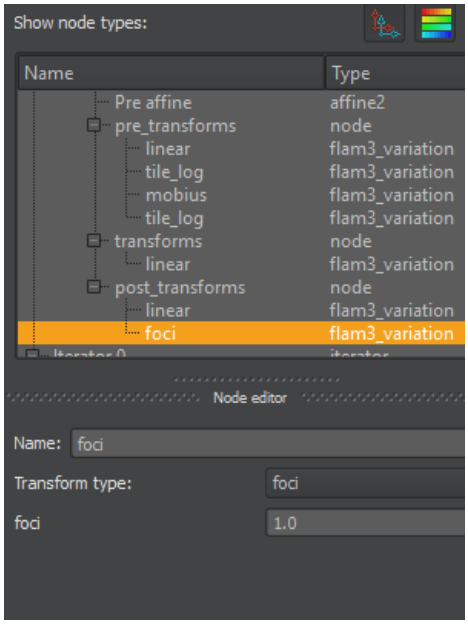


Step 6: add a linear post transform. Leave its value as 1 for now. This is the transform we will use as multiplier later on.

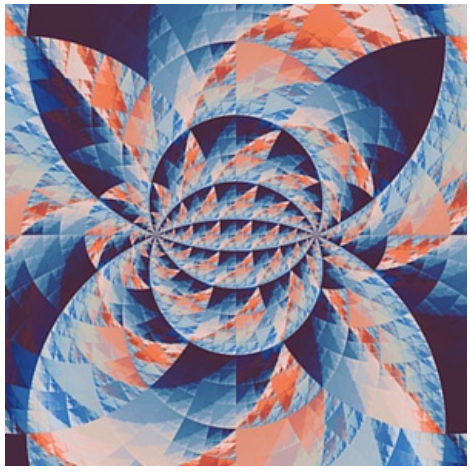




Step 7: finally, add a foci post transform. This is the one that makes it cool.



Now, lets back to the linear added in step 4. Change its value from 1 to 0.5, 0.25, 0.125 and so on. Below, two examples: 0.5 (left) and 0.25 (right).



Example 2: Not So Square Tile

In this section, we will take a look at two not so square cases.

We will start with linear tile with a shared linked xform.

- Video tutorial
- Parameters

Here goes a preview of the structure (featuring a lovely gradient by Boxtail):

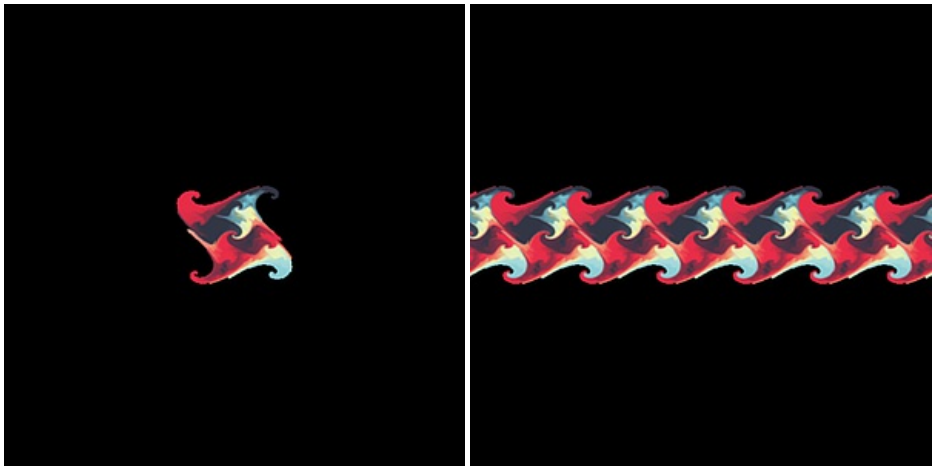
Notice that while it is not square, it looks like it would tile seamlessly.

Lets repeat the steps from Example 1.

Step 1: Toggle Camera Transform and add a linear pre transform to it. Set linear to 0.5

Step 2: add a tile_log pre transform.

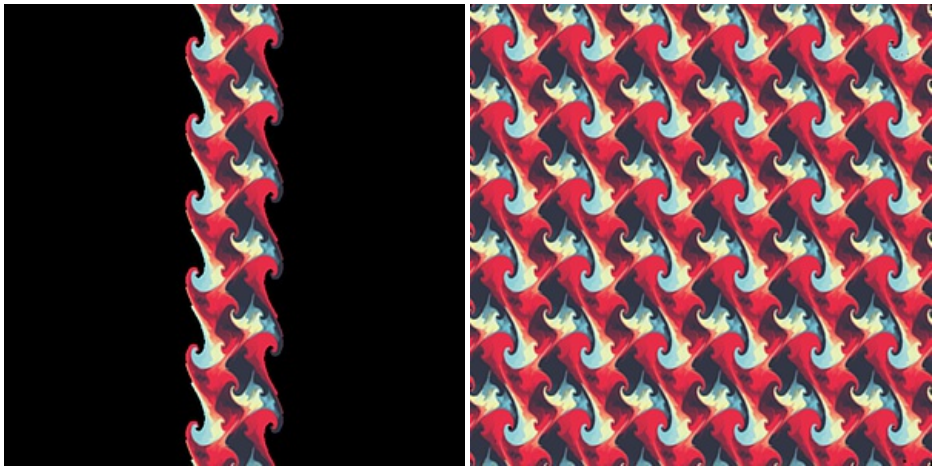
Below, result of step 1 (left) and step 2 (right):



Step 3: add a mobius pre transform. Set Im A variable to 1 and Im D variable to -1.

Step 4: add another tile_log transform.

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



Step 5: change the linear transform (regular transform) value from 1 to 3.14159.

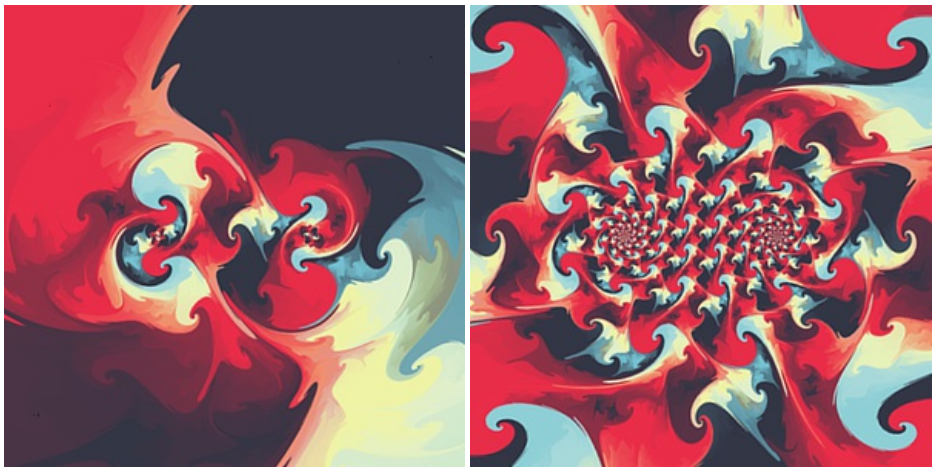
Step 6: add a linear post transform.

Below, result of step 5 (left) and step 6 (right):

Step 7: finally, add a foci post transform.

Tweaking: go back to the linear post transform and change its value from 1 to 0.25.

Below, result of step 7 (left) and the tweaked result (right):



Example 3: Even Less Tileable

The original idea of this example belongs to Boxtail, I just implemented it.

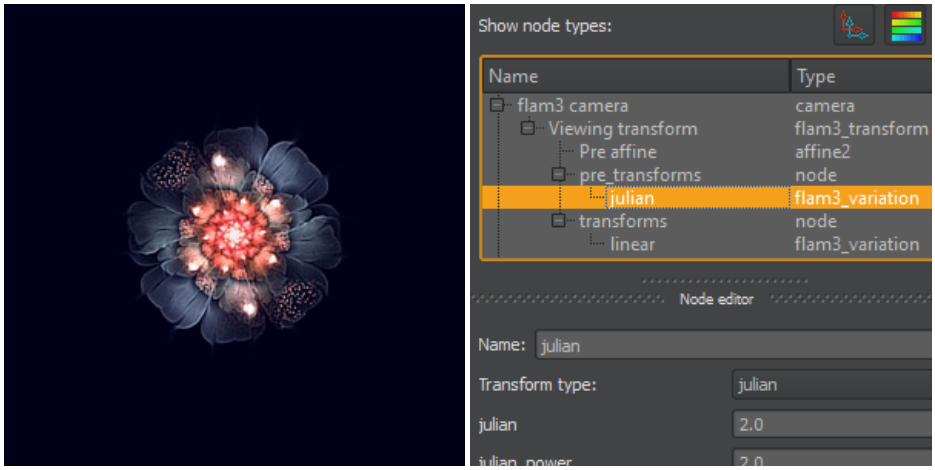
To make a basic bloom, you may use this awesome tutorial by Lindelokse.

For this tutorial, I will work with this bloom:

Step 0: Before we start, we need to do some preparations.

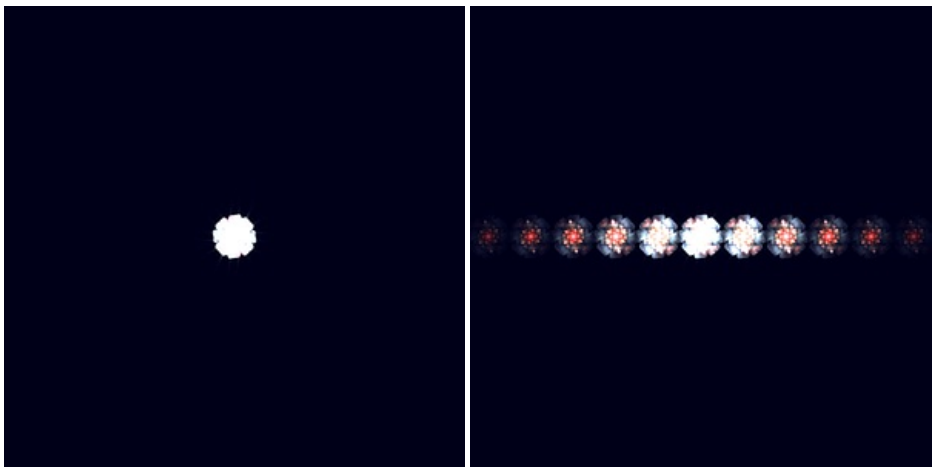
First, set opacity of all transforms except for the spherical+linear one to 0. You should then have a round-ish bloom such as pictured below.

Second, on camera transform, you may have placed julian as a regular transform. Replace that one with a linear and place julian as your first **pre transform**.

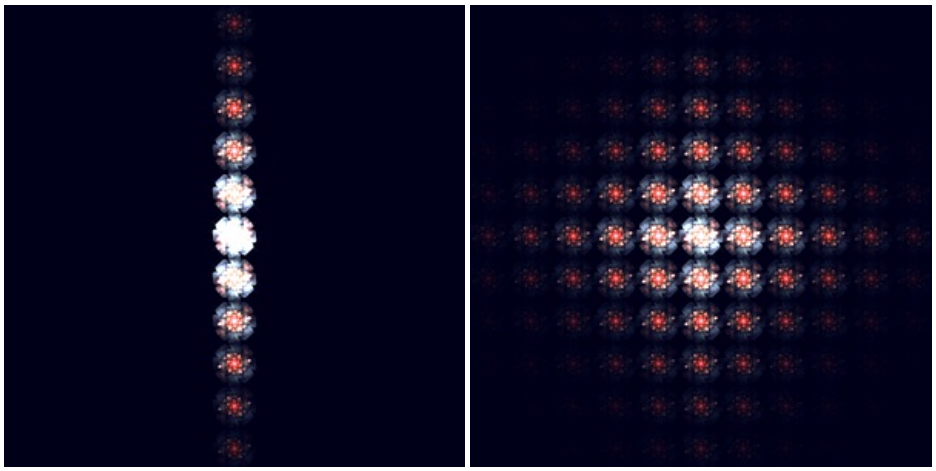


Now, we can just go through steps 1-7.

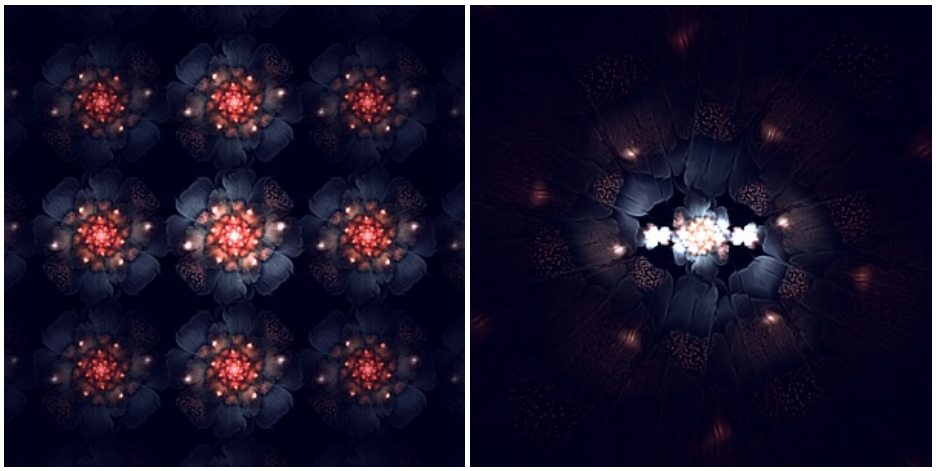
On left picture, the bloom after **Step 1**. In this example, I used 0.2 linear value. Right, after adding tile_log on **Step 2**.



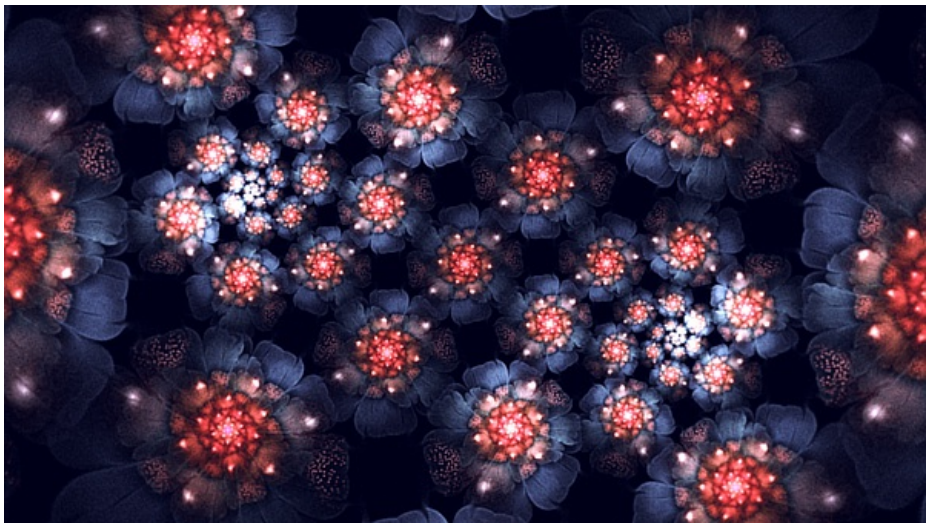
Then, add a mobius pre transform (**Step 3**, and left picture), and another tile_log pre transform (**Step 4**, right).



Finally, set up the linear transform and the linear post transform (**Step 5** and **Step 6, left preview). And finish with a foci post transform (Step 7**, right preview).



After some tweaking, I ended up with this:



Example 4: Whatever You Want

First, lets review the transforms we have used in previous steps.

We start with a square and, to achieve the tile + foci effect, we add the following transforms to camera:

1. Linear
2. Tile_log
3. Mobius
4. Tile_log
5. Linear
6. Linear
7. Foci

Observe that while I've spread them between pre, regular and post transform, this is not necessary: for example, you can add them all as pre transforms to camera.

In order to apply this technique to any fractal, we need first to convert it to square.

In this example, I will work with elliptic splits.

If you are willing to do the same, you can try any of those tutorials:

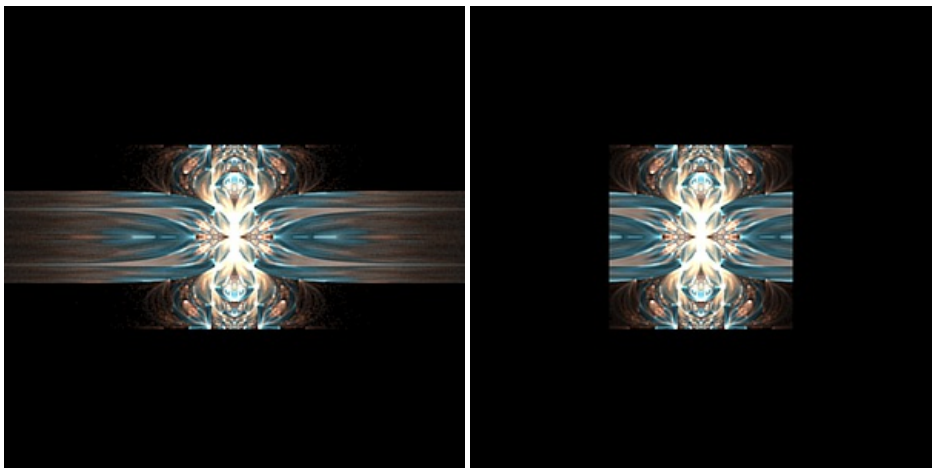
- Elliptic Splits (Apophysis)
- Basic Elliptic Splits (Apophysis)
- Elliptic Splits (Chaotica tutorial)

This is not a must: in fact, you can start with a Chaotica random =) The only requirement is an empty camera transorm.

The first thing we need to do is to make a square out of it. There are several transform combinations that lead to a square. I will use bipolar + crop combo:

- First, add a bipolar pre_transform to camera
- After that, add a cropn pre_transform, with power = 4, radius 1, scatterdist = 0, zero = 0, no edge = 1

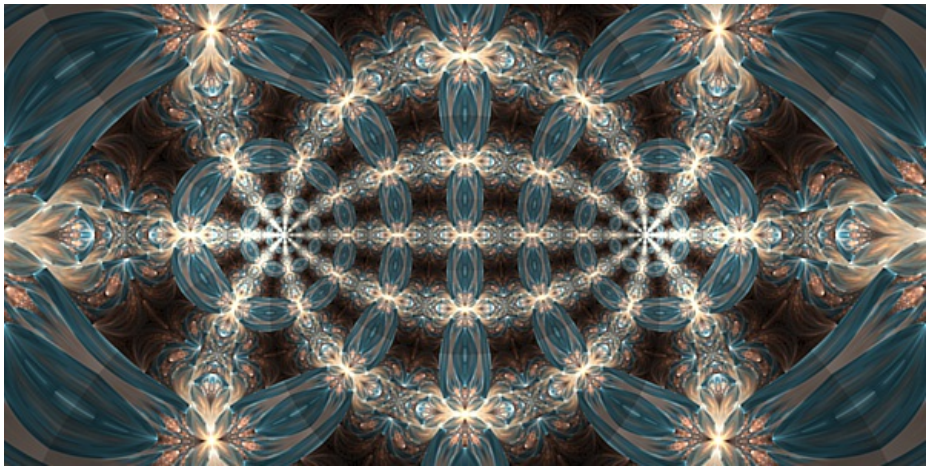
Below, the bipolar (left) and then same after the crop is applied (right).



Now, we just add the transforms as described in previous examples:

1. Linear = 1, 0.5, 0.25, 0.125 and so on depending on parameters
2. Tile_log
3. Mobius with Im A = 1 and Im D = -1
4. Tile_log
5. Linear = 3.14159
6. Linear = 1, 0.5, 0.25, 0.125 and so on depending on parameters
7. Foci

And you will get something like this:

