tatasz's

LINKED TRANSFORMS

WED MAY 27, 2015, 6:28 AM

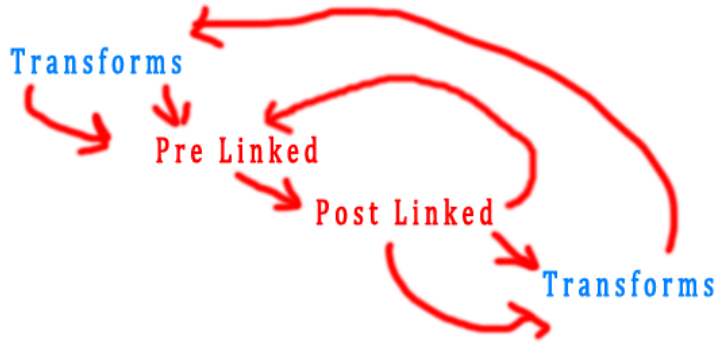
This is the second tutorial of the Xaos series, and is about linked transforms, both pre and post.

You may also check: **Linked Xform Tutorial**

There is always a pair of transforms: first transform is pre linked to second, and second is post linked to first (or just linked). A pair of linked transforms has the following xaos settings:

1. Pre linked transform, "from" weight modifiers can assume any values, as long as condition 2 is respected (so "from" pre linked transform itself is 0)
2. Pre linked transform, "to" values are all 0, except the one to the post linked transform
3. Post linked transform, "from" values are all 0, except the one from the pre linked transform
4. Post linked transform, "to" values can be anything, as long as condition 3 is respected (so "to" post linked transform itself is 0)

On the fluxogram below, the linked pair of transform is in red, while other transforms of the fractal are in blue:



Note: a post linked transform can send points to the pre linked transform also, because that will not go against conditions 3 and 4 above.

The variations of the first transform act as pre_ variations to the second transform. The variations of the second transform act as post_ variations to the first transform. Lets take a look at a simple example.

Bubble with post_curl - a bubble with pre_blur, with added post_curl (c1=1)
Bubble with linked curl - instead of post curl, a transform with curl (c1=1) is linked to a transform with bubble and pre_blur

Now try guessing which of the pictures below is post_curl and which is linked curl

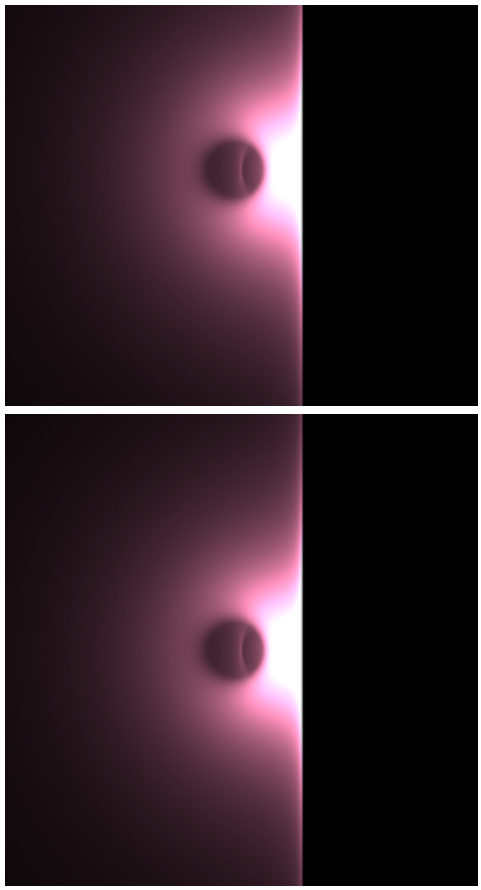
Details

Uploaded on	May 27, 2015
File Size	0 bytes
Submitted with	Sta.sh Writer

Stats

Views	971
Comments	1

tatasz's



While this is not universal, i prefer using linked transforms to pre_ and post_ variations, mostly because of better usability and flexibility.

- Using pre_ and post_ variations requires many plugins, and Apophysis handles large number of plugins badly
- Not all pre_ and post_ variations are avaliable as plugins
- Ones avaliable as plugins may not have a 64bits version
- Or one just can't use plugins (Mac users for example would have troubles with the dlls)

- Linked transforms have an extra affine transform to tweak around - something not easy when using pre_ and post_variations
- Linked transforms can hold pre_ and post_variations and also a whole mix of regular variations
- You can link a transform to more than one transform
- Linked transforms can be easily modified into other xaos structures

