Shader Math Explained

Here I explain the math used on this shader:

<https://www.shadertoy.com/view/Mdf3zM>

Here the author explains a little bit more:

<https://reindernijhoff.net/2014/05/escher-droste-effect-webgl-fragment-shader/>

The author is: [Reinder.Nijhoff@infi.nl](mailto:Reinder.Nijhoff@infi.nl)

This is what he said on his email:

I think that I have done something like this:

h(w) = w^((2πi + log scale)/(2πi)) = w^(1 - i \* log scale / 2π) = w^( 1 + i \* sn )

where: log scale  = L(256)

and

w^( 1 + i \* sn) = exp( ln(w) \* (1 + i \* sn) ) = exp( (ln|w| + i \* arg(w)) \* (1 + i \* sn)) =

exp( (lnr + i \* th) \* (1 + i \* sn) ) = exp ( (lnr - th \* sn) + i \* (lnr \* sn + th) ) =

exp (lnr - th\*sn) \* exp ( i \*(lnr \* sn + th) )

so when I want to go back to Cartesian coordinates I get a coordinate that corresponds with a vector of length exp(lnr - th\*sn) rotated (sn \* lnr + th) around the origin.

(NOTE: I asked him about this: (0.4/256.)\*deformationScale but he didn’t reply on that, he might not remember)

## This is my explanation of what he did:

First (unconfirmed) I believe he didn’t use the complex plane, this allows him to do some extra tricks with his math.

He started from Escher’s formula:   
h(w) = w^((2πi + log scale)/(2πi))

Then he applied the factorization we know to arrive to this one:

 w^(1 - i \* log scale / 2π)

Then he expressed “sn” as a negative quantity on his code   
float sn = -log(deformationScale)\*(1./(2.\*3.1415926));

This allows him to write:

w^( 1 + i \* sn )

Now you can use the trick: e^ln(x)= x (i.e. apply exponential and log)

So basically: W^a = exp[L(W^a)]

Then by property of logs: L(W^a) = L(aW) = L(Wa)

exp( ln(w) \* (1 + i \* sn) )

Explanation of what he did here:

exp( (ln|w| + i \* arg(w)) \* (1 + i \* sn))

so, this is tricky… imagine “w” represents the complex plane (which is usually represented as “Z” in math books). So what you have is really:

w = r.e^i.th //note that “th” = arg(w)

Then if we take the natural log:

Ln(w) = Ln(r.e^i.th) = Ln(|r|) + i.th // where r = sqrt(x^2+y^2) 🡪 but this is actually a real number

So he went ahead and wrote this, which it’s not OK, as “w” should now be “r”

exp( (ln|w| + i \* arg(w)) \* (1 + i \* sn))

In any case, later he corrects and he does this, where now “lnr” is the modulo of the real numbers

exp( (lnr + i \* th) \* (1 + i \* sn) )

What follows is just distributing with imaginary numbers, signs change as (i^2 = -1) (we don’t do this in our code, as it’s not really needed, he just does it for convenience)

= exp ( (lnr - th \* sn) + i \* (lnr \* sn + th) ) =

Then he uses: e(a.b) = e^a.e^b

exp (lnr - th\*sn) \* exp ( i \*(lnr \* sn + th) )

with:

lnr = log(length(uv))

sn = sn = -log(deformationScale)\*(1./(2.\*3.1415926))

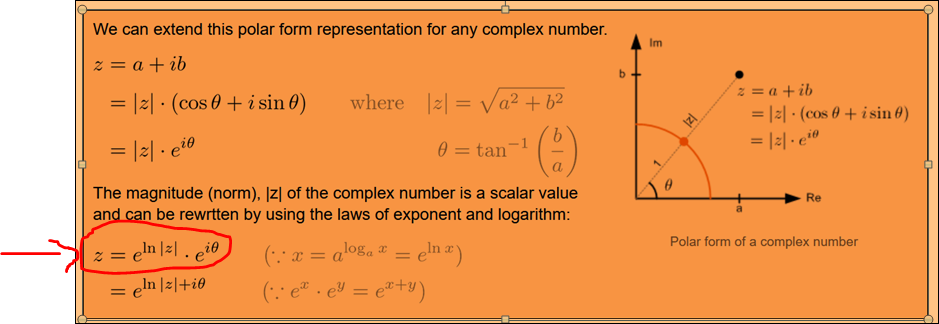
th = atan( uv.y, uv.x )

The reason he wanted to express it like this is what he says below:

“….so when I want to go back to Cartesian coordinates I get a coordinate that corresponds with a vector of length exp(lnr - th\*sn) rotated (sn \* lnr + th) around the origin.

In other words, he wanted to have it expressed as below:

<http://www.songho.ca/math/euler/euler.html>



Explanation: what he has on the last expression is the typical “rotation of a complex number”

<http://www.songho.ca/math/euler/euler.html>