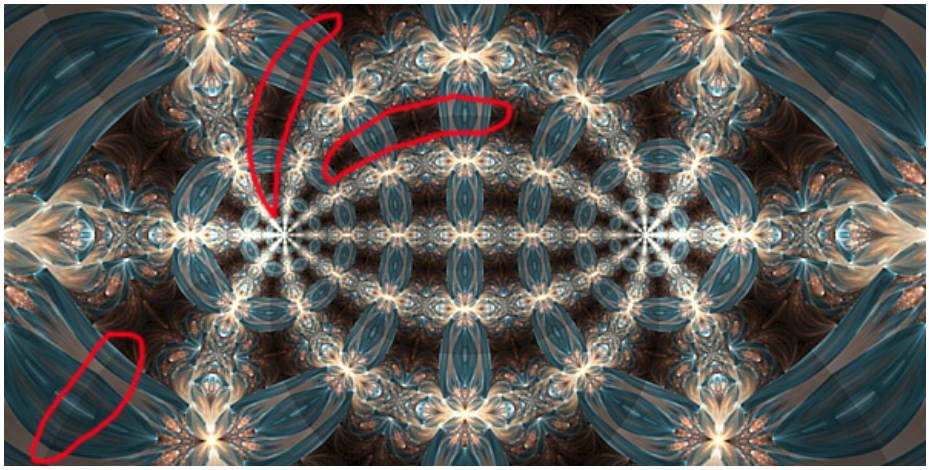


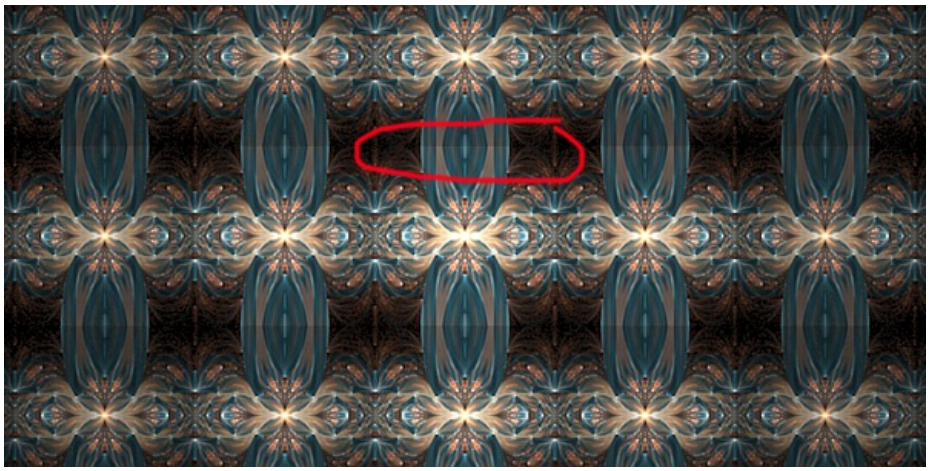
There are a few other ways to make your fractal into a square:

- Bipolar + crop (as described above)
- Elliptic + crop
- Julian (specially with power = 2) + crop
- Just crop =)
- Hemisphere + ngon (power = 0)
- Scry + ngon (power = 0)

Fancy Details - Part 1: Smooth Transitions

Take a closer look at the pictures below. You can see that there are some not smooth transitions. This is because when we tile a square-ish element, the tiles will have slightly different brightness.

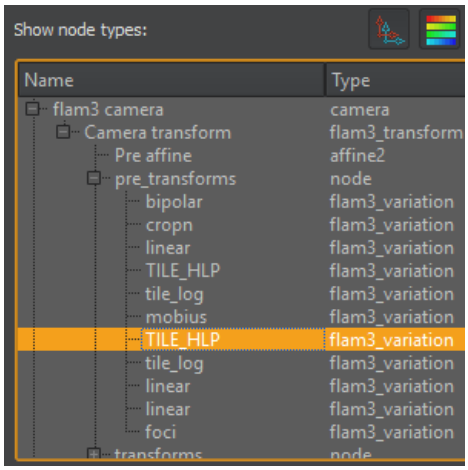


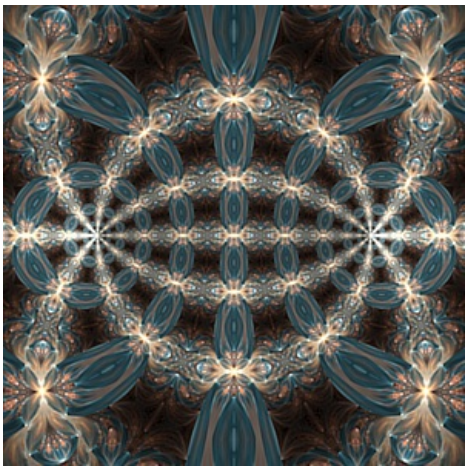


This can be avoided by using tile_hlp transform before every tile log we use. Our camera transforms list would turn into something like:

1. Linear = 1, 0.5, 0.25, 0.125 and so on depending on parameters
2. **Tile_hlp**
3. Tile_log
4. Mobius with Im A = 1 and Im D = -1
5. **Tile_hlp**
6. Tile_log
7. Linear = 3.14159
8. Linear = 1, 0.5, 0.25, 0.125 and so on depending on parameters
9. Foci

Tile_hlp smoothes the transitions between the tile pieces, so we end up with something like:





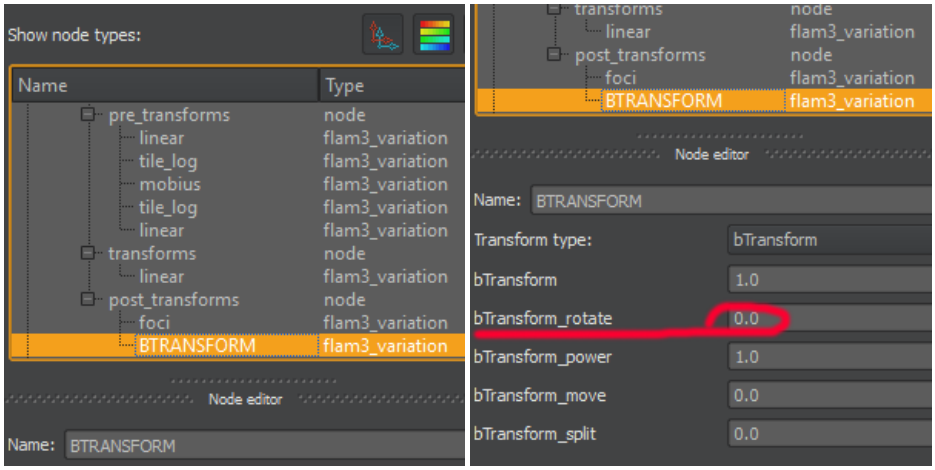
Fancy Details - Part 2: Rotations

Now, we will learn how to rotate the pattern. In this example, I will use the fractal from example 2:

The first type of rotation can be achieved by adding **bTransform** after foci:

1. Linear
2. Tile_log
3. Mobius
4. Tile_log
5. Linear = 3.14159
6. Linear
7. Foci
8. **BTransform**

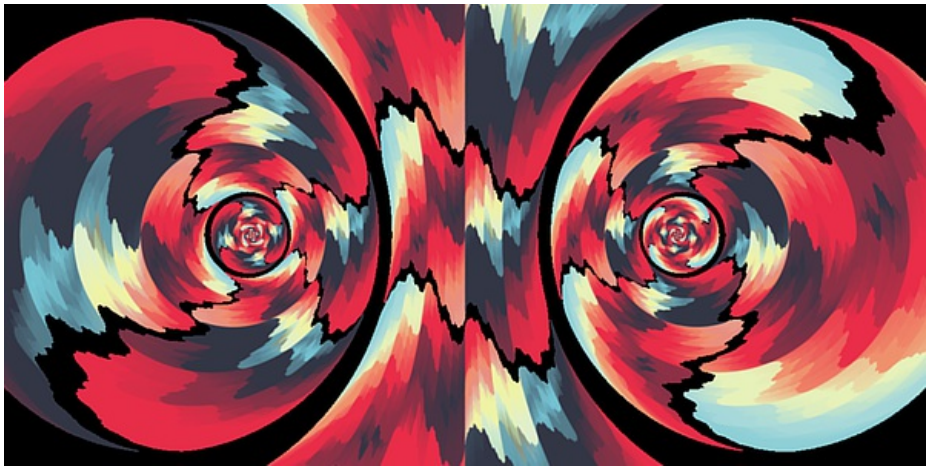
Now, just change the **rotate** variable:



Compare rotate = 0 (left) and rotate = 0.3 (right):

Another transform that enables rotations is **mobius**. Mobius should be added before foci (and after the second tile_log). It can go either before or after or between the linears, it doesn't matter.

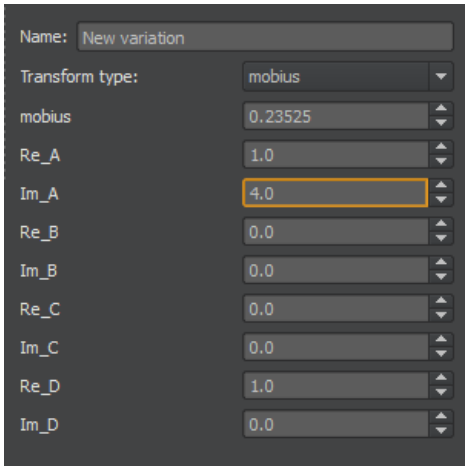
Here, we will use the fractal below as example:



1. Linear
2. Tile_log
3. Mobius
4. Tile_log
5. **Mobius**
6. Linear = 3.14159
7. Linear
8. Foci

Once you add mobius, try changing the Im A and Im D variables. Then, to remove overlaps, decrease mobius amount.