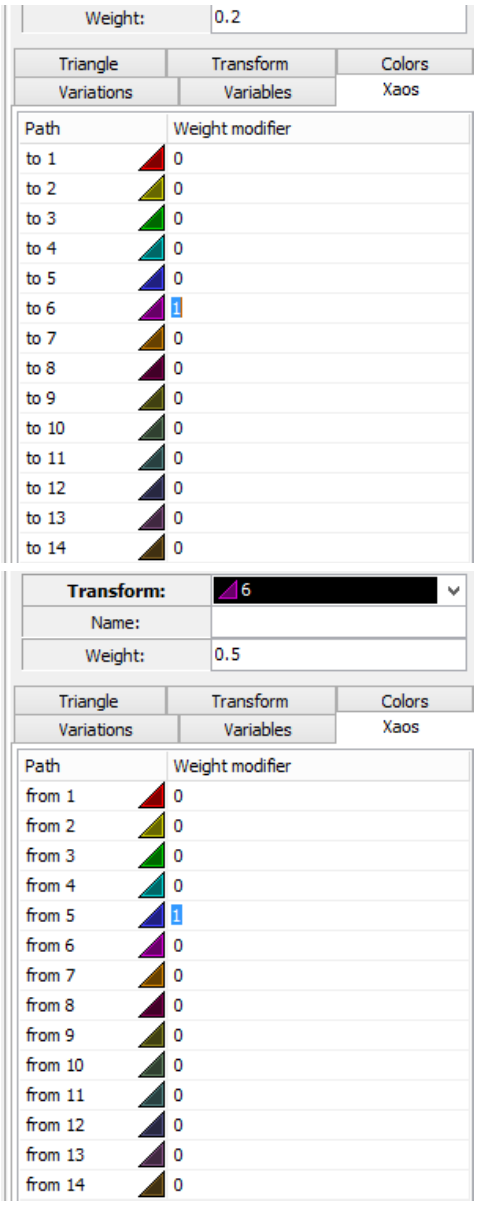
ADDING A POST LINKED TRANSFORM

To add a post linked transform to a transform, select the transform - xform 5 here - and click the "add linked transform" button. This automatically adds a new transform with color speed one, sets the opacity of the selected transform to 0 (so it is not visible, as our final "image" will be the linked transform) and sets the following xaos values:

- transform: all "to" weight modifiers are set to 0, except one to the linked transform, which is set to 1
- linked: all "from" weight modifiers are set to 0, except one from the transform, which is set to 1
- linked: all "to" weight modifiers are set to same values as first transform had before the linked xform was added

You can also add a linked transform by setting all those values manually:

tatasz's



The opacity 0 for first transform and the color speed 1 for the second are also part of the usual linked transform setup, but not at all a must.

ADDING A PRE LINKED TRANSFORM

There is no shortcut button for this one, so you just have to set it up manually, in a manner similar to the described above:

- transform: all "from" weight modifiers are set to 0, except one to the pre linked transform, which is set to 1
- pre linked: all "to" weight modifiers are set to 0, except one from the transform, which is set to 1
- pre linked: all "from" weight modifiers are set to same values as the transform had before the pre linked xform was added

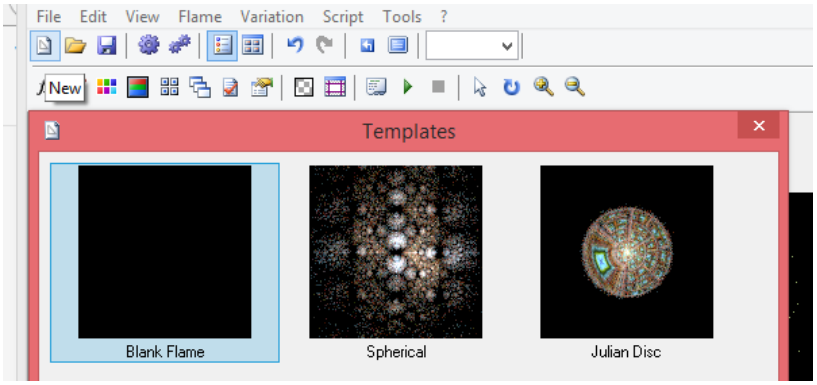
To avoid setting it up manually in fractals with many transforms, you may also use the following hack: add a post linked transform using the button, then carefully copy all settings of the transform to the post linked one - variations, shift, scale, rotation. Now, your original transform is pre linked to an exact copy of itself, so you can replace whatever you had on it with what you wanted on the pre linked transform.

Example: we have a spherical transform (not rotated, scaled or shifted), and want to add a pre linked loonie to it.

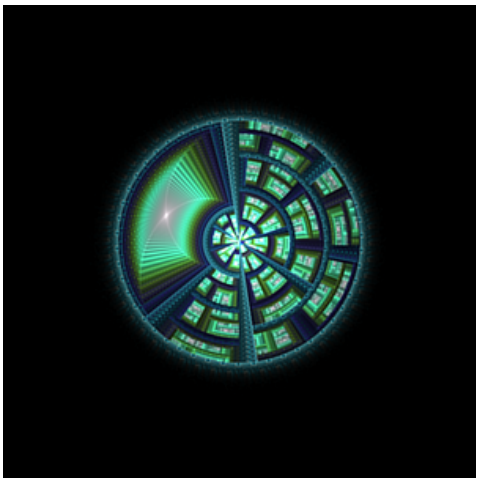
tatasz's

EXAMPLE: DISC

To make a basic disc design, click on the "New" button and select "Julian Disc":

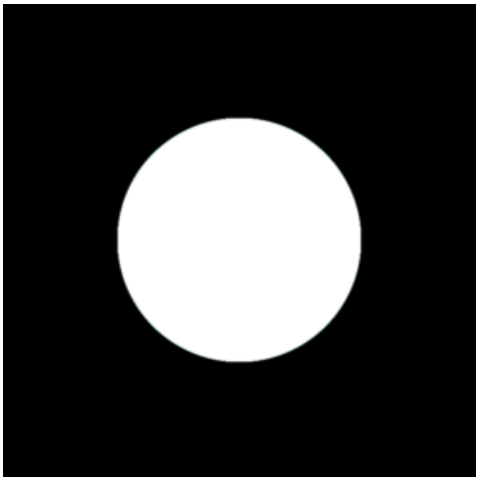


You should also take a look at this tutorial about Julian Disc: **Apo-Tuto_Disc-Julian**



This framework has quite a lot of empty background, and we will fill it with disc julian patterns using linked transforms. The basic idea is to "inverse" the disc, so it fills the outside, and then make both original disc and the inverted version visible.

For this, we will use spherical. Lets take a look at how spherical works. The right picture features a circleblur. The left picture, a circleblur with a linked spherical transform:



tatasz's

Now if we link another spherical to the linked transform in the example above, we will again have a circle (we invert the circle, and then invert it again, back to the original shape).

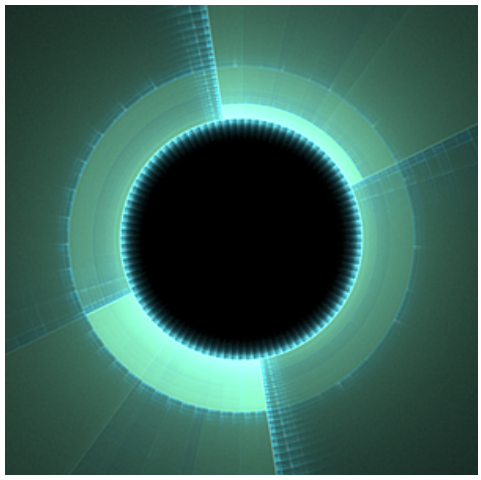
Lets go back to the Julian Disc and take a look at the xaos weight modifiers. All values are 1 for both transforms:

Following the description above, we can draw the weights fluxogram for when we add a linked spherical transform:

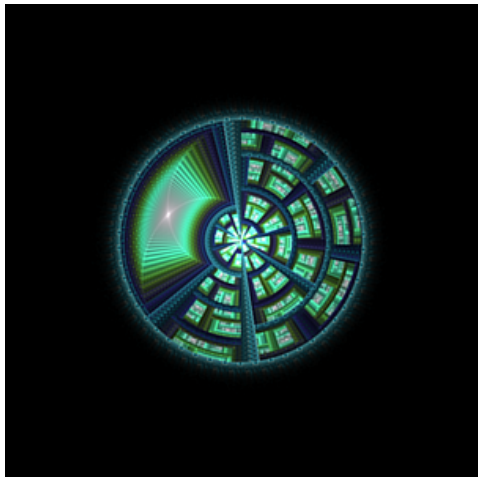
Before, disc transform was sending information both to julian and to itself. Now, it will send points to spherical (post linked transform), which will then send points to julian and to disc (as disc did before the addition of the linked transform). Spherical will recieve information only from disc. Julian will still send information to disc and to itself, but now will recieve information from spherical instead of disc.

As spherical does an inversion, this sounds like something that could break the

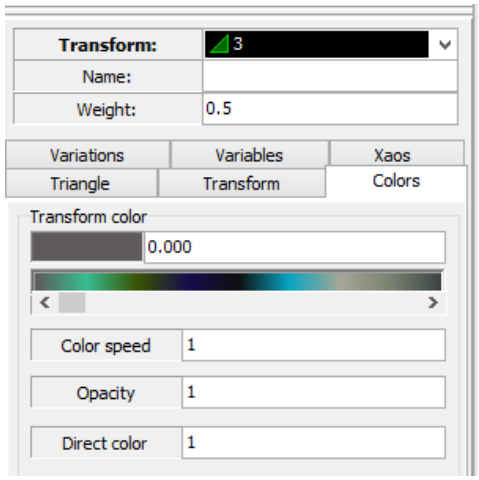
tatasz's



To get the original shape back, select the 3rd transform, the linked spherical we just added, and add another linked transform with spherical. How does it works: the first spherical transforms the circle (disc transform) into an inverted circle, and the second linked spherical transforms it back into a circle - so disc keeps working properly and the patterns are not changed:



But we did not do all this work to just have a result identical to the starting params. Go to transform 3 (the first linked spherical) and change its opacity to 1 on the color tab to make the inverse of Julian Disc pattern visible:



tatasz's

You can take a look at the parameters here: **Julian Disc with inverse**

1 COMMENT

Linked Transforms

by tatasz

fractal2cry **Edited** Nov 29, 2015

awesome

Reply

Add a Comment: