Clojure Art

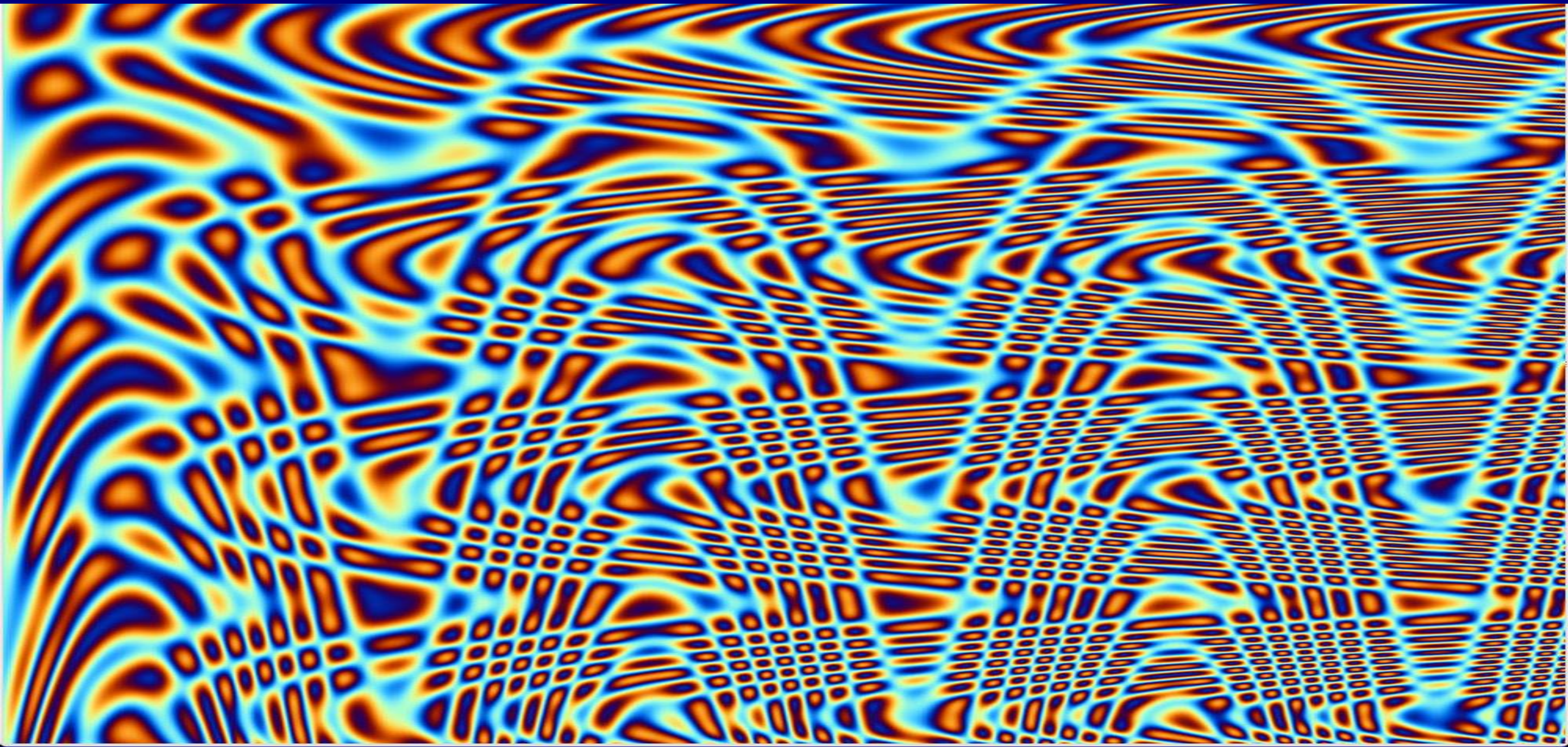
Karl Brodowsky

IT Sky Consulting GmbH

http://www.it-sky-consulting-com/

http://brodowsky.it-sky.net/

Some Fun



Disclaimer

- Images not created with Clojure are from Wikimedia Commons licensed under CC

How to get started

- Use frame (Swing) and draw:

```
(defn make-frame []
  (let [frame
    (doto (javax.swing.JFrame.)
          (.setSize (java.awt.Dimension. 1000 1000))
          (.setVisible true))]
   frame))
```

Drawing (pixelwise)

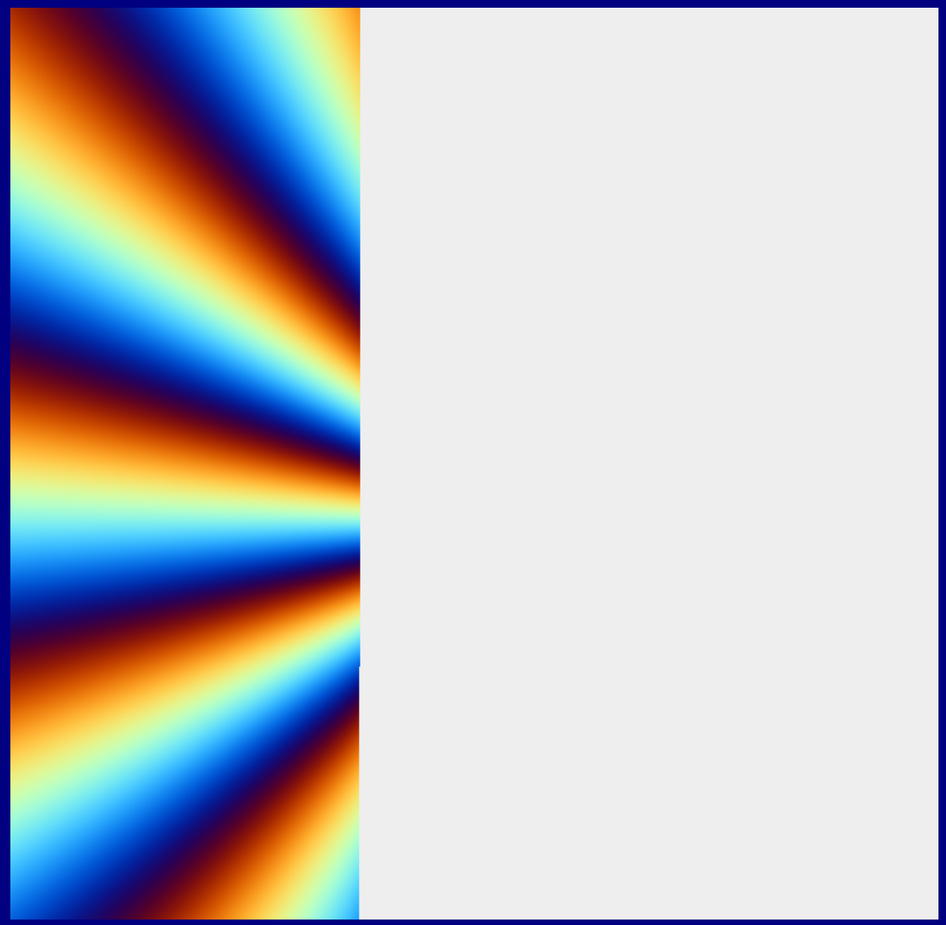
```
(defn draw-pixel [frame color x y]
     (let [gfx (.getGraphics frame)]
       (.setColor gfx color)
       (.fillRect gfx x y 1 1)))
```

Let’s get functional

- How to make a picture of a function?

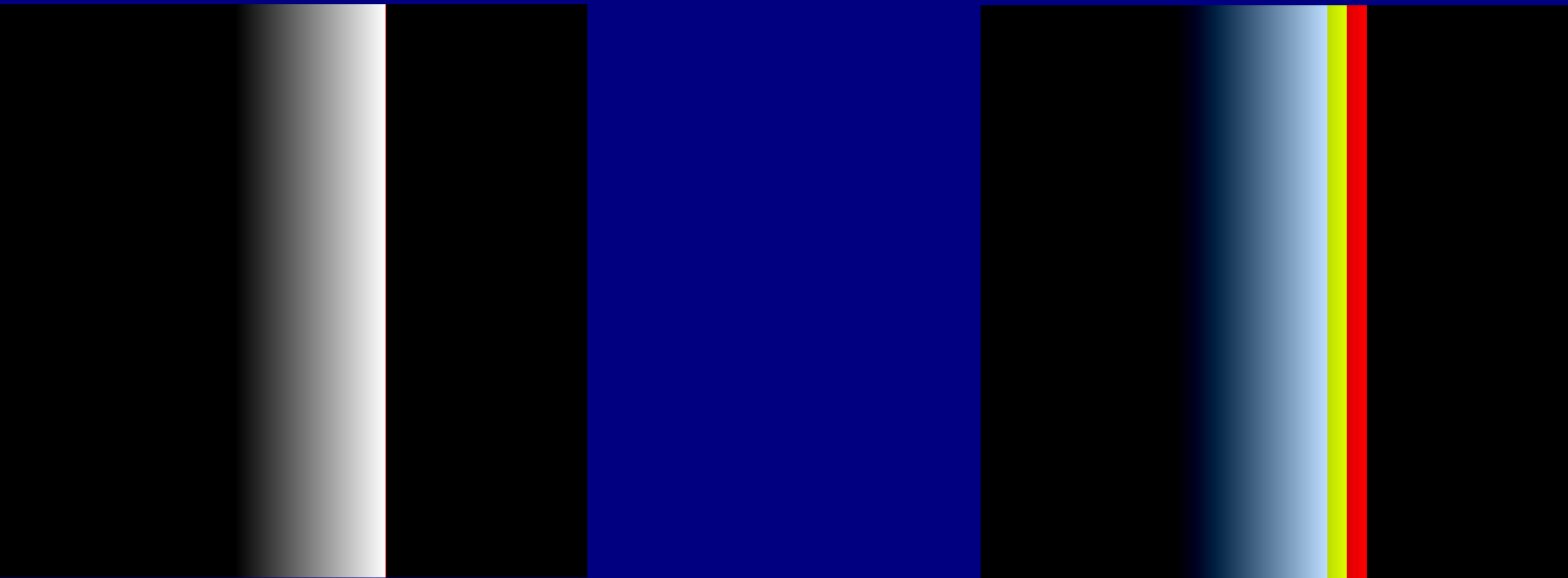
Challenges

- Exceptions: just try to ignore them...



Let’s get functional

- How to make a picture of a function? --> Start with the x-axis

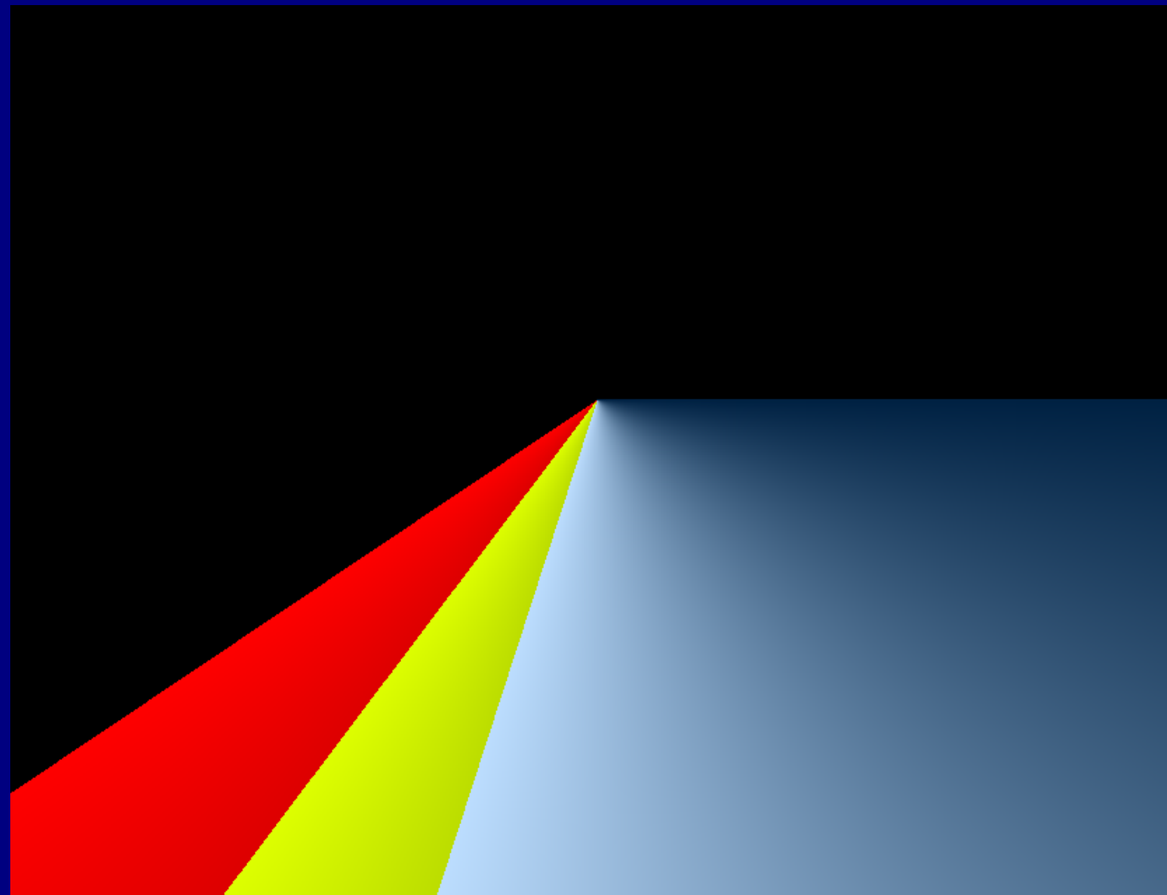


Naïve Approach

- Just create 3 functions
- f_r(x, y), f_g(x, y), f_b(x, y)
- Calculate colors
- Draw it...

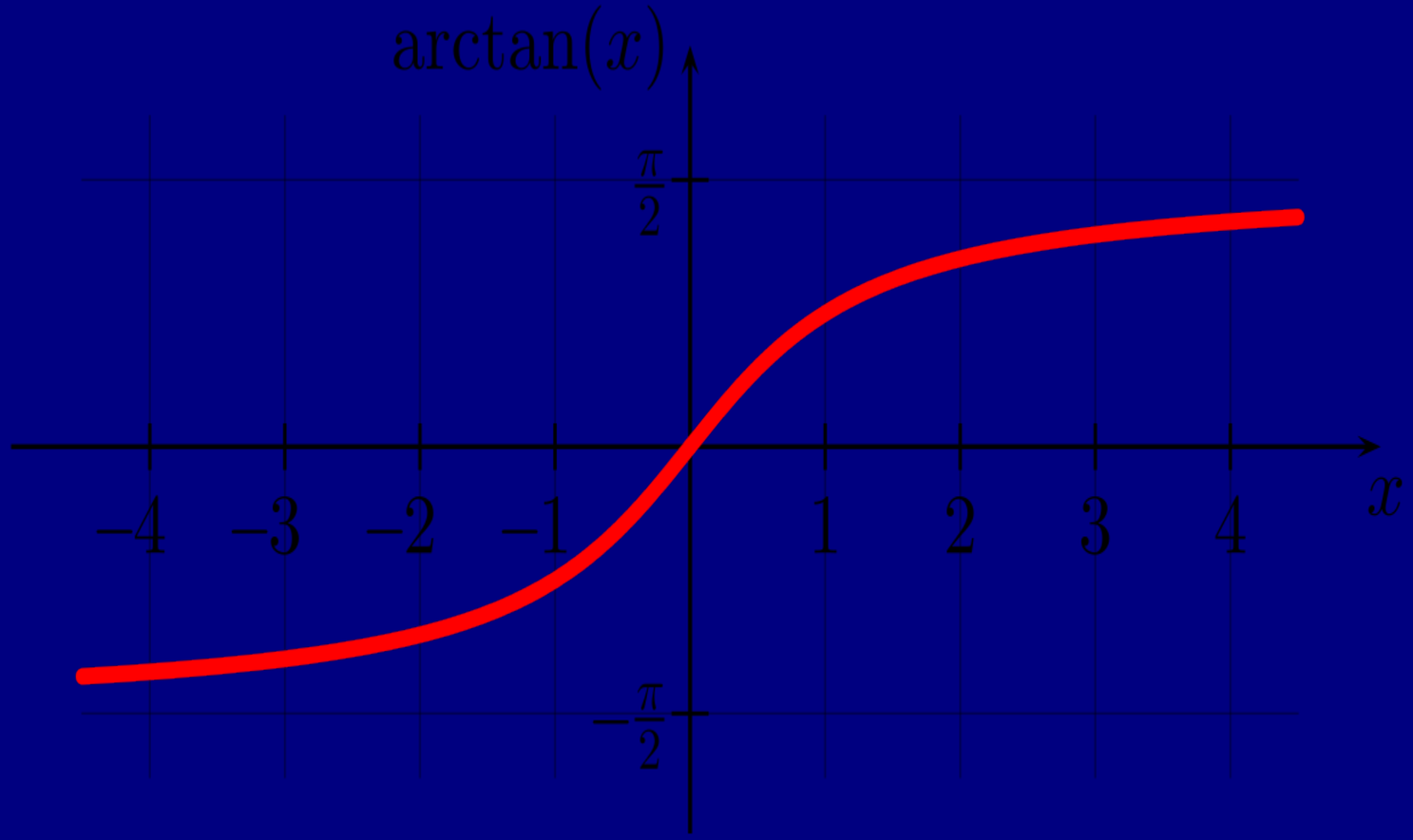
Challenges

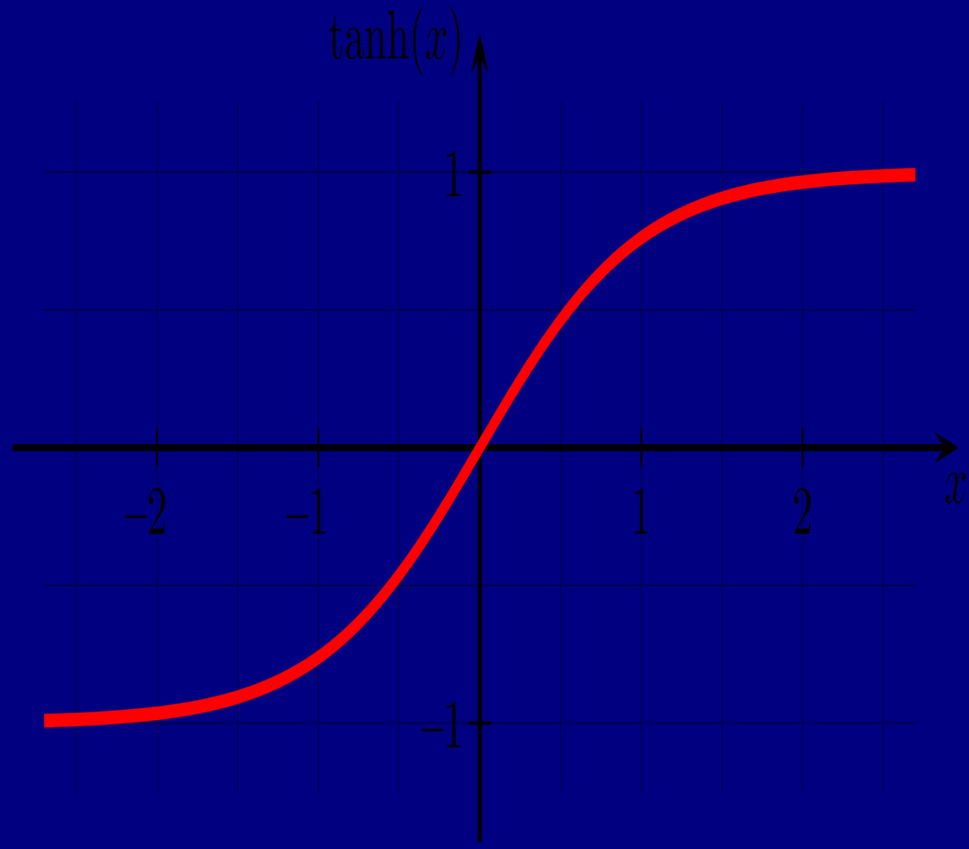
- How do we constrain values to 0..255?
  --> we do not want to worry about that when writing our function



Constrain colors to 0..255

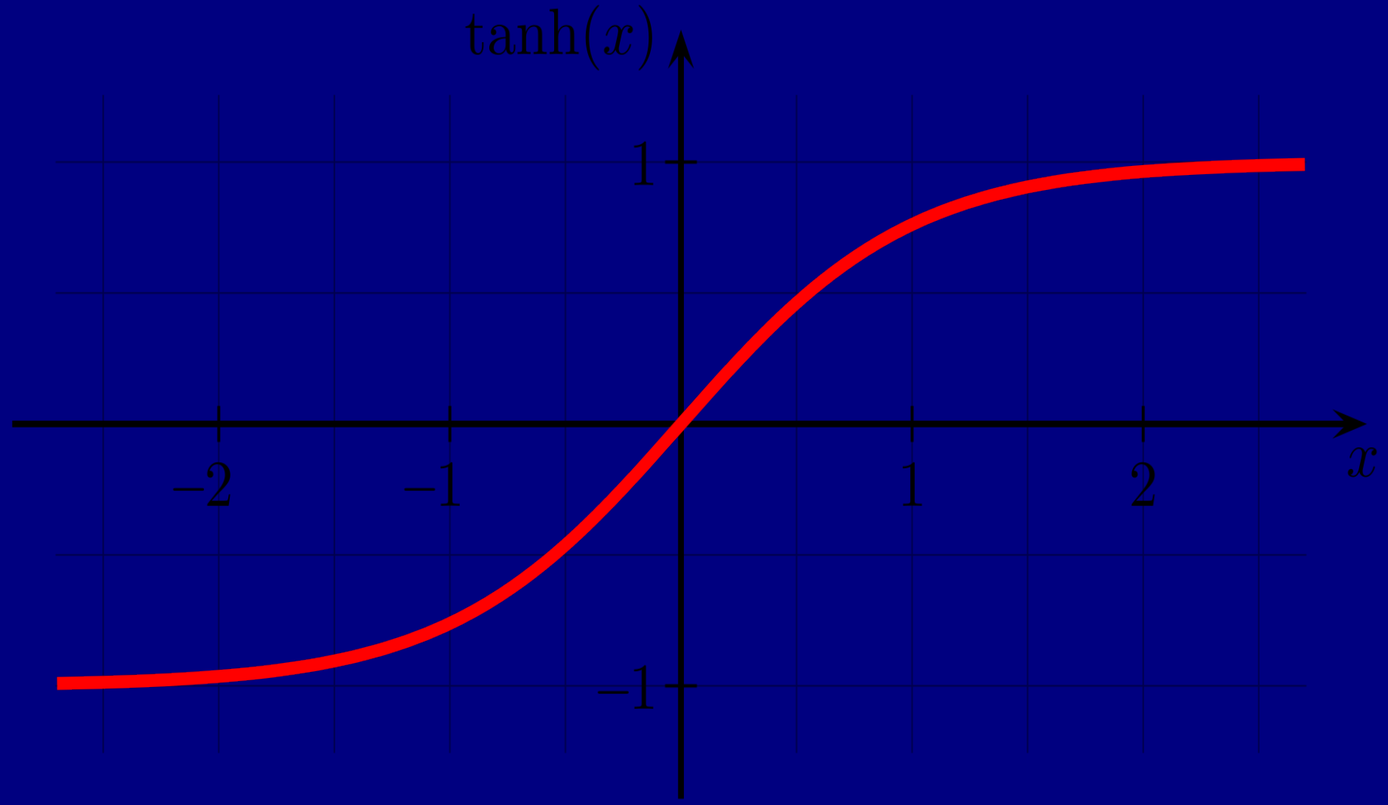
- How to make a picture of a function? --> Try arctan or tanh

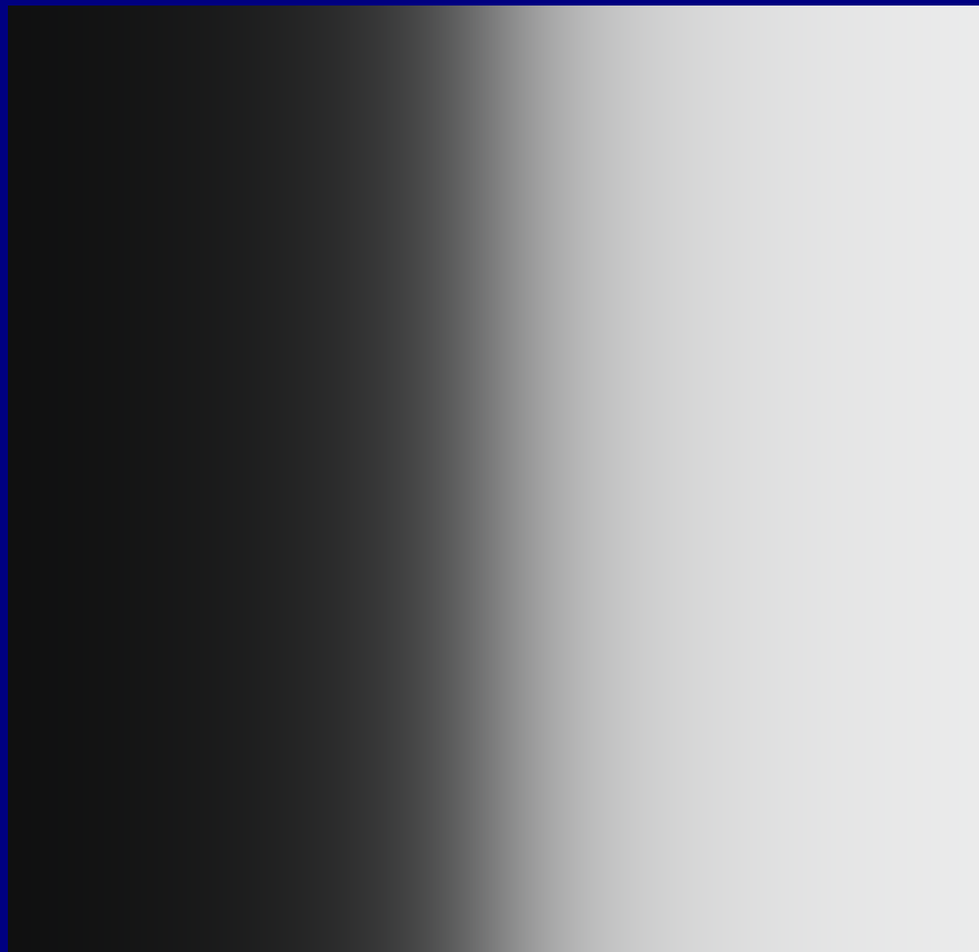




Constrain colors to 0..255

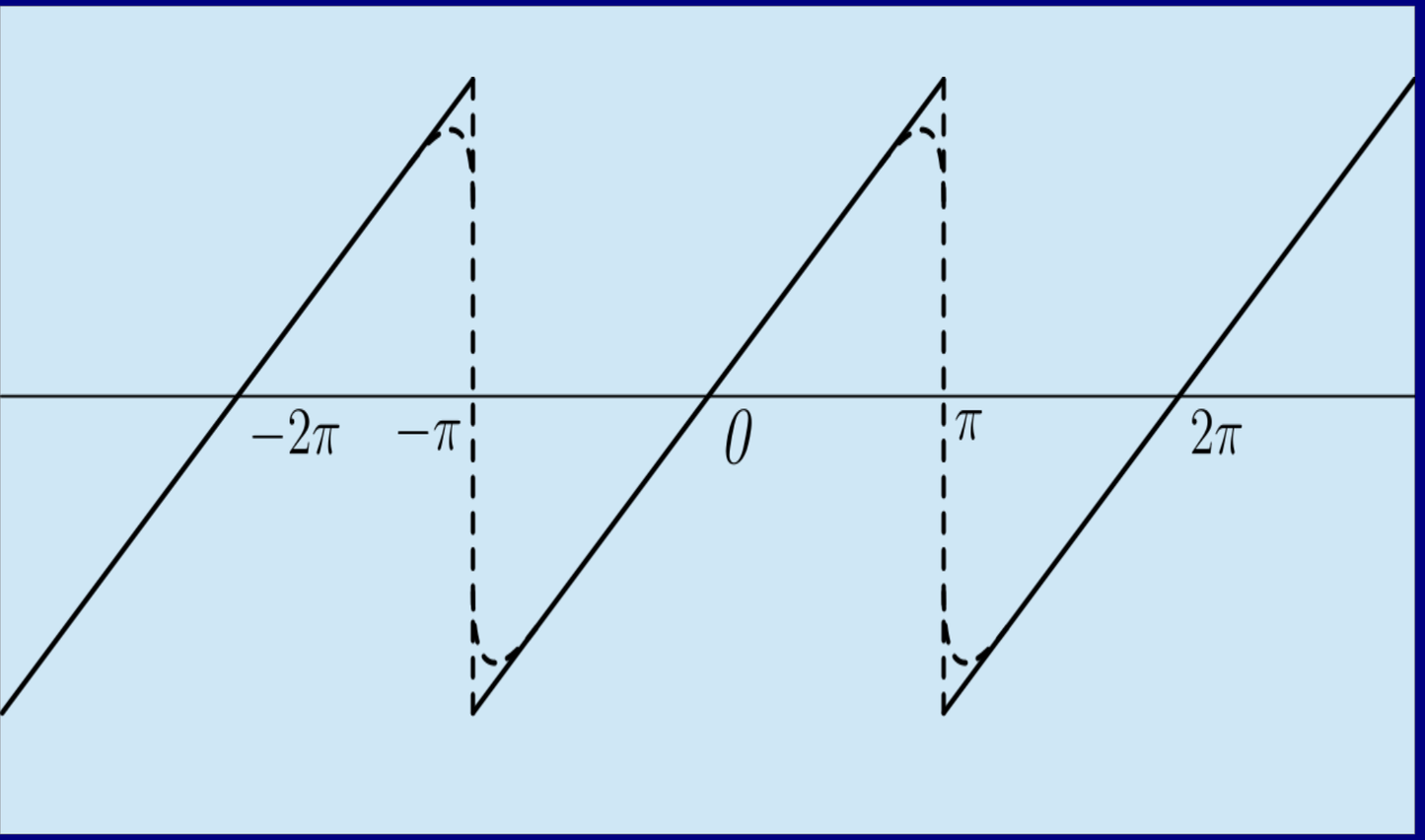
- How to make a picture of a function? --> Try tanh

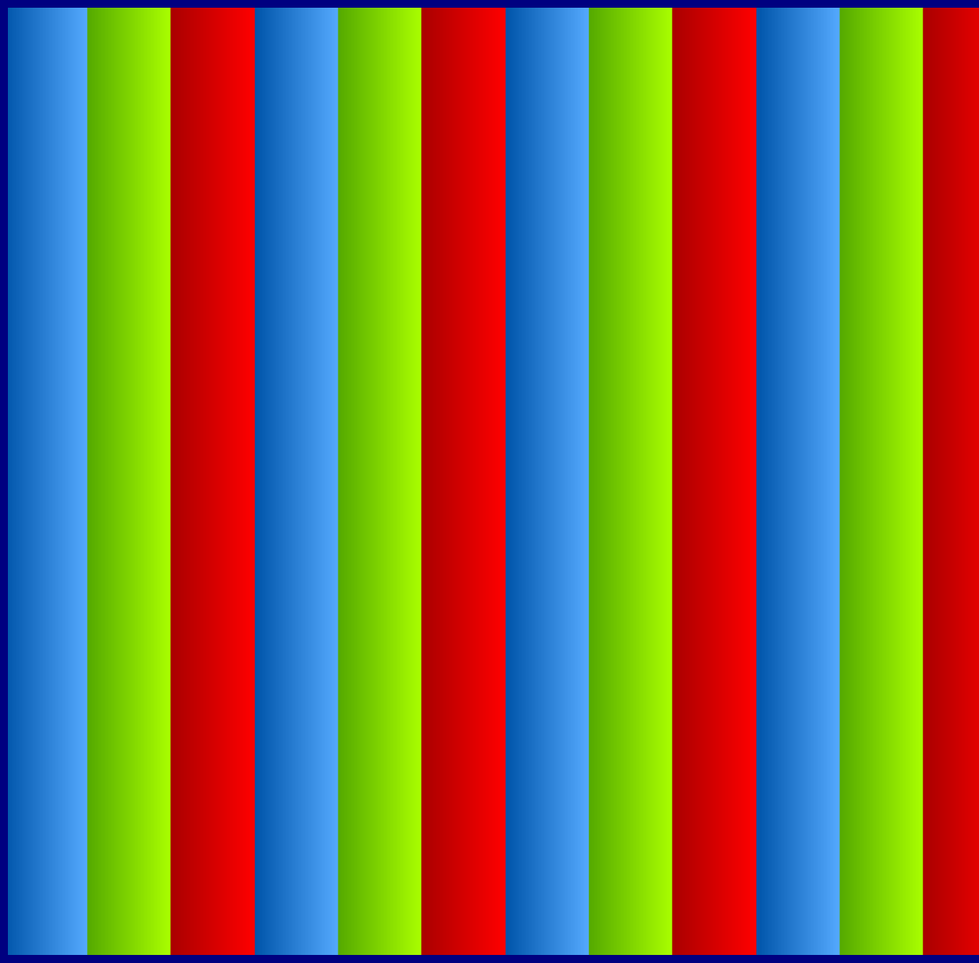




Constrain colors to 0..255

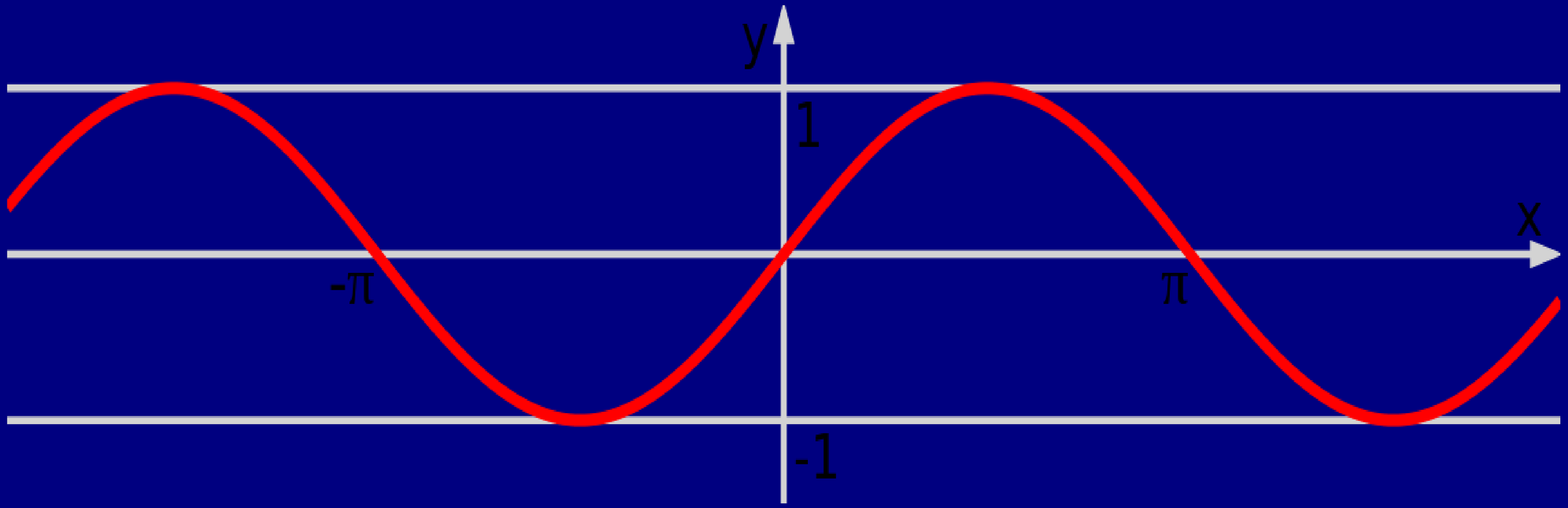
- How to make a picture of a function? --> Try bit-and





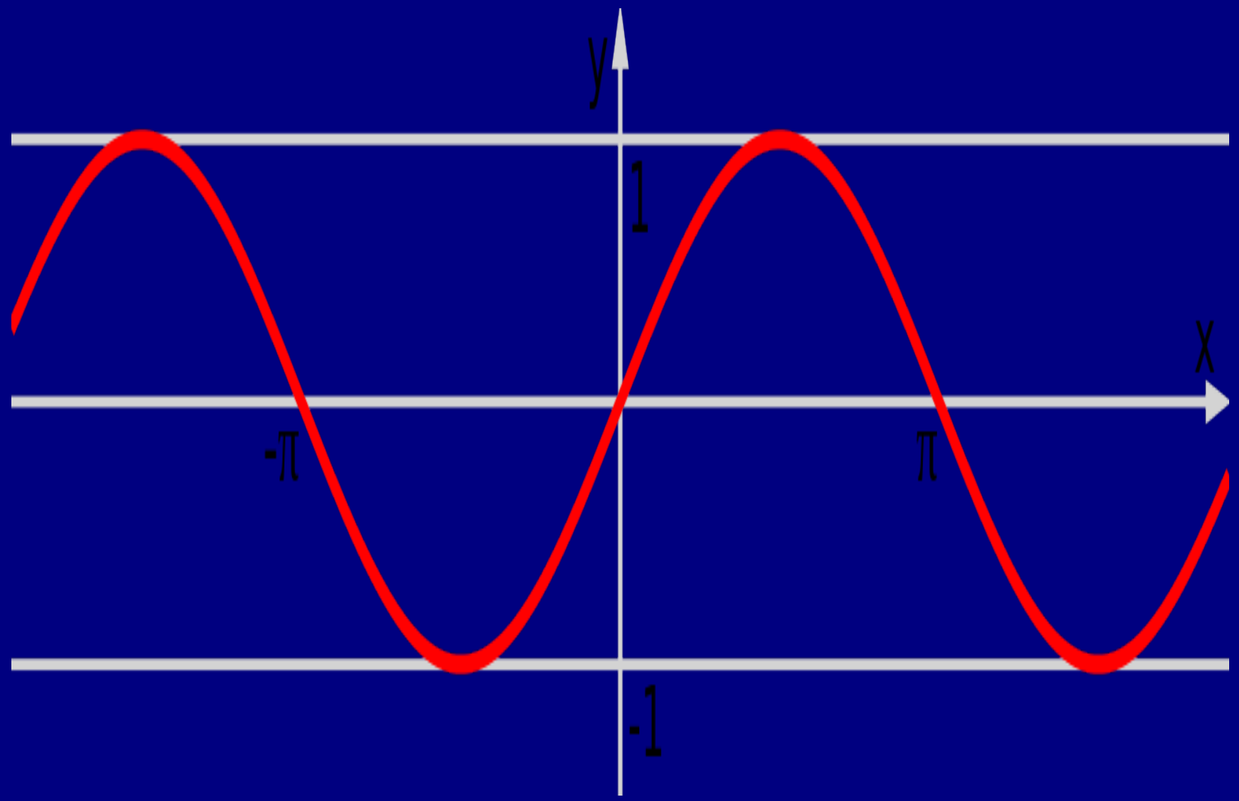
Constrain colors to 0..255

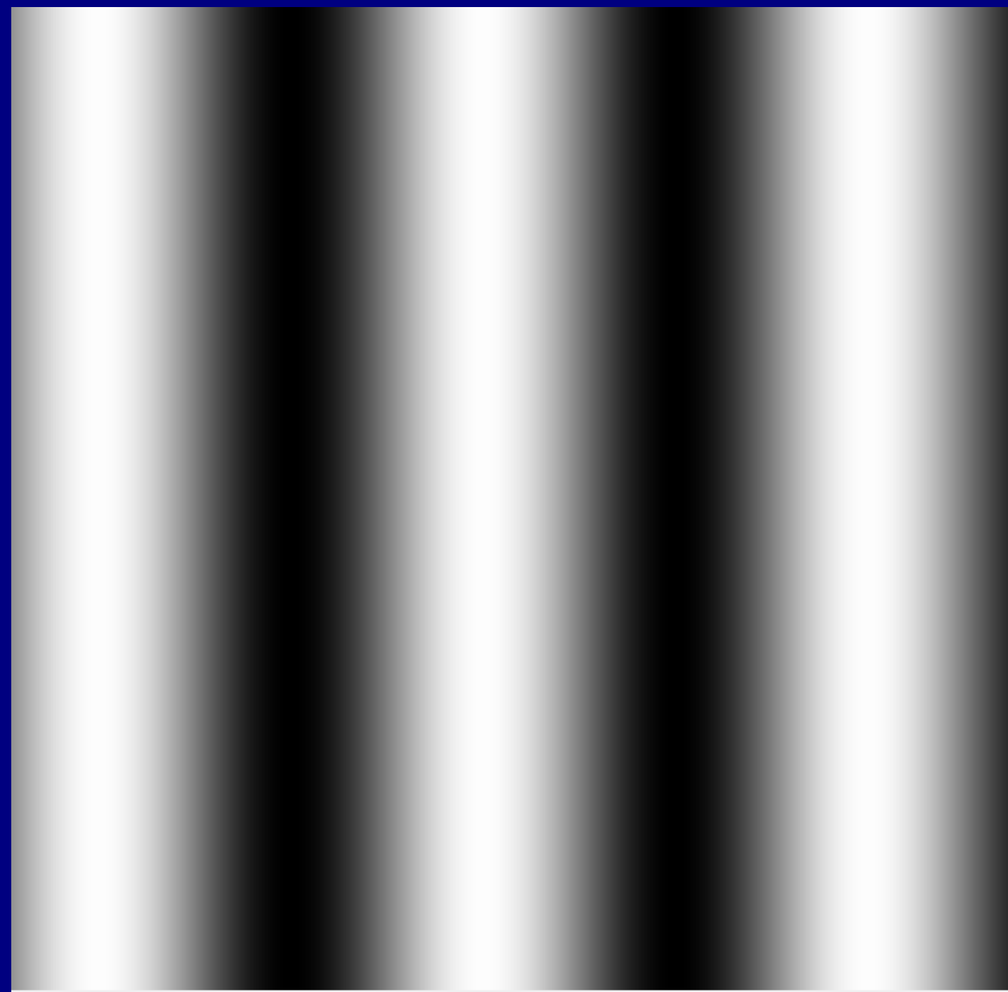
- How to make a picture of a function? --> Try sin



Constrain colors to 0..255

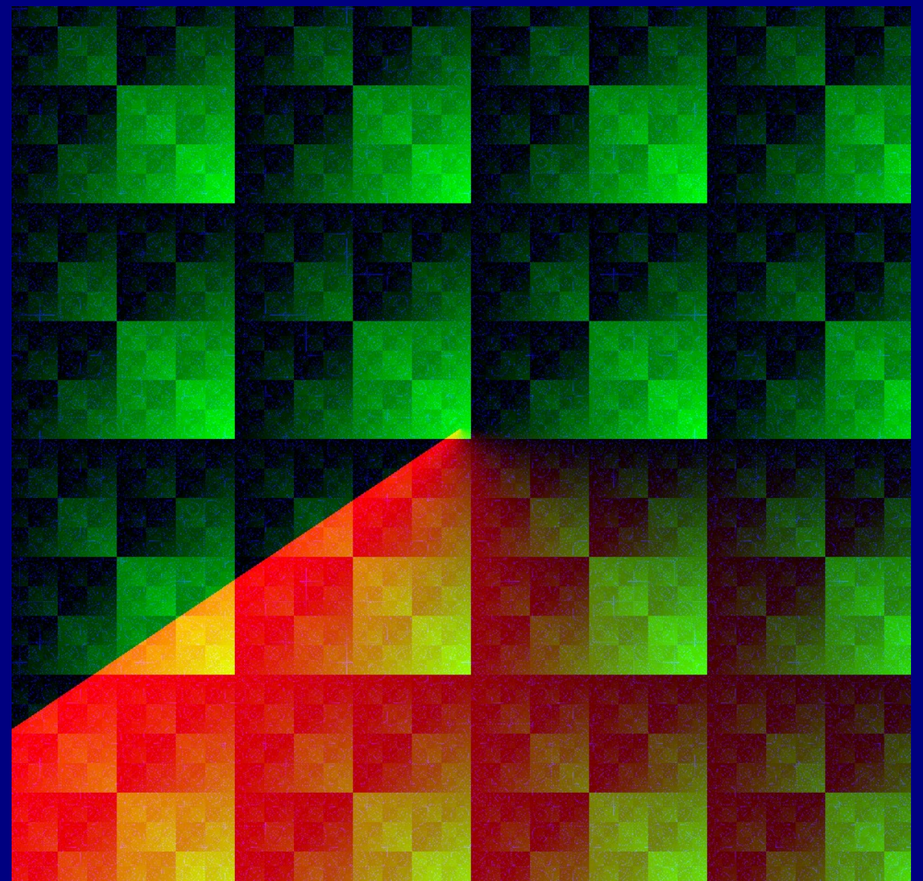
- How to make a picture of a function? --> Try sin





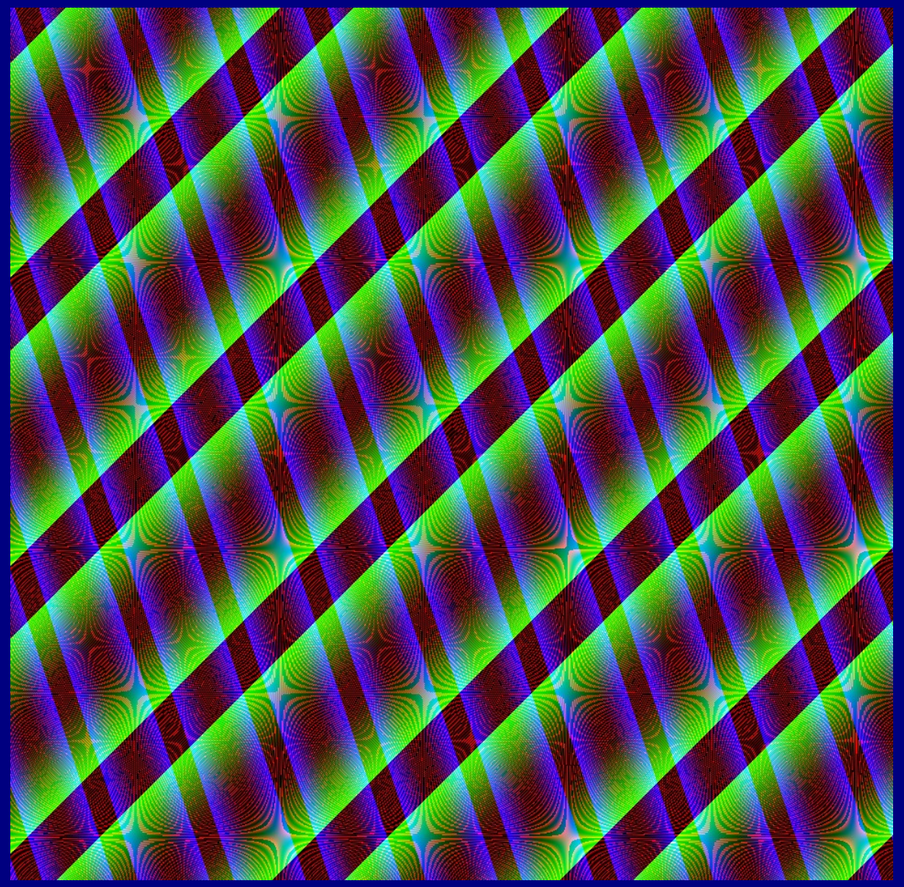
Challenges

- We want f_r, f_g and f_b to be different to have colors work, but not too independent -- three totally different functions



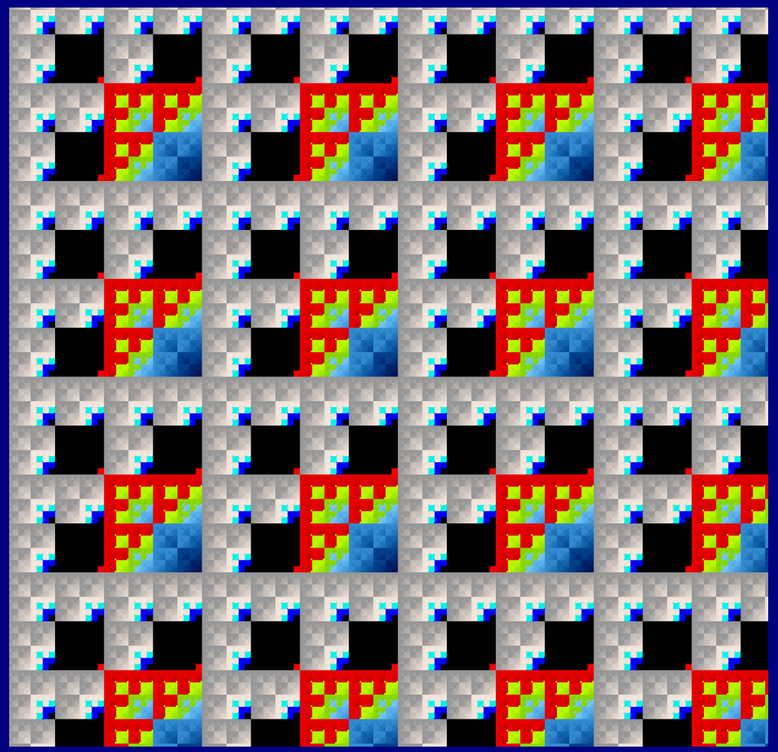
Challenges

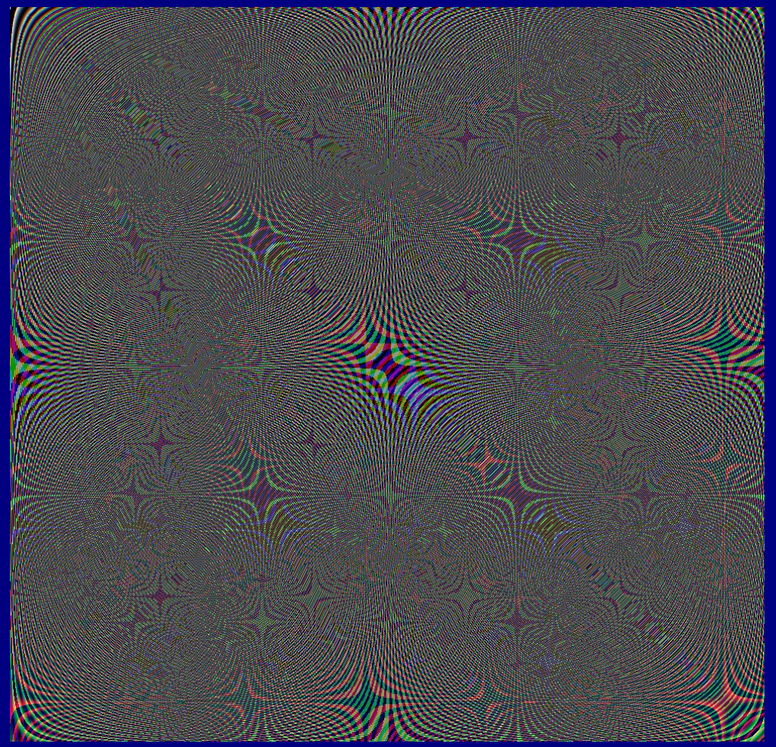
- We want f_r, f_g and f_b to be different to have colors work, but not too independent – sin of three different functions

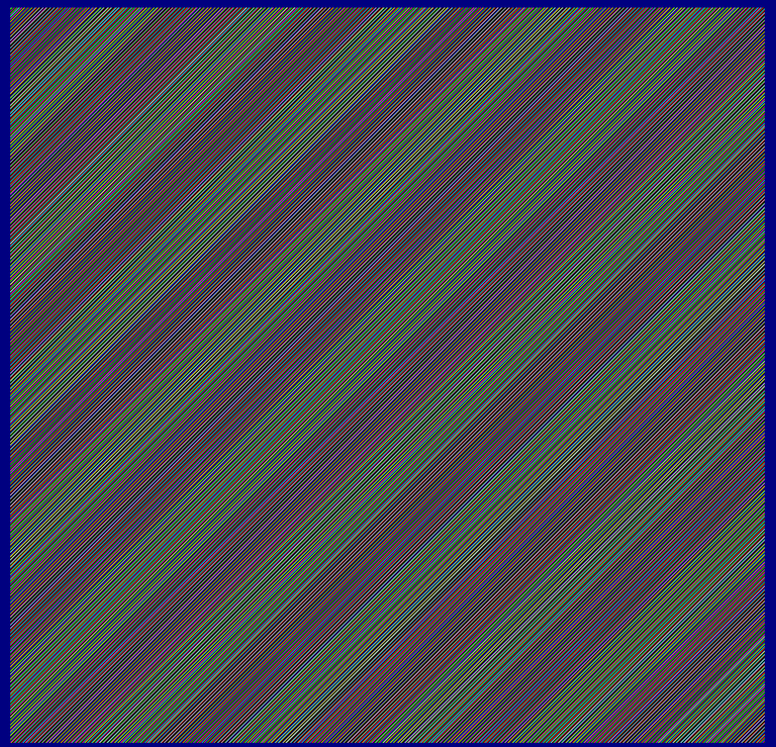


Challenges

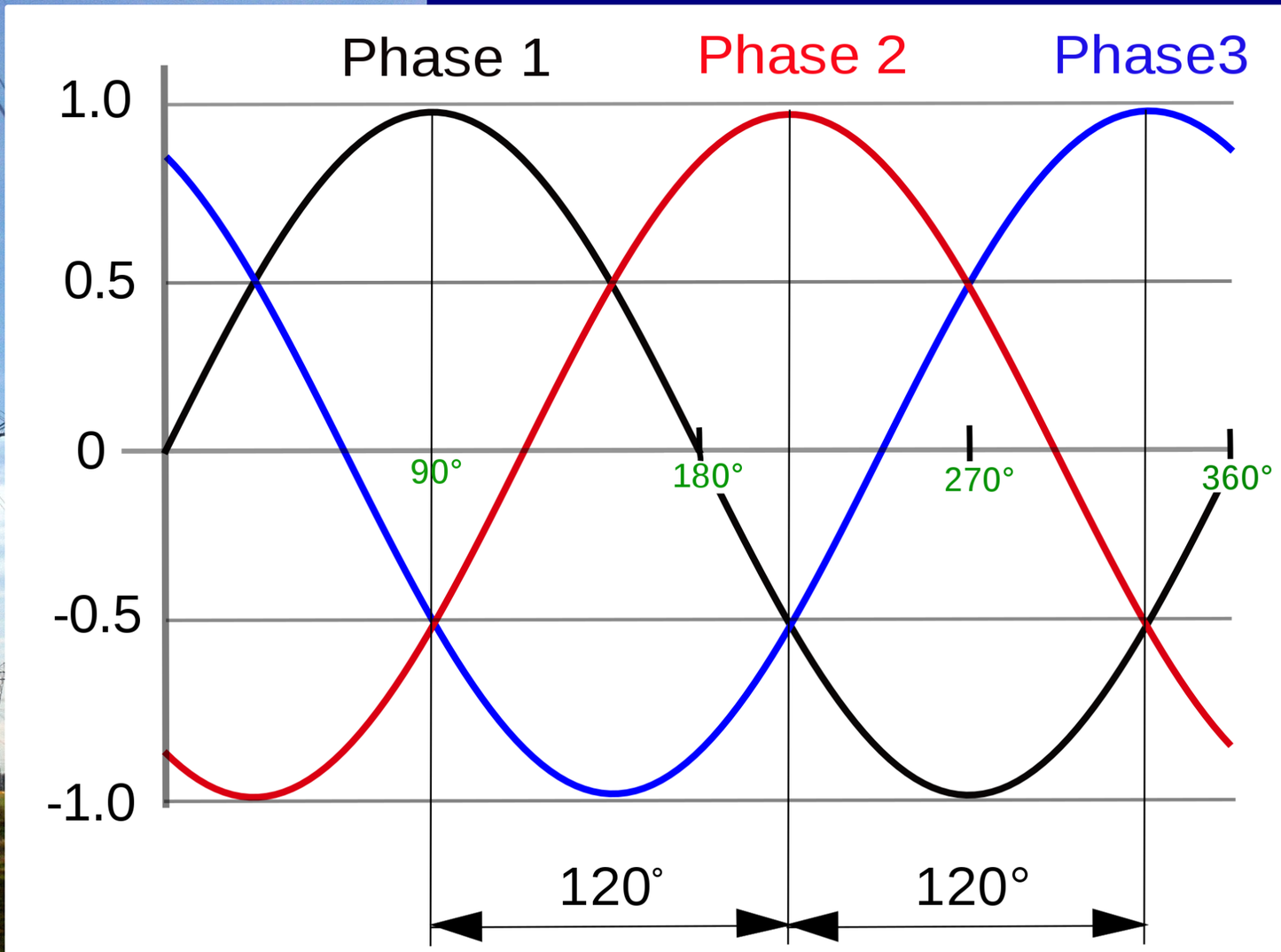
- We want f_r, f_g and f_b to be different to have colors work, but not too independent – sin of three similar functions



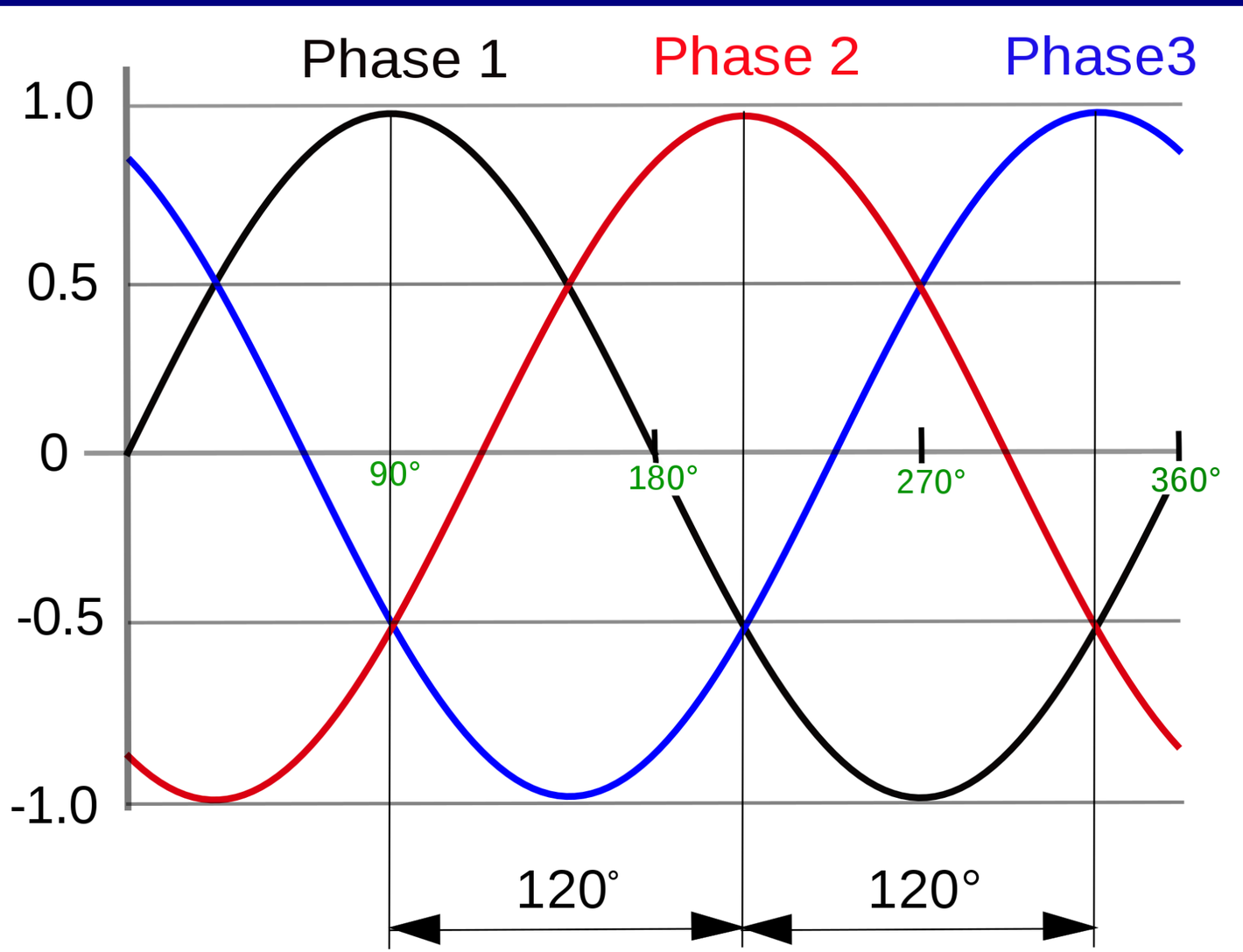




RGB like 3 Phase Electricity

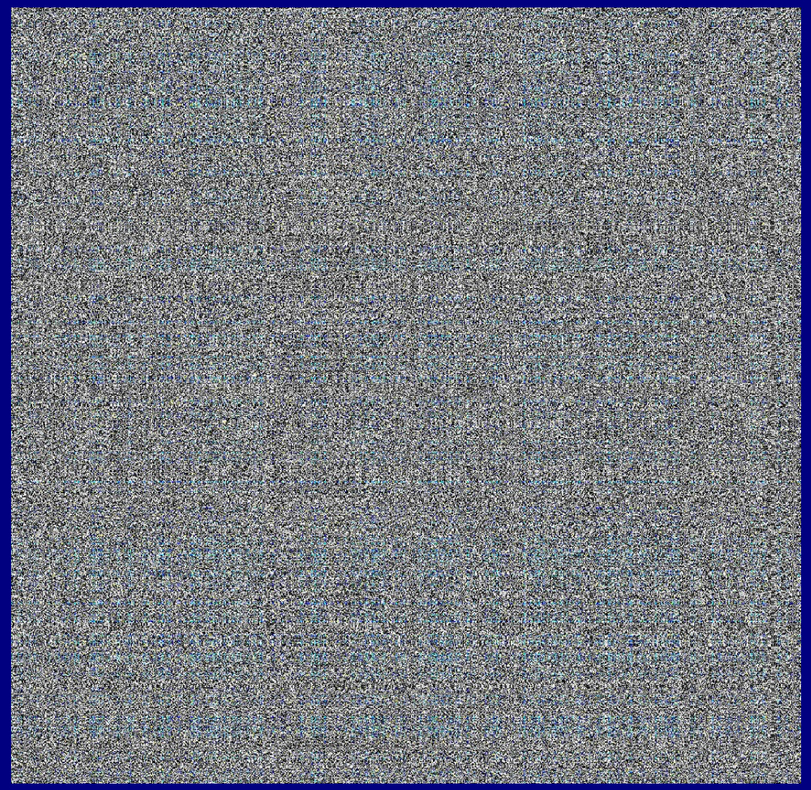


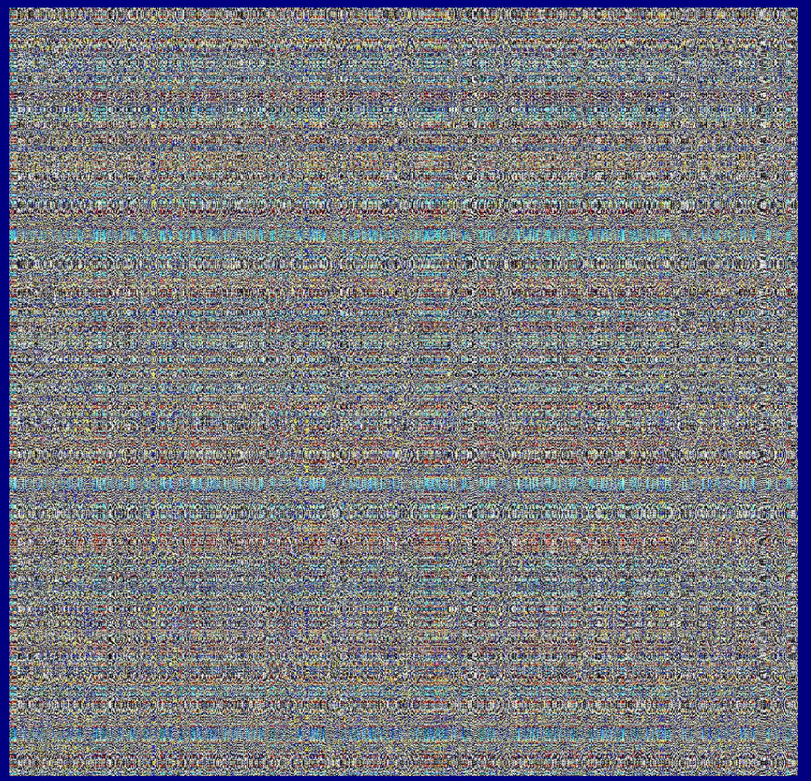
RGB like 3 Phase Electricity



Challenges

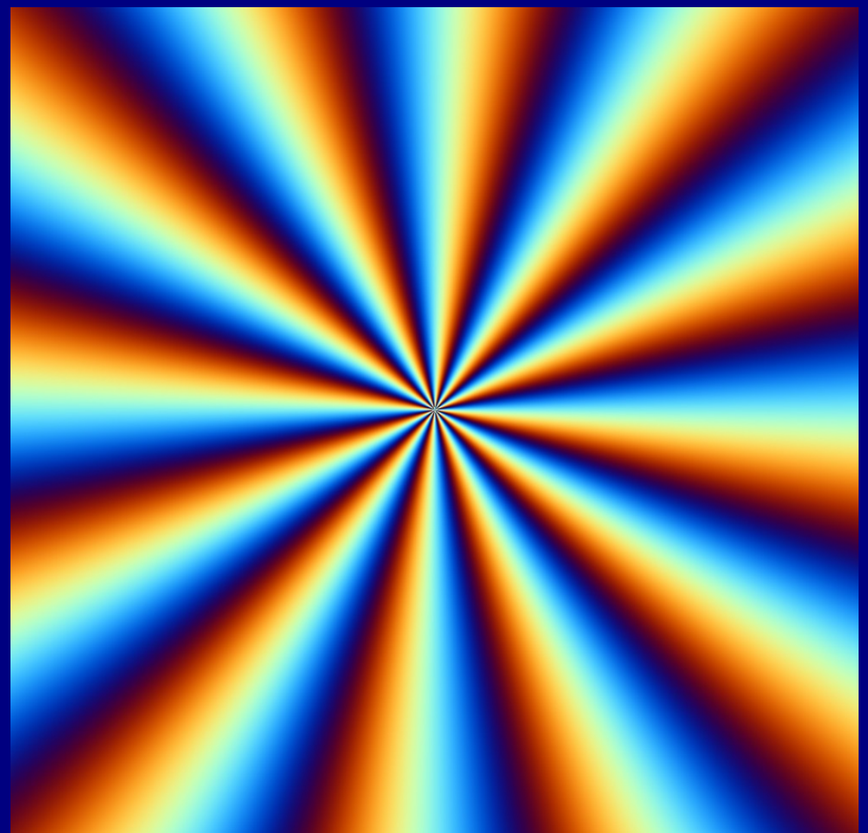
- Find the right “speed” of the functions
- Here we vary the color based on function values





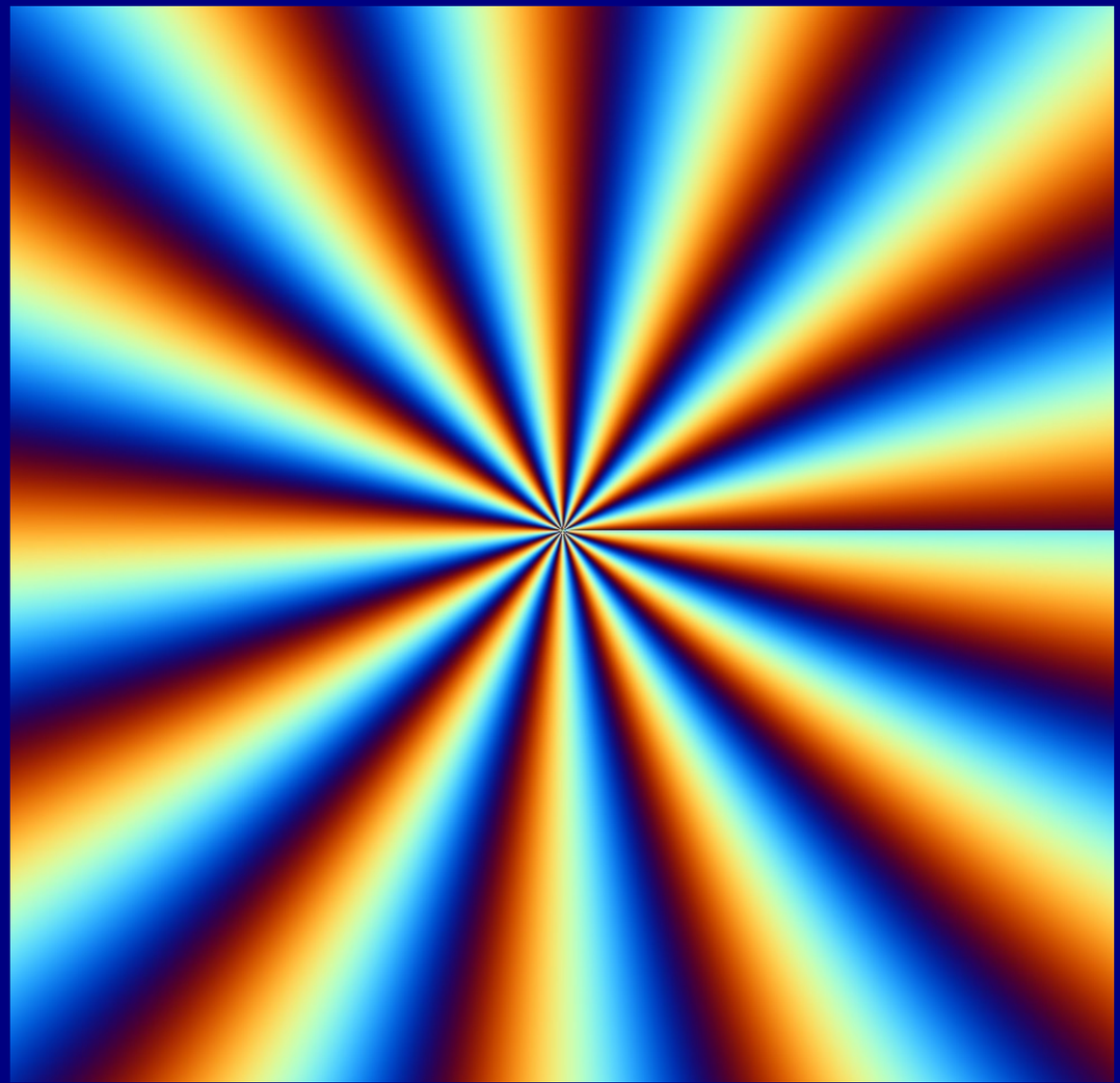
What works

- atan2 -> angle -> vary color with angle
- Put value in sin with different phases for different colors



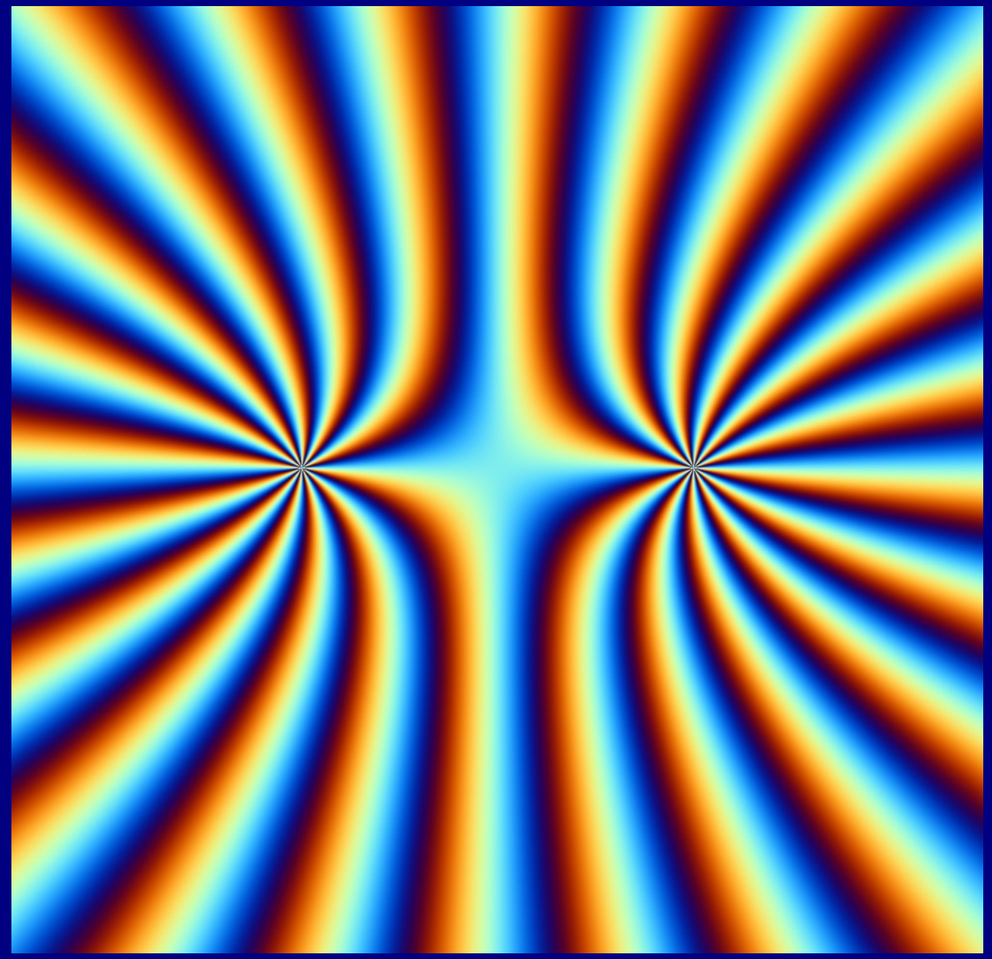
Challenge

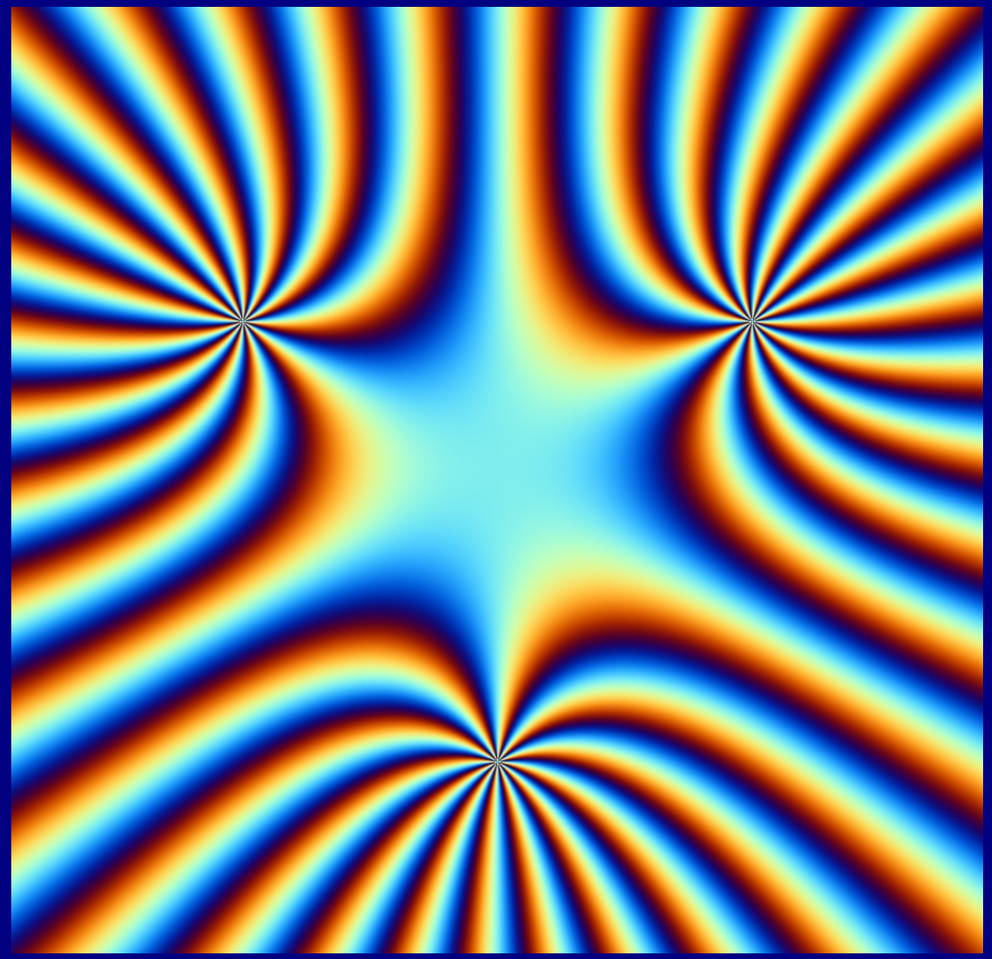
- Keep it smooth



What works

- Combine two or three centers with +



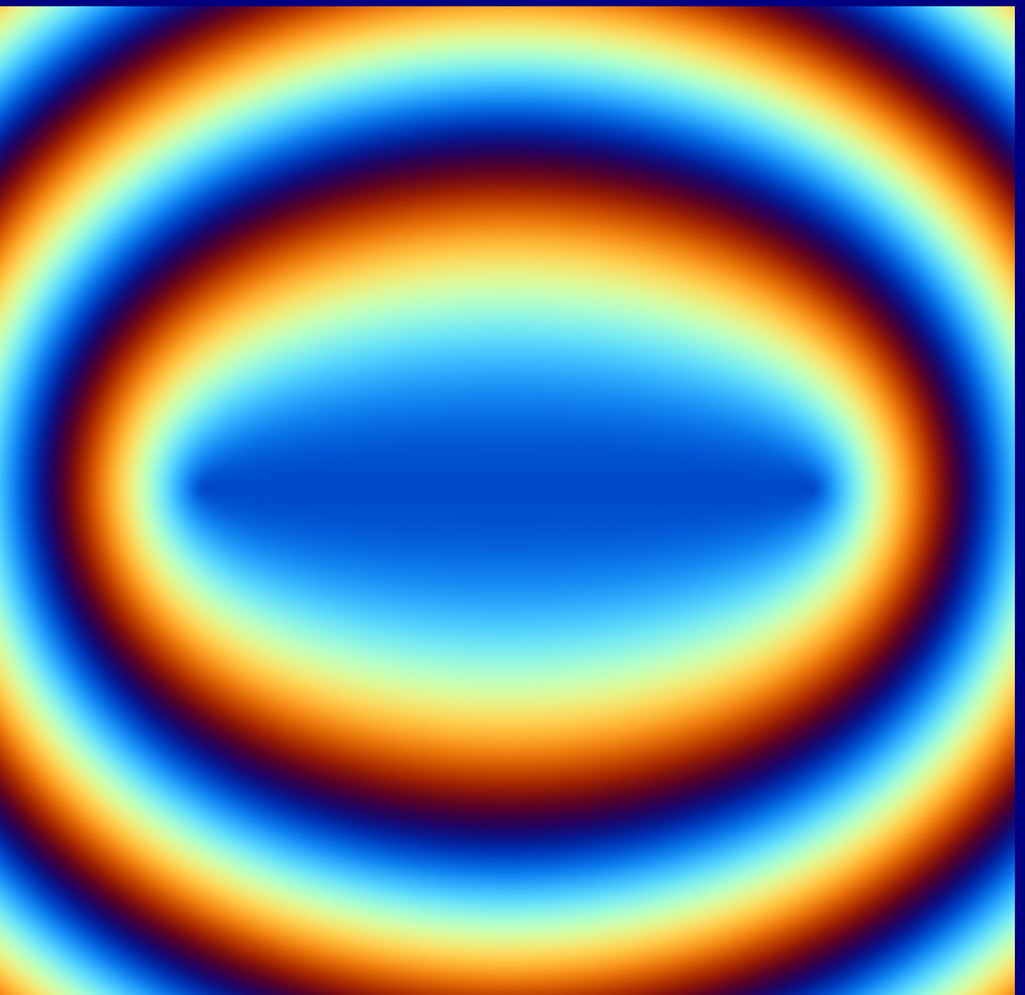


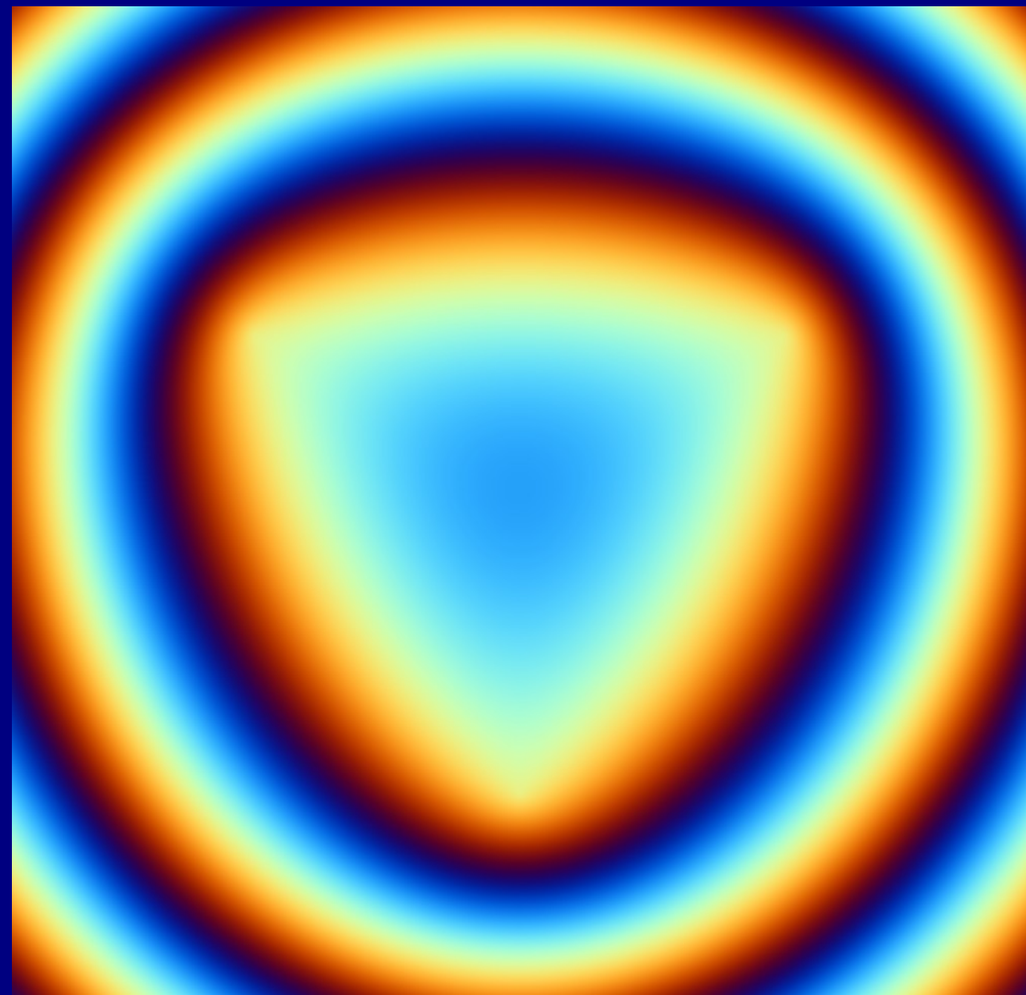
What works

- Use distance from point instead of angle

What works

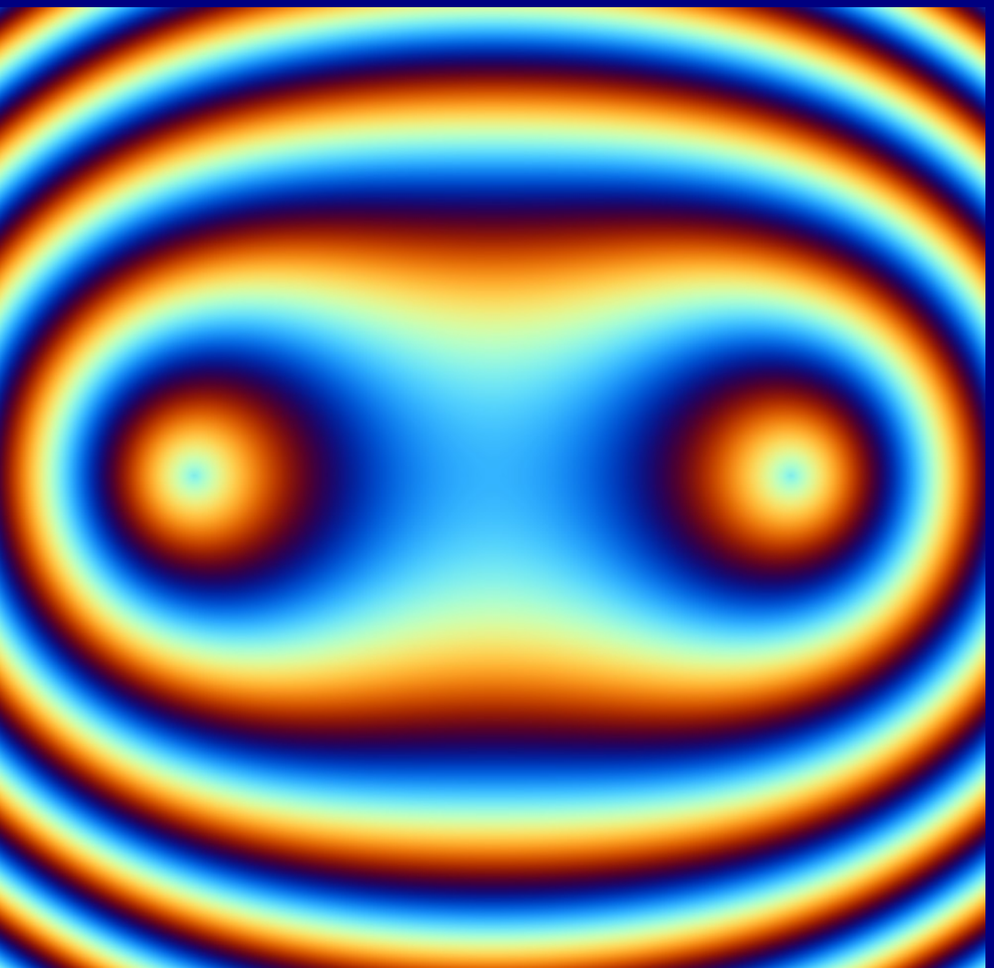
- Add distance from multiple points

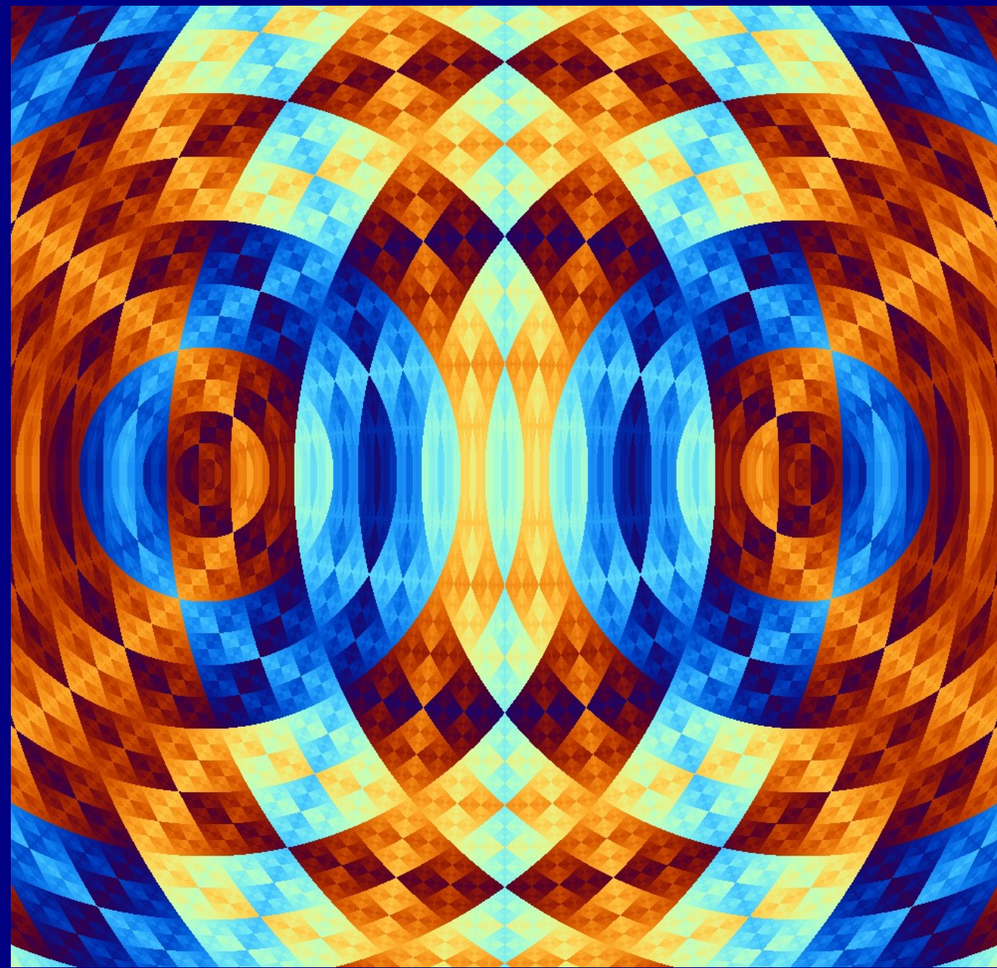




What works

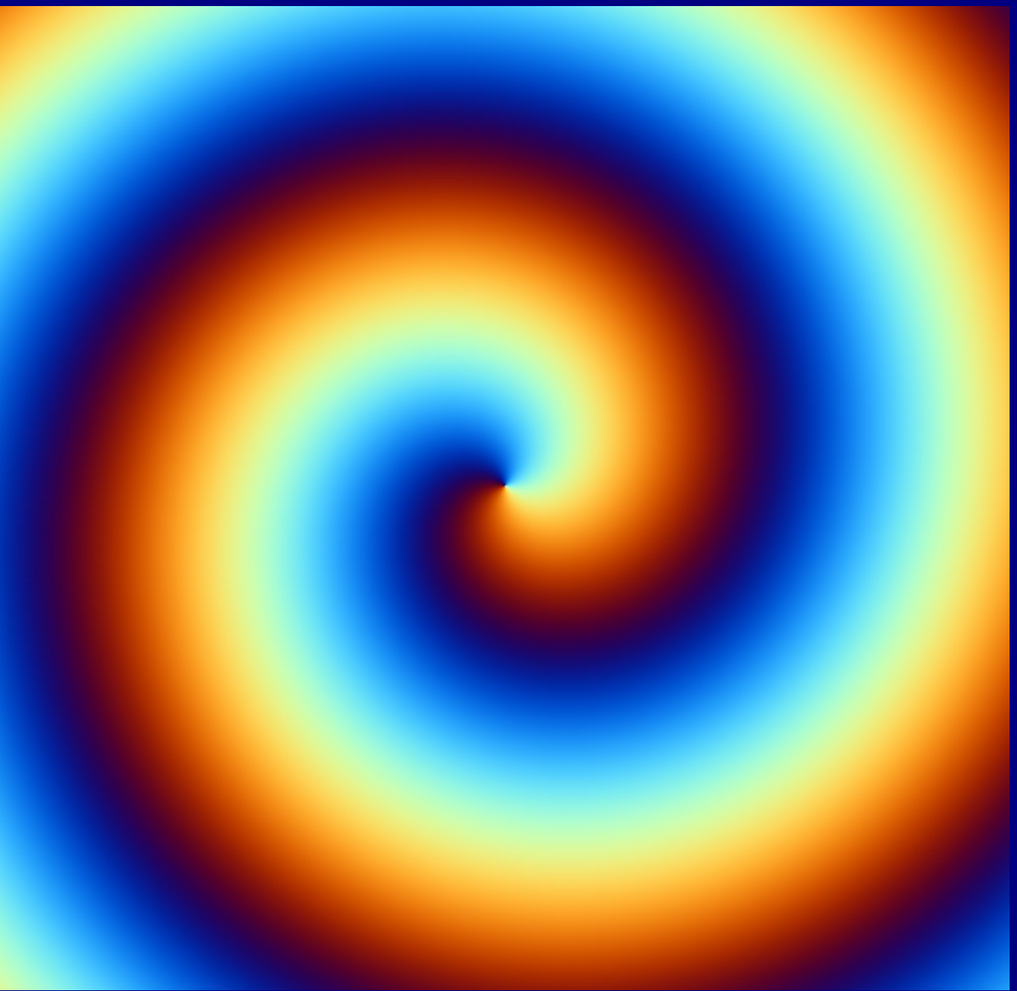
- Multiply or xor distance from multiple points

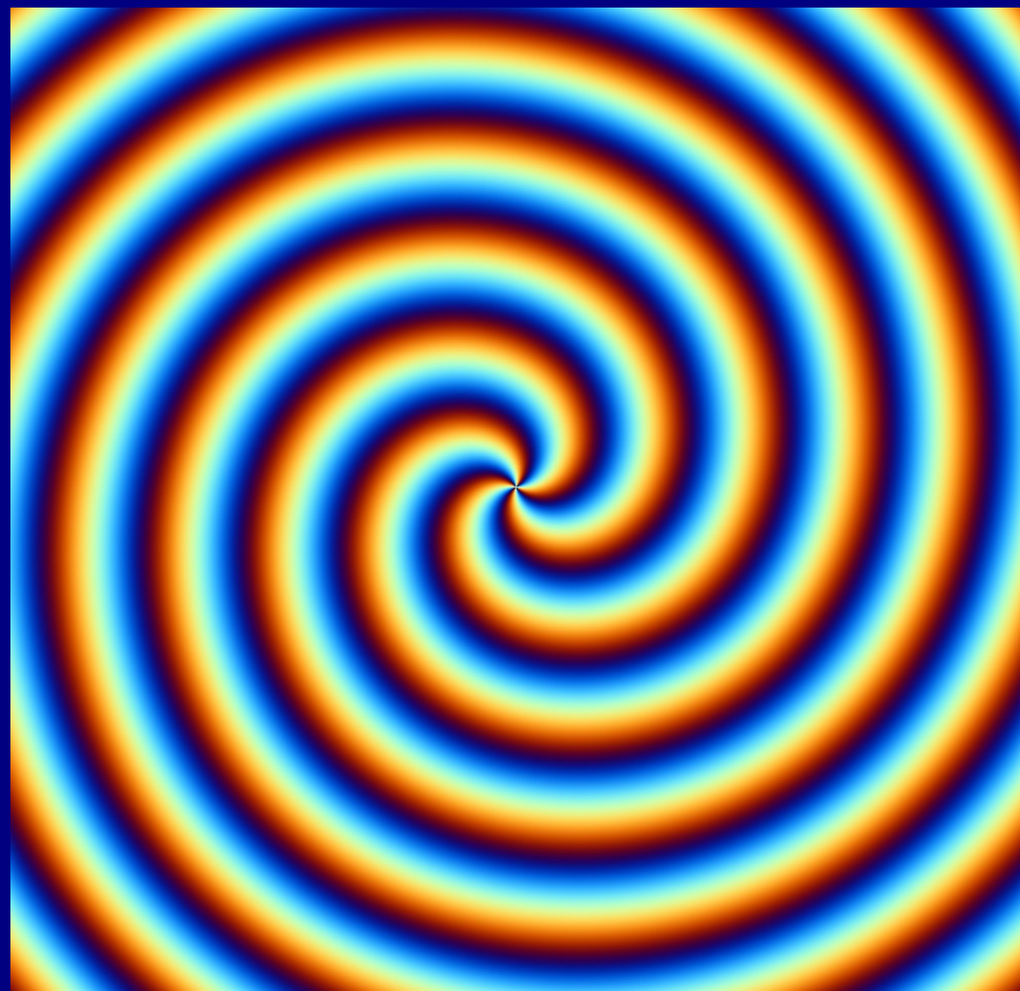




What works

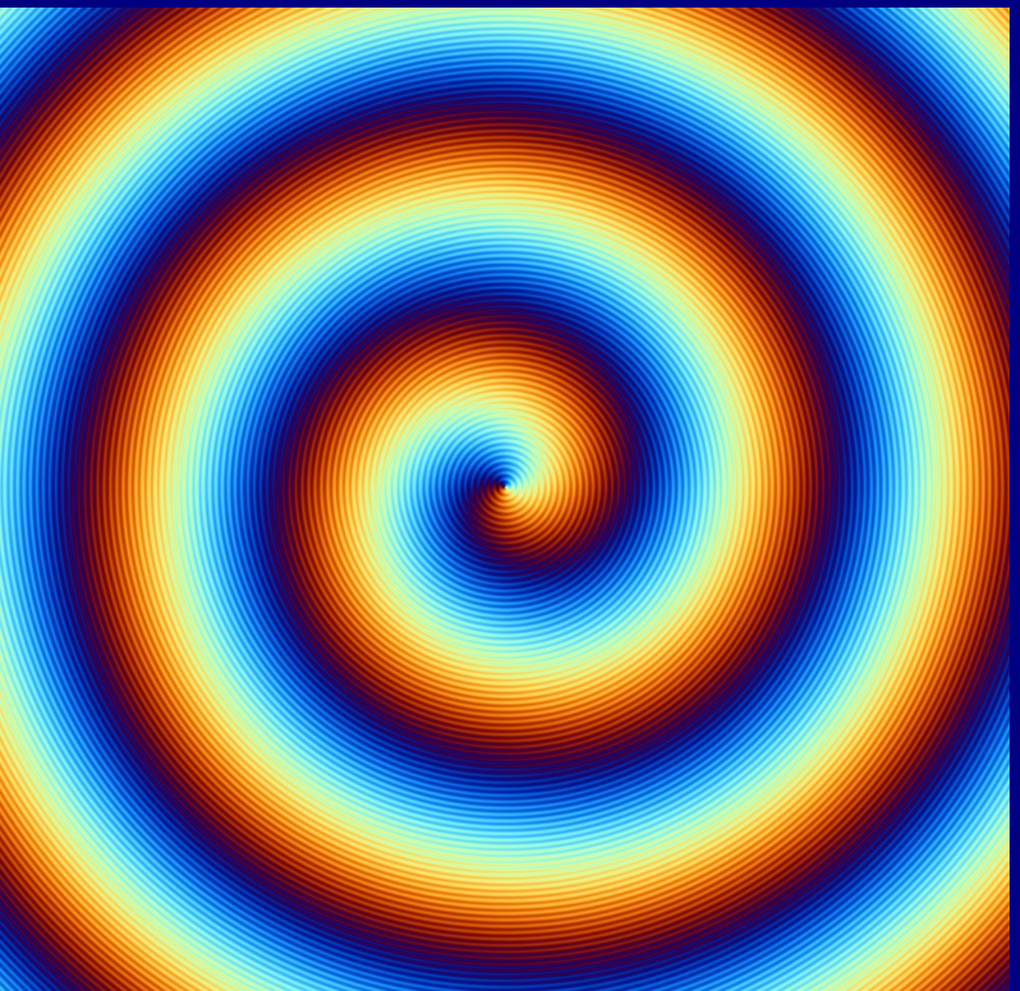
- One center, combine radius and angle

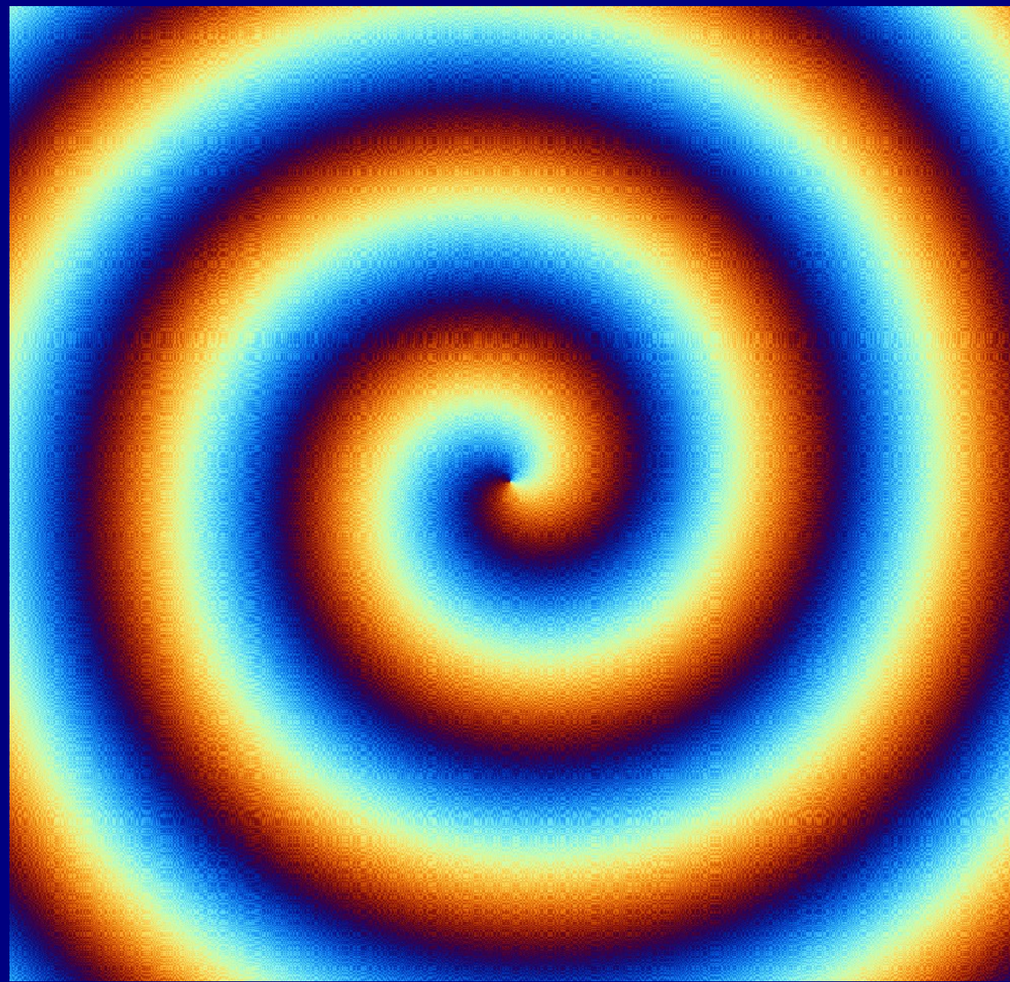




What works

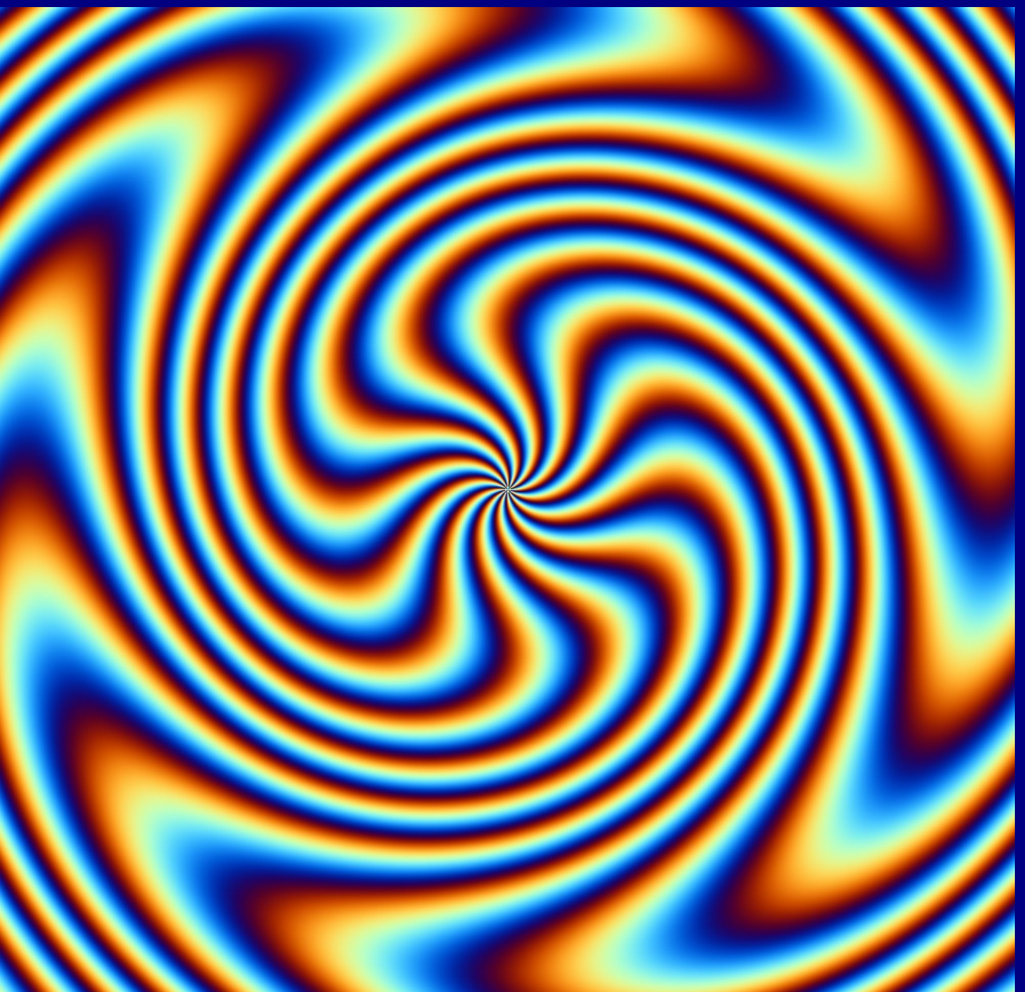
- Combine with an oscillation or roughness

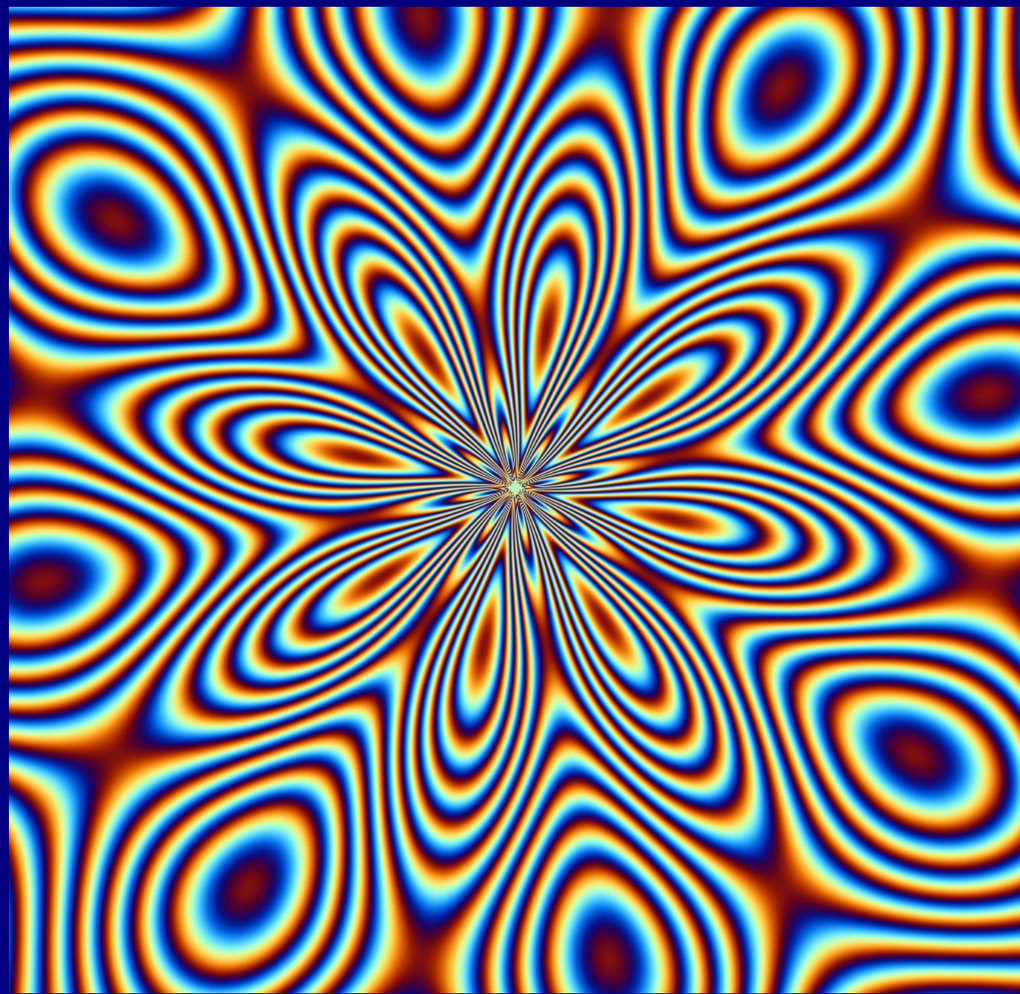




What works

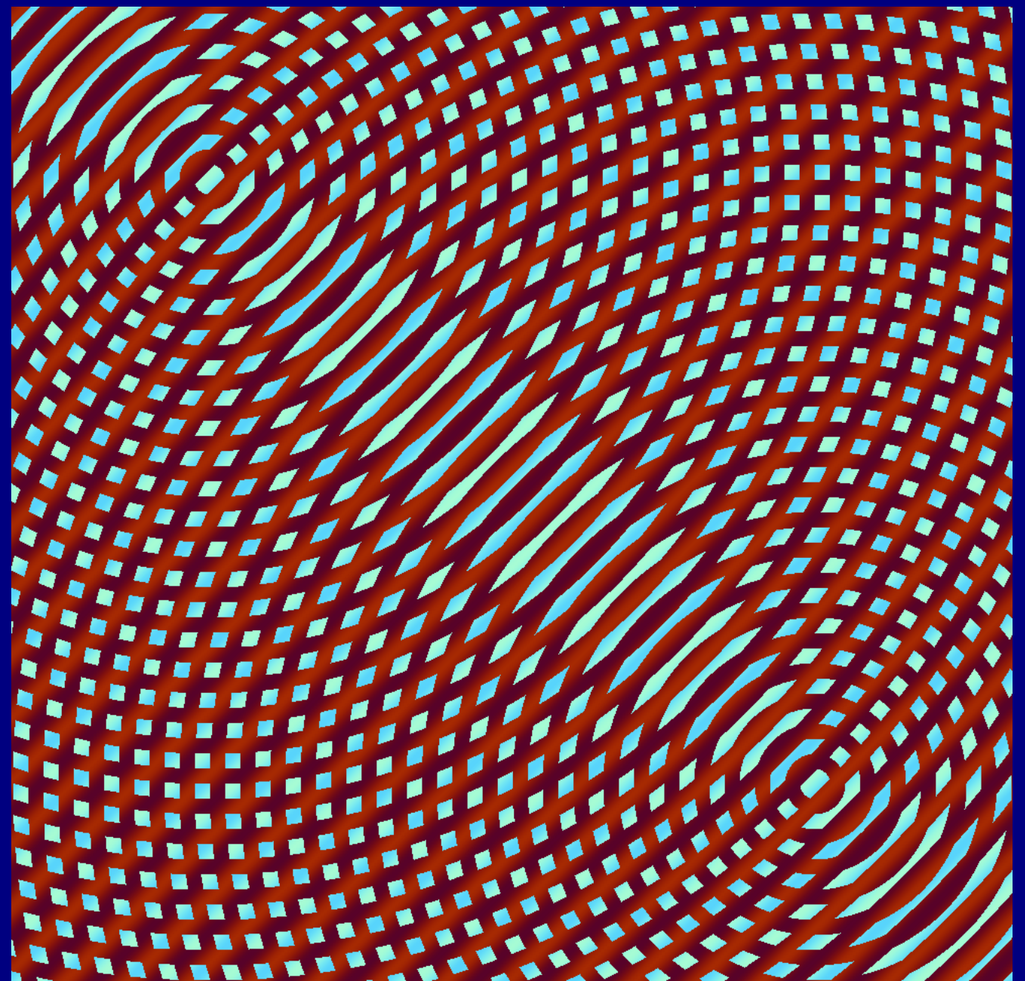
- Apply functions (here sin) to radius and/or angle





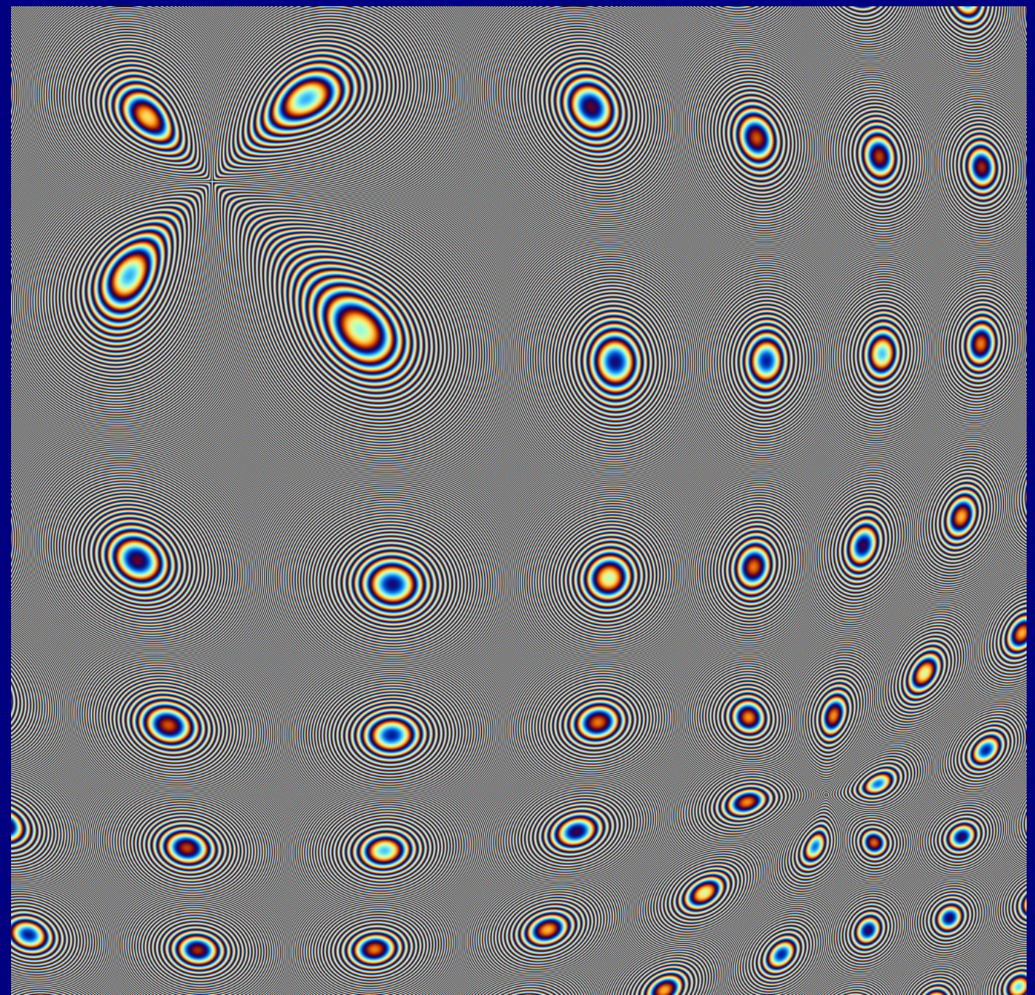
What works

- Create grids using gcd of distances divided by something



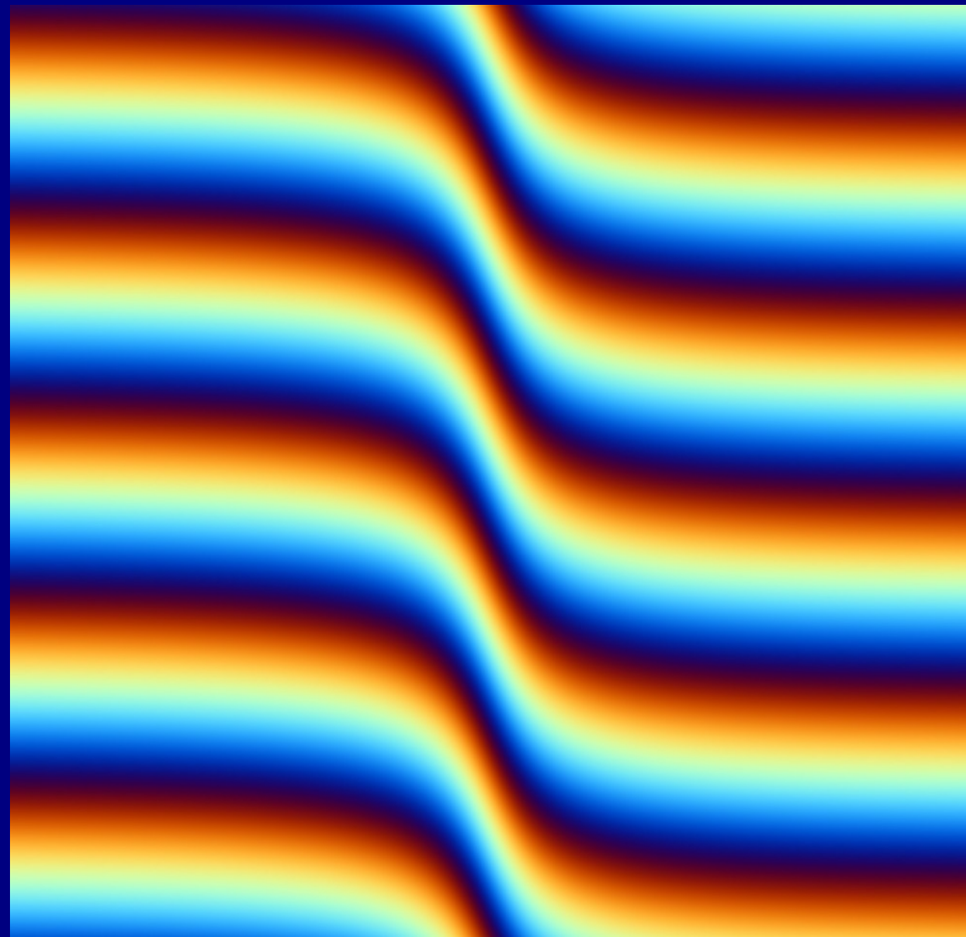
What works

- Polynomials (here degree 3) can give these bubbles



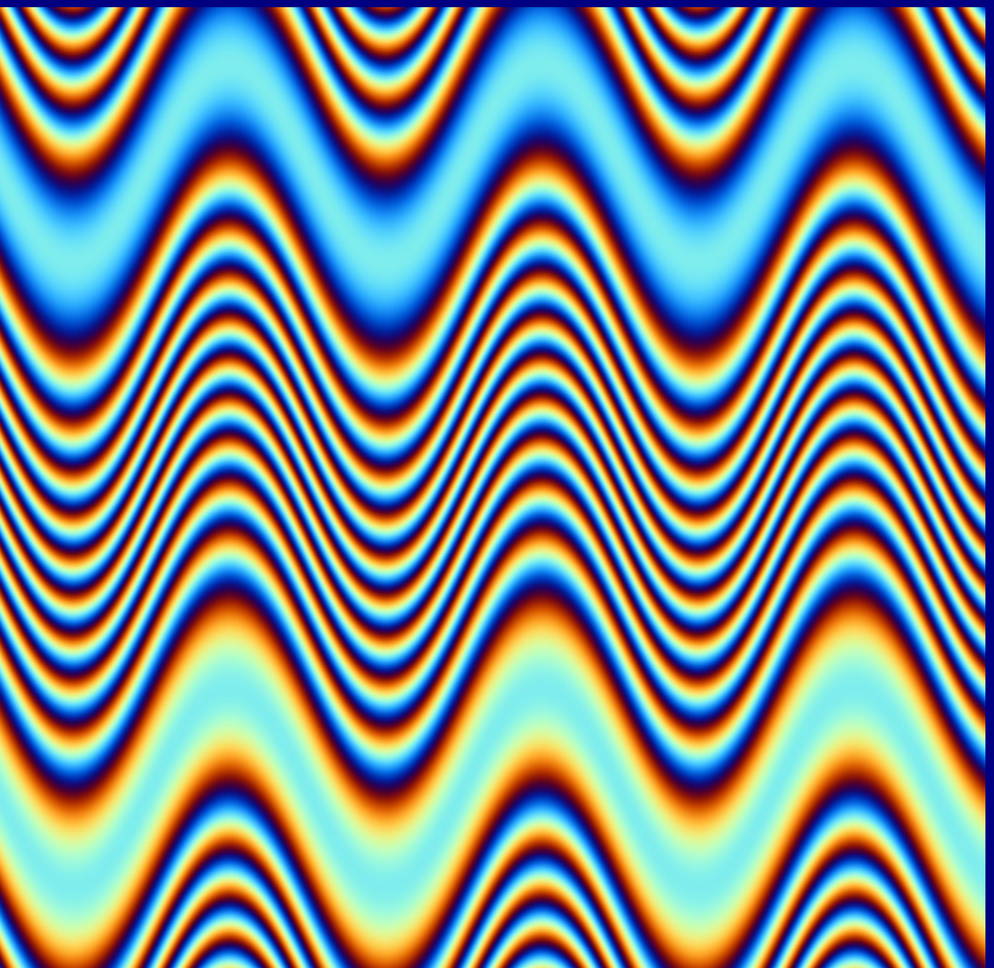
Plot functions

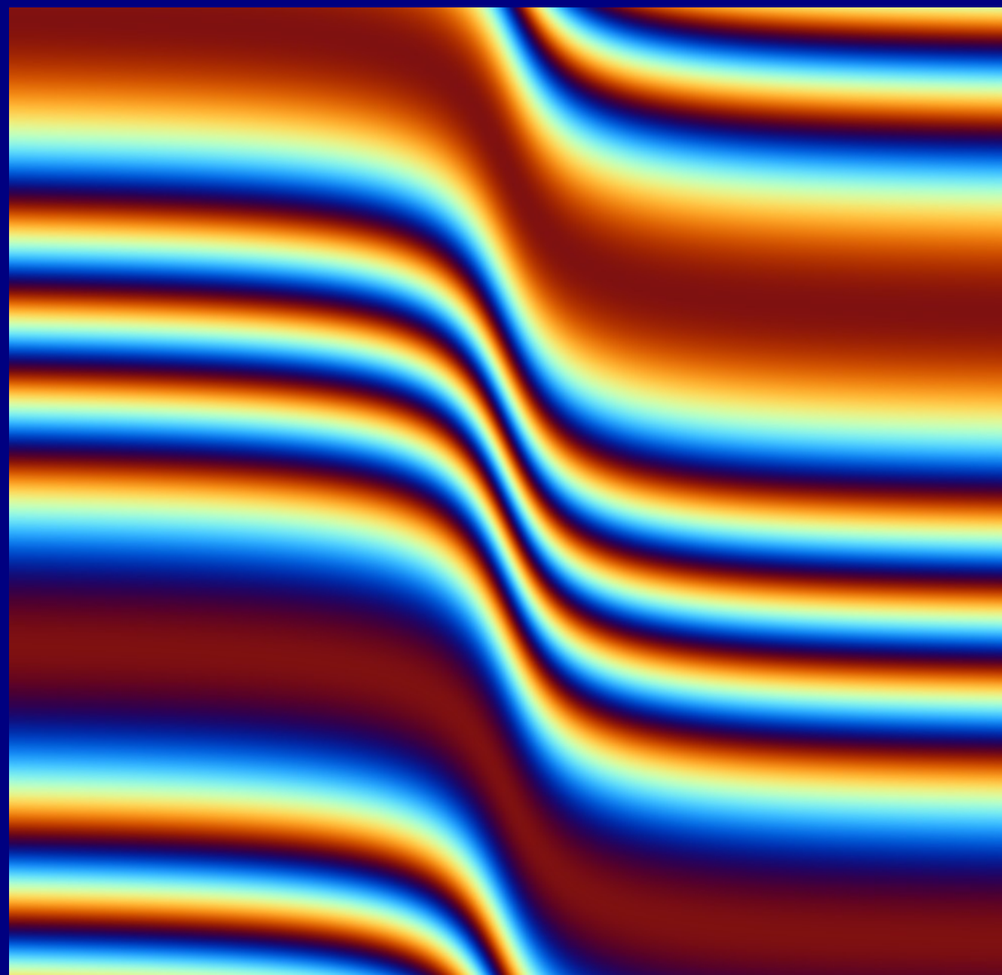
- Something like f(x) – y: sin(x) and atan(x)



Plot functions

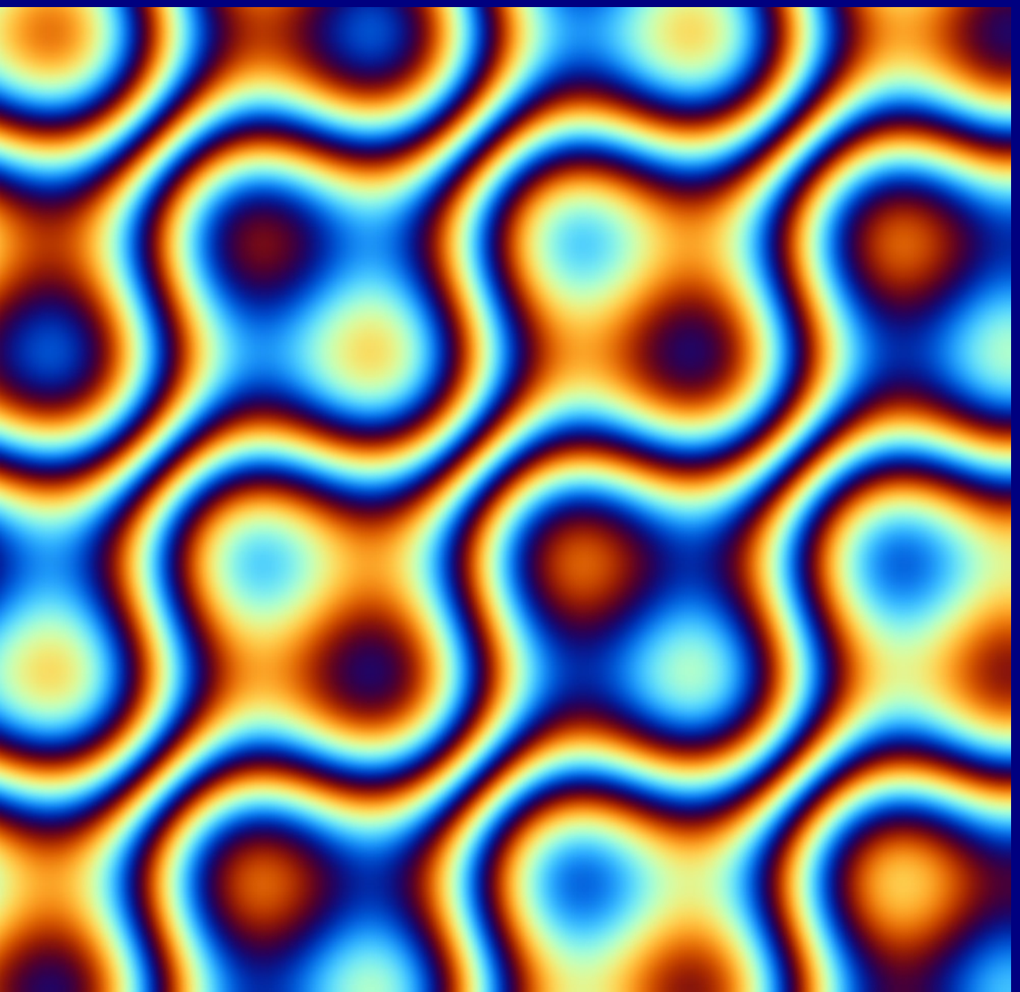
- More dynamic sin(f(x)-y) with right scalings





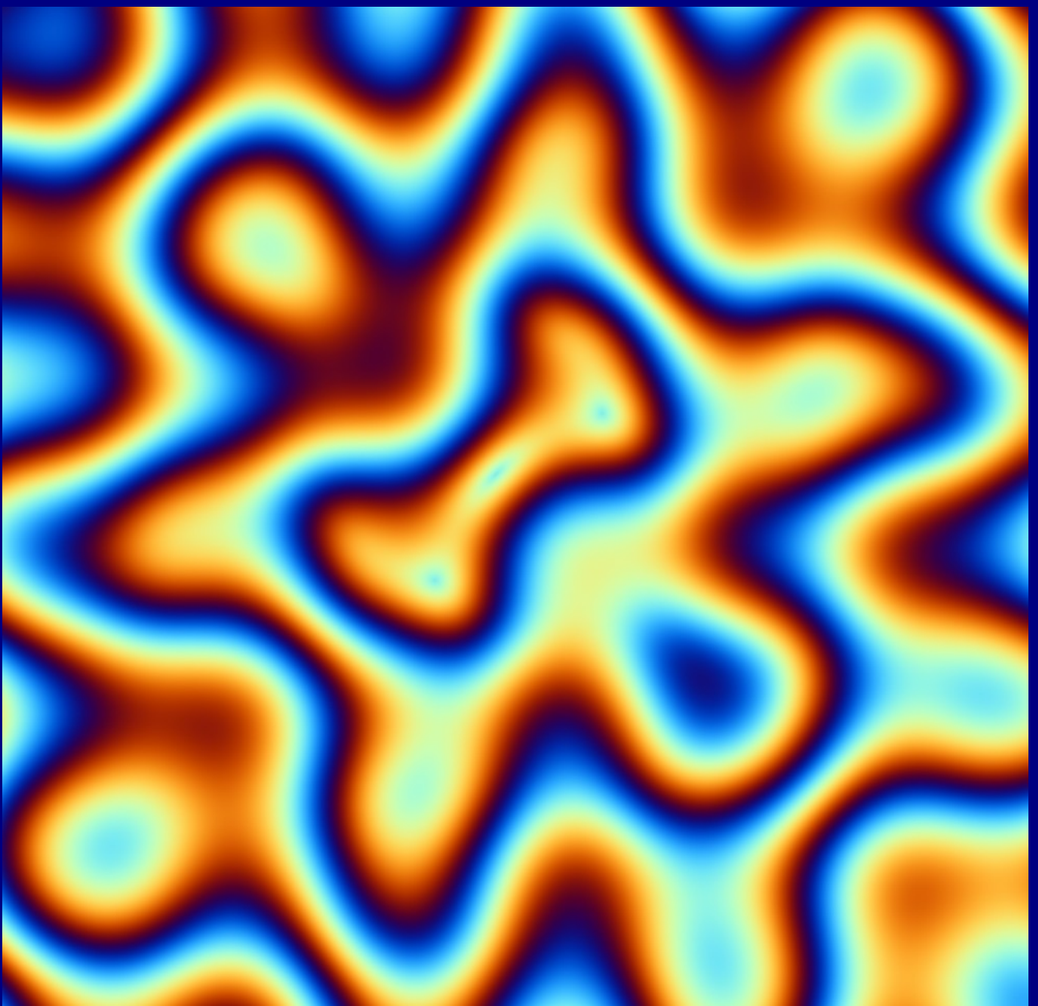
Plot functions

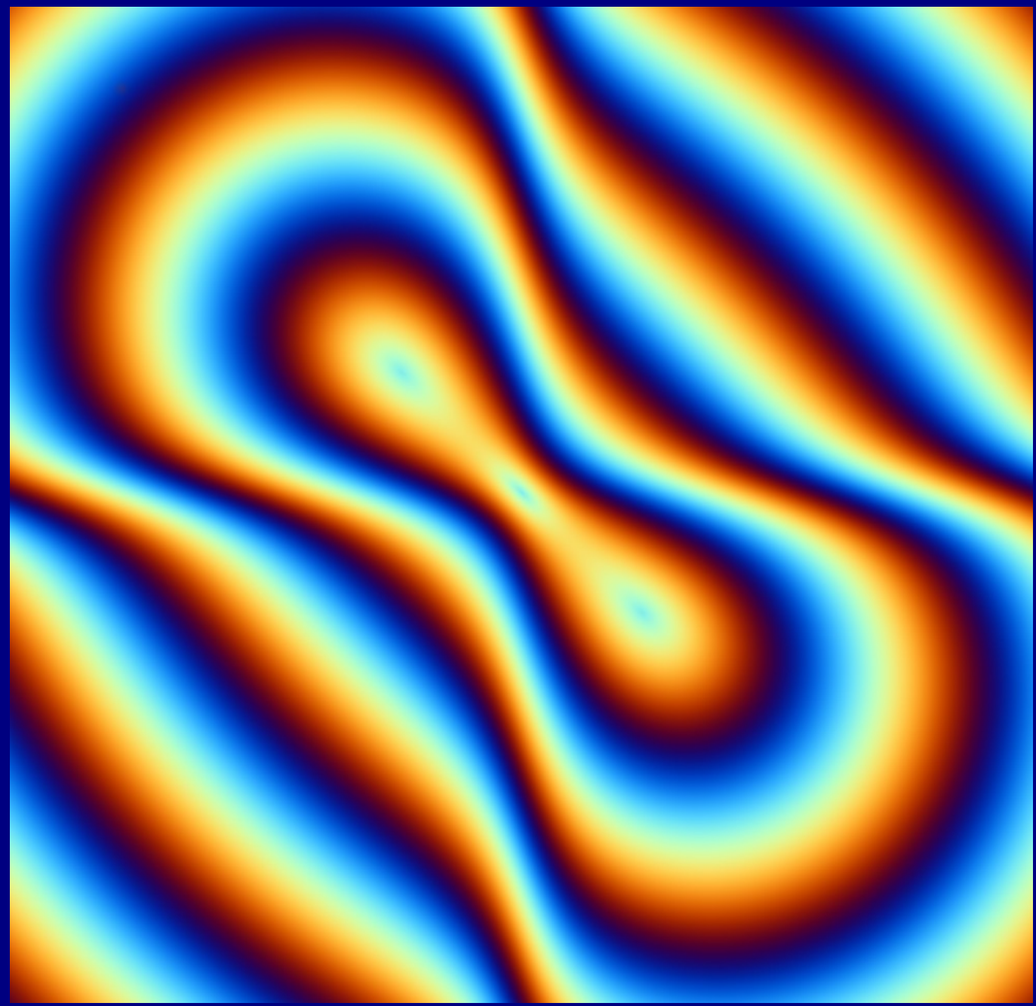
- Combining the graphs for f(x)-y and f(y)-x with +



Plot functions

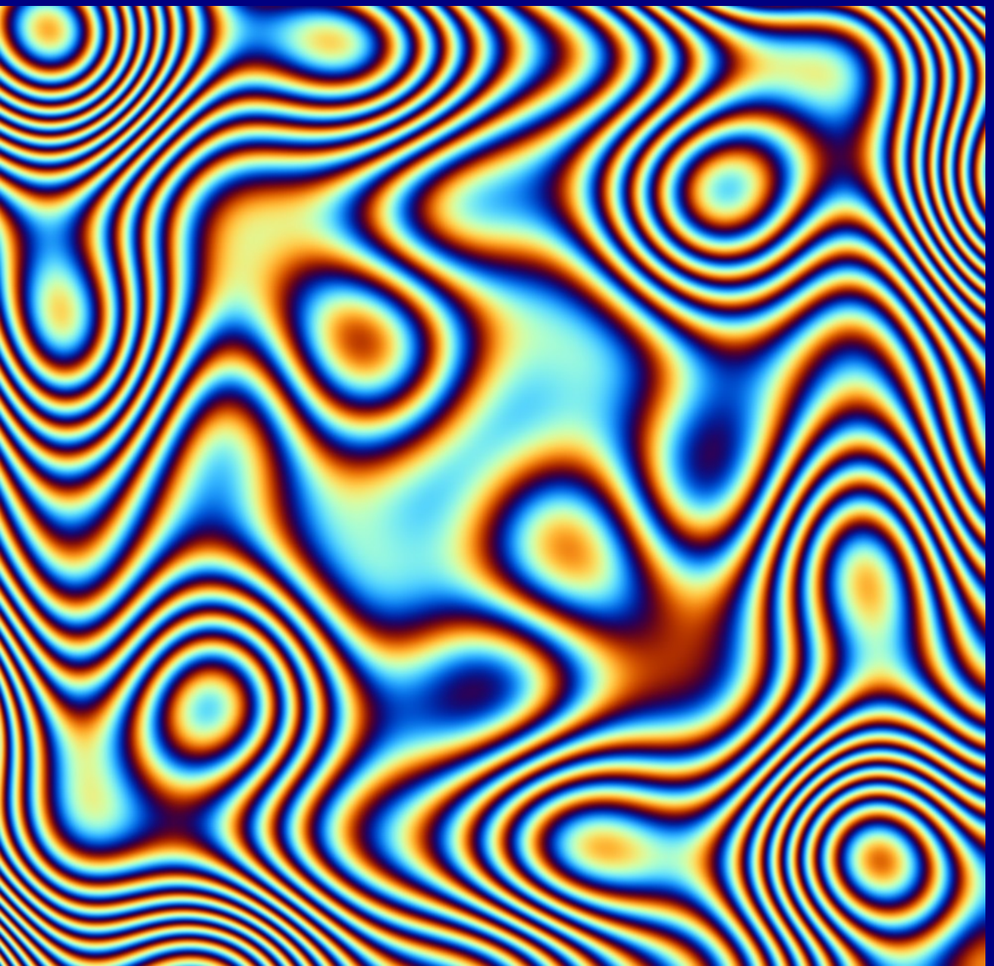
- Combining the graphs for f(x)-y and f(y)-x with Pythagoras

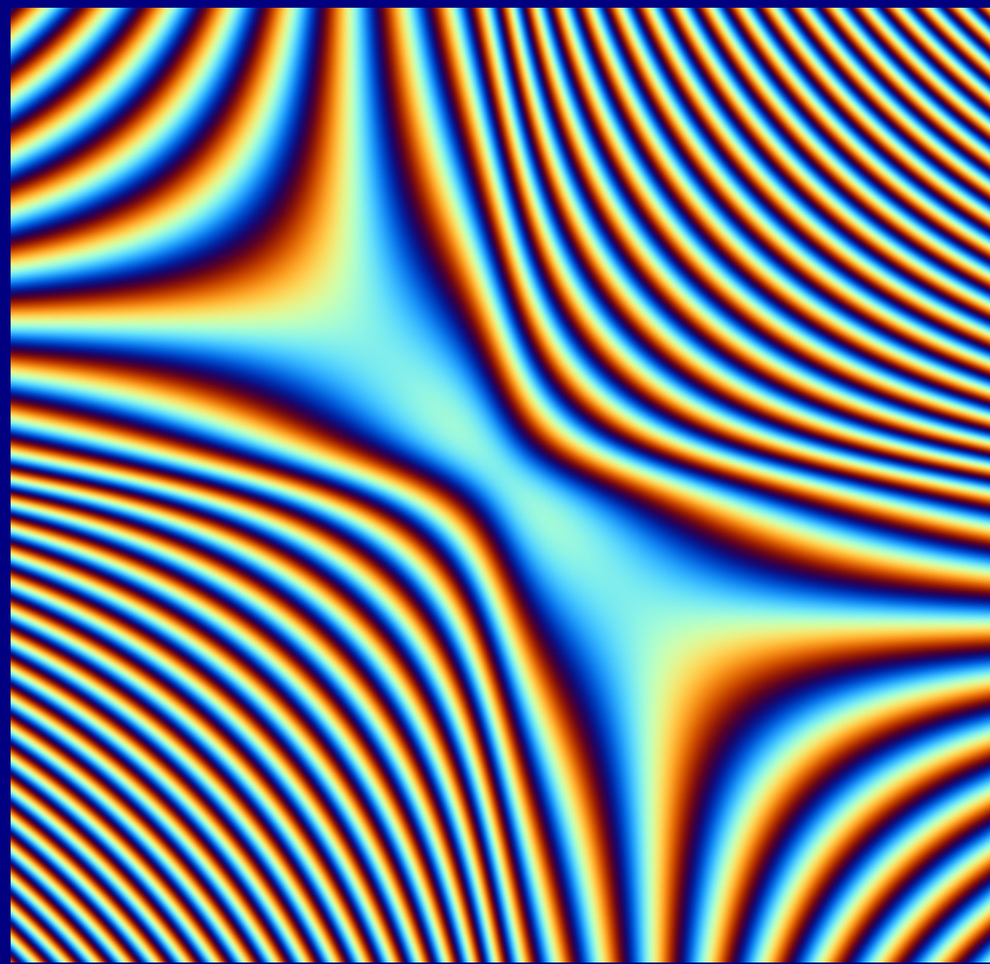




Plot functions

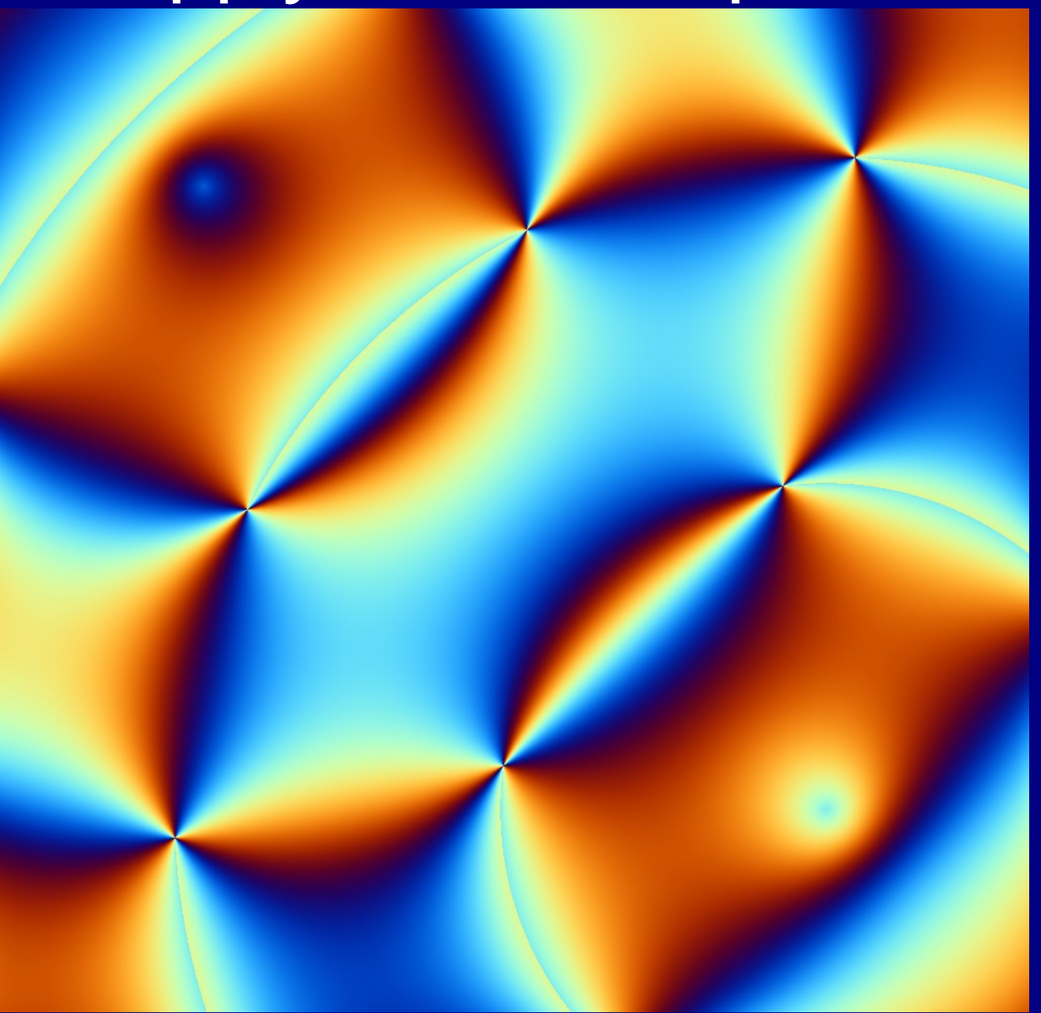
- Combining the graphs for f(x)-y and f(y)-x with Multiplication

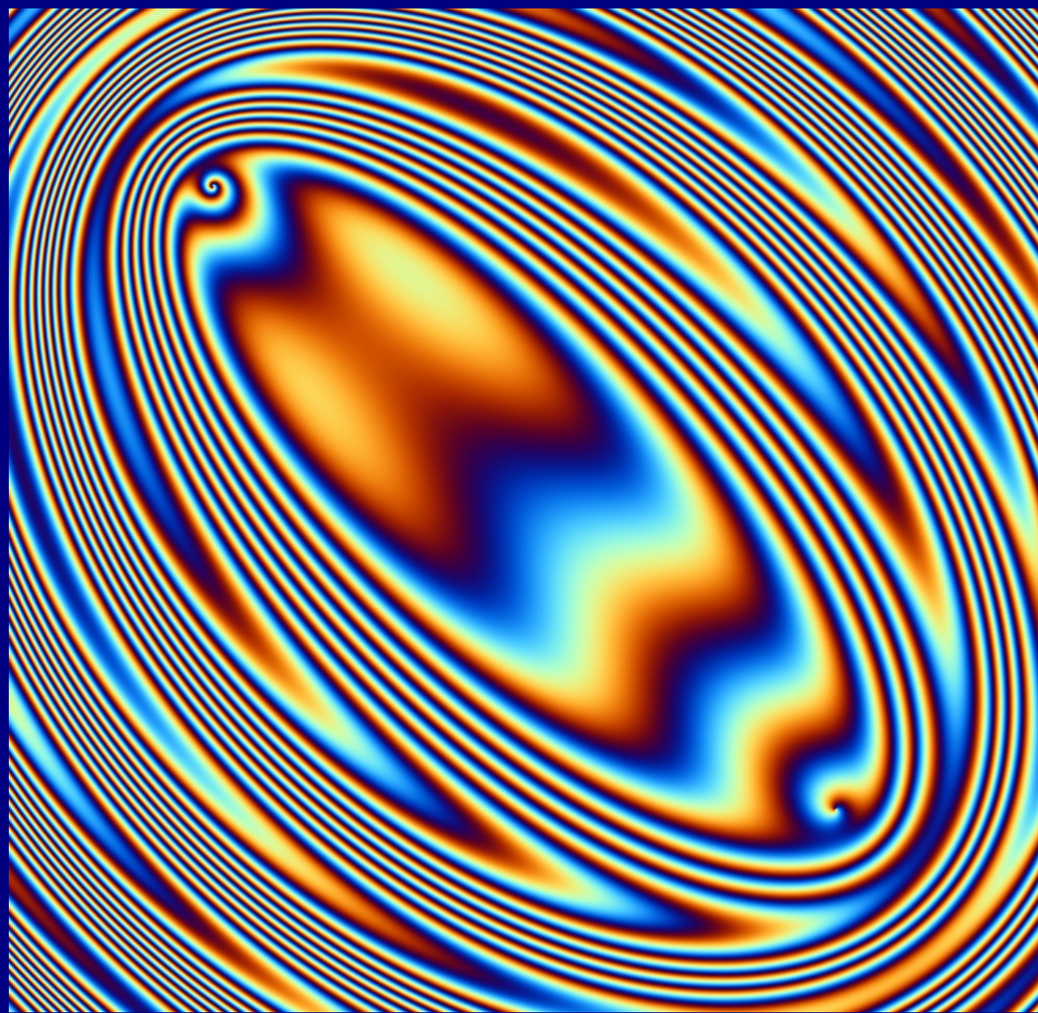




What works

- Apply more complex functions



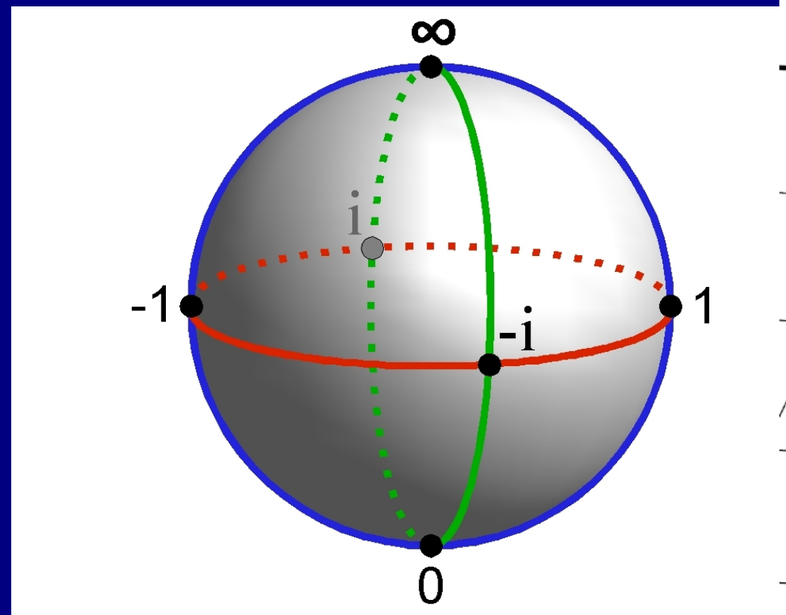


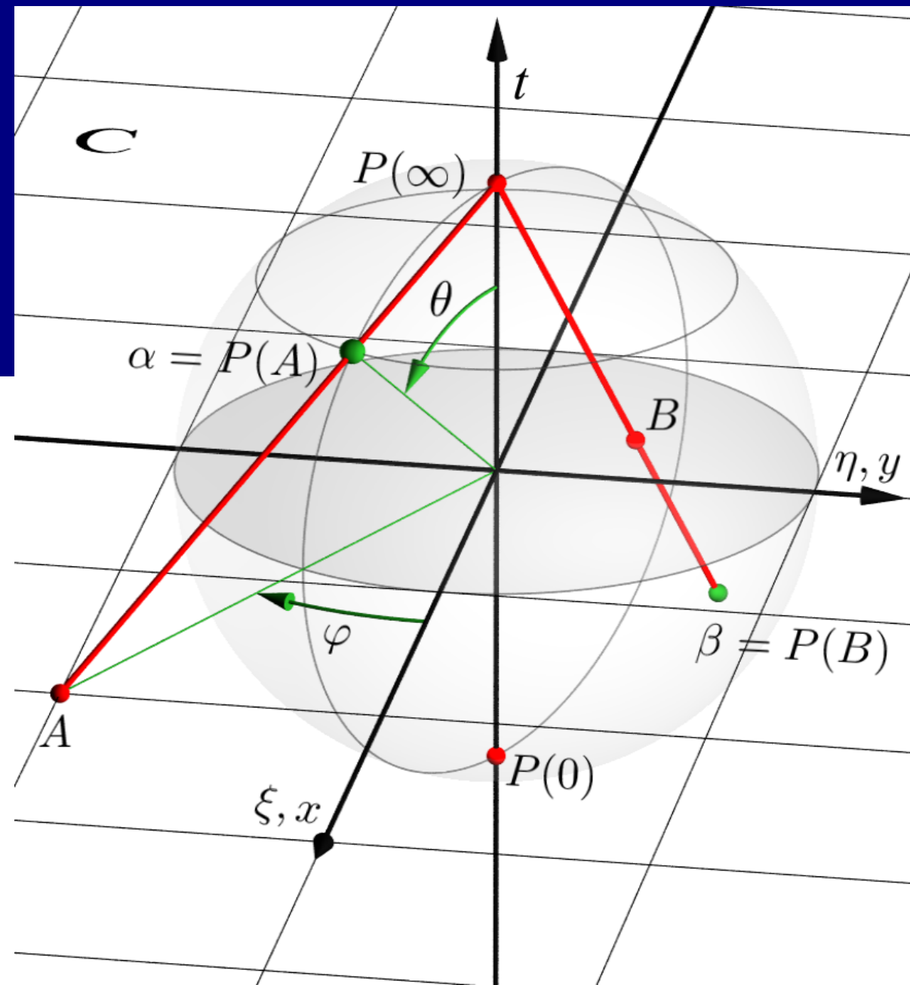
Other ideas

- Use additional functions to darken/lighten points of the image differently or to change saturation (with care)

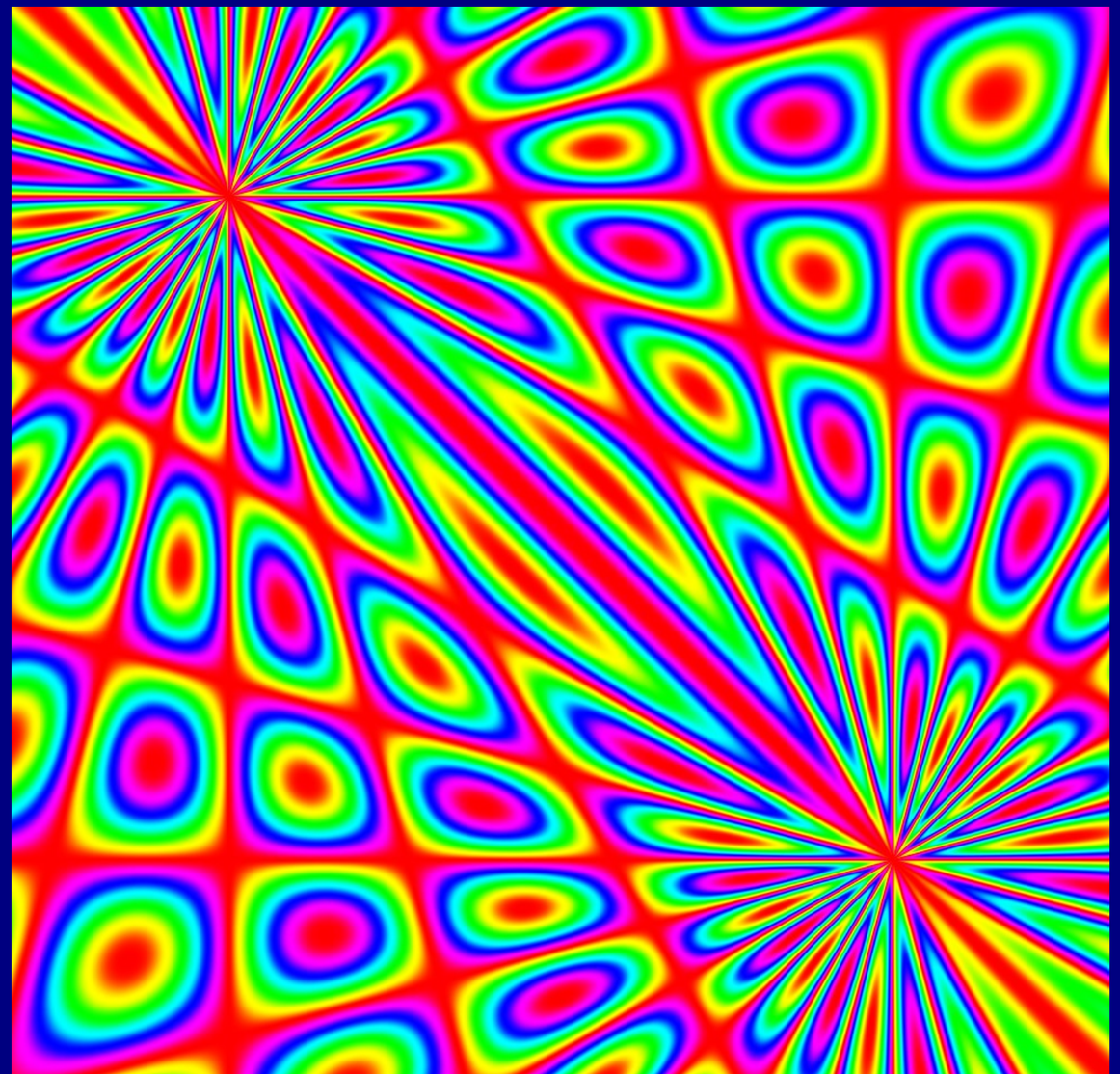
Riemann Sphere

- Use functions (x,y)->(x,y) (or complex functions if you like)
- Use Riemann Sphere
- Use color encoding
- 0/infinity = red
- +/- i = blue
- +/- 1 = green





Use HSV instead of RGB



Karl Brodowsky

http://brodowsky.it-sky.net/

http://www.it-sky-consulting.com/

https://github.com/bk1/clojure-art

@bk1_168 #reClojure #ClojureArt

Questions

??????????

The End