For example, below, I set up bot variables (to find the right amount of mobius, I just kept changing it by small amounts until there was no overlaps):



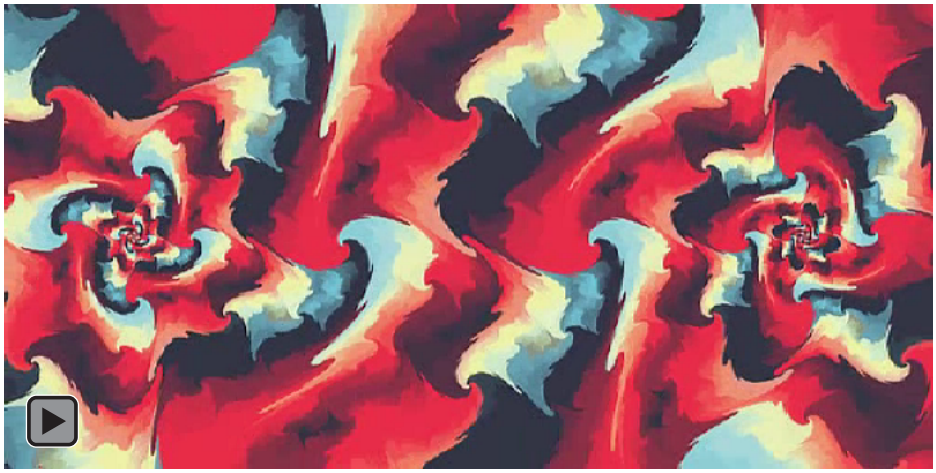
A few combinations that work: mobius = 1 and Im A = 1 (left) and mobius = 0.6 and Im A = 0.33333333 (right).

Fancy Details - Part 3: Looping Animations

In this section, we also have two tricks.

The first one is using bTranform to create perfect loops (see the Fancy Details - Part 2). Observe that, when you set **rotate** to Pi, the result is same as when you set it to 0 (or you may need some multiple of Pi).

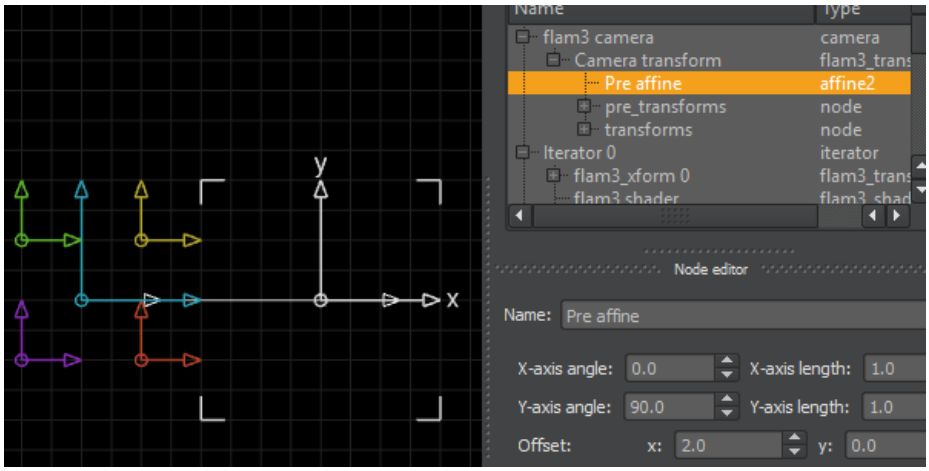
For example, in the gif below, **rotate** begins with 0 and goes up to Pi. The loop is seamless:



Now, let's learn how to seamlessly animate the movement of the original square.

We start with this fractal:

We will first try to just move camera transform 2 units to right, like this:



Notice that, while the pattern matches, the brightness does not:

To make the animation seamless, add a **modulus** transform before everything else:

1. **Modulus**
2. Linear
3. Tile_log
4. Mobius
5. Tile_log
6. Linear = 3.14159
7. Linear
8. Foci

With this change, the same animation will look like this:

No more sharp brightness transitions, as you can see.

As usual, those are only a few ideas, and the rest is left for you to explore.