TexSyn

A library for procedural texture synthesis

LazyPredator

A library for evolutionary computation

Craig Reynolds — *preliminary* — November 18, 2020

Interactive Evolution of Camouflage

Craig Reynolds*
Sony Computer Entertainment,
US R&D

Keywords
Biology, interactive hybrid systems, evolutionary computation, genetic programming, texture synthesis, vision

A version of this paper with color figures is available online at http://dx.doi.org/10.1162/artl_a_00023. Subscription required.

Abstract This article presents an abstract computation model of the evolution of camouflage in nature. The 2D model uses evolved textures for *prey*, a background texture representing the *environment*, and a visual *predator*. A human observer, acting as the predator, is shown a *cohort* of 10 evolved textures overlaid on the background texture. The observer clicks on the five most conspicuous prey to remove ("eat") them. These lower-fitness textures are removed from the population and replaced with newly bred textures. Biological morphogenesis is represented in this model by *procedural texture synthesis*. Nested expressions of generators and operators form a texture description language. Natural evolution is represented by *genetic programming* (GP), a variant of the *genetic algorithm*. GP searches the space of texture description programs for those that appear least conspicuous to the predator.

1 Introduction

That animals often resemble their environment has been observed since ancient times. This sometimes startling visual similarity highlights the adaptation of life to its environment. Since the earliest publication on evolution, camouflage has been cited as a key illustration of natural selection's effect:

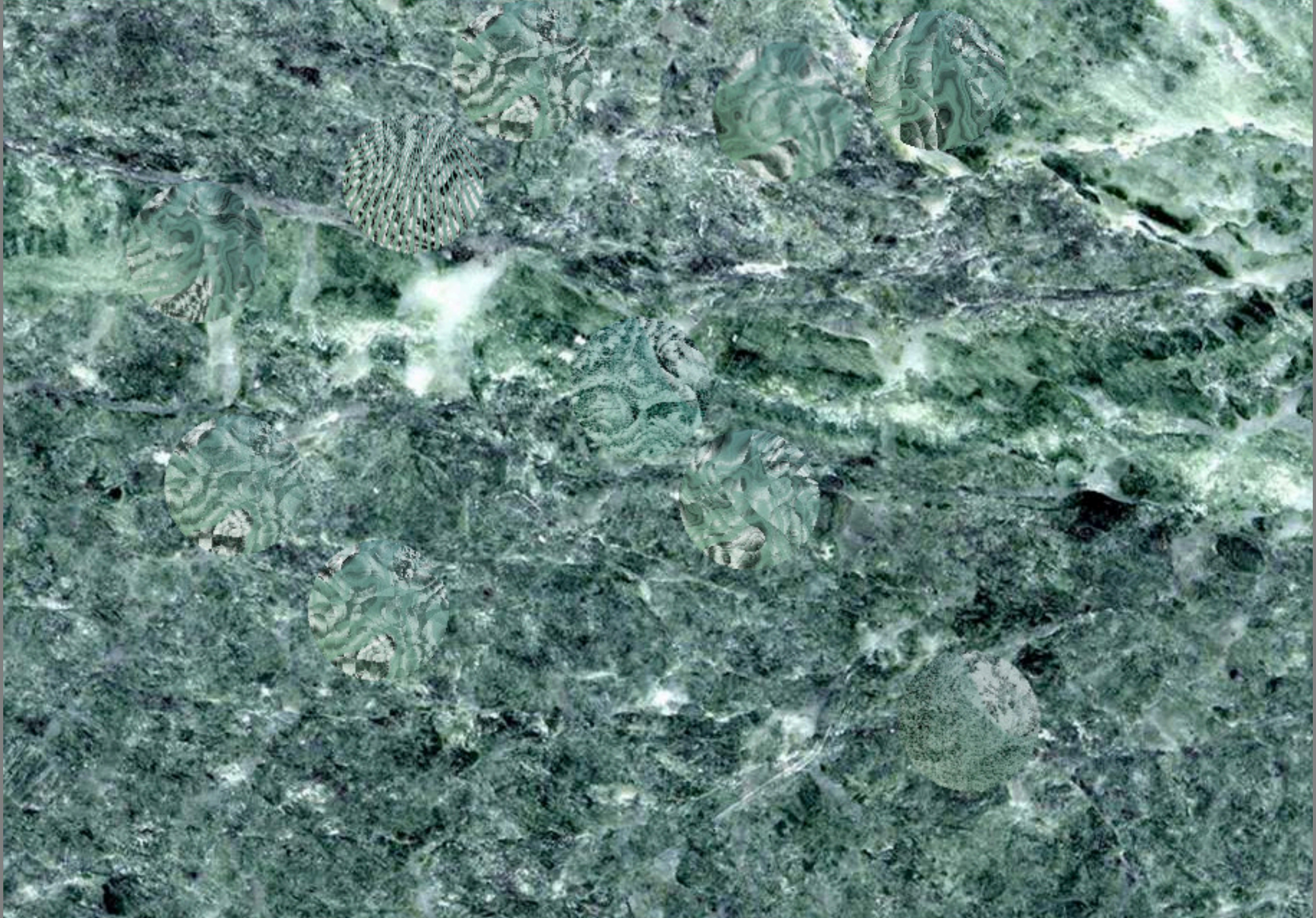
> When we see leaf-eating insects green, and bark-feeders mottled-gray; alpine ptarmigan white in winter, the red-grouse the colour of heather, and the black-grouse that of peaty earth, we must believe that these tints are of service to these birds and insects in preserving them from danger.—Charles Darwin, 1859, *On the Origin of Species* [9]

Natural camouflage appears to result from coevolution between predator and prey. Many predators use vision to locate their prey, so prey have a survival advantage if they are harder to see. Predators with superior vision are better able to find prey, giving them a survival advantage. Over time this leads to well-camouflaged prey and to predators with excellent eyesight and a talent for breaking camouflage.

The hypothesis for these experiments was that selection pressure from a visual predator will gradually eliminate the most conspicuous (least well camouflaged) prey from the evolving population. Prey would then converge on more effective camouflage. The results presented here in Figures 1, 2, and 3 lend support to this idea and point the way to more powerful human-computer

* Sony Computer Entertainment, US R&D, 989 E. Hillsdale Blvd., Foster City, California, USA. E-mail: craig_reynolds@playstation.sony.com

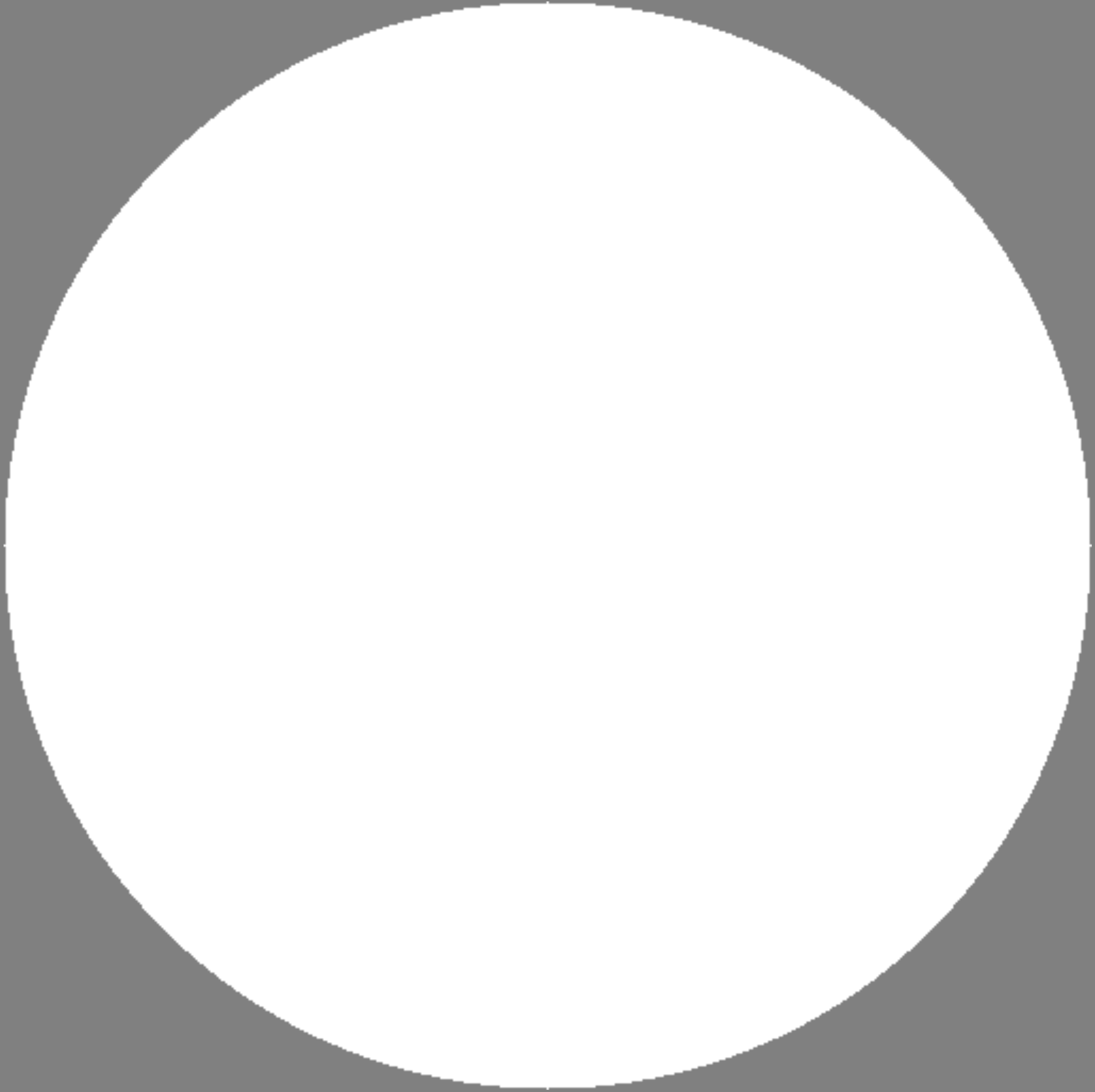
© 2011 Massachusetts Institute of Technology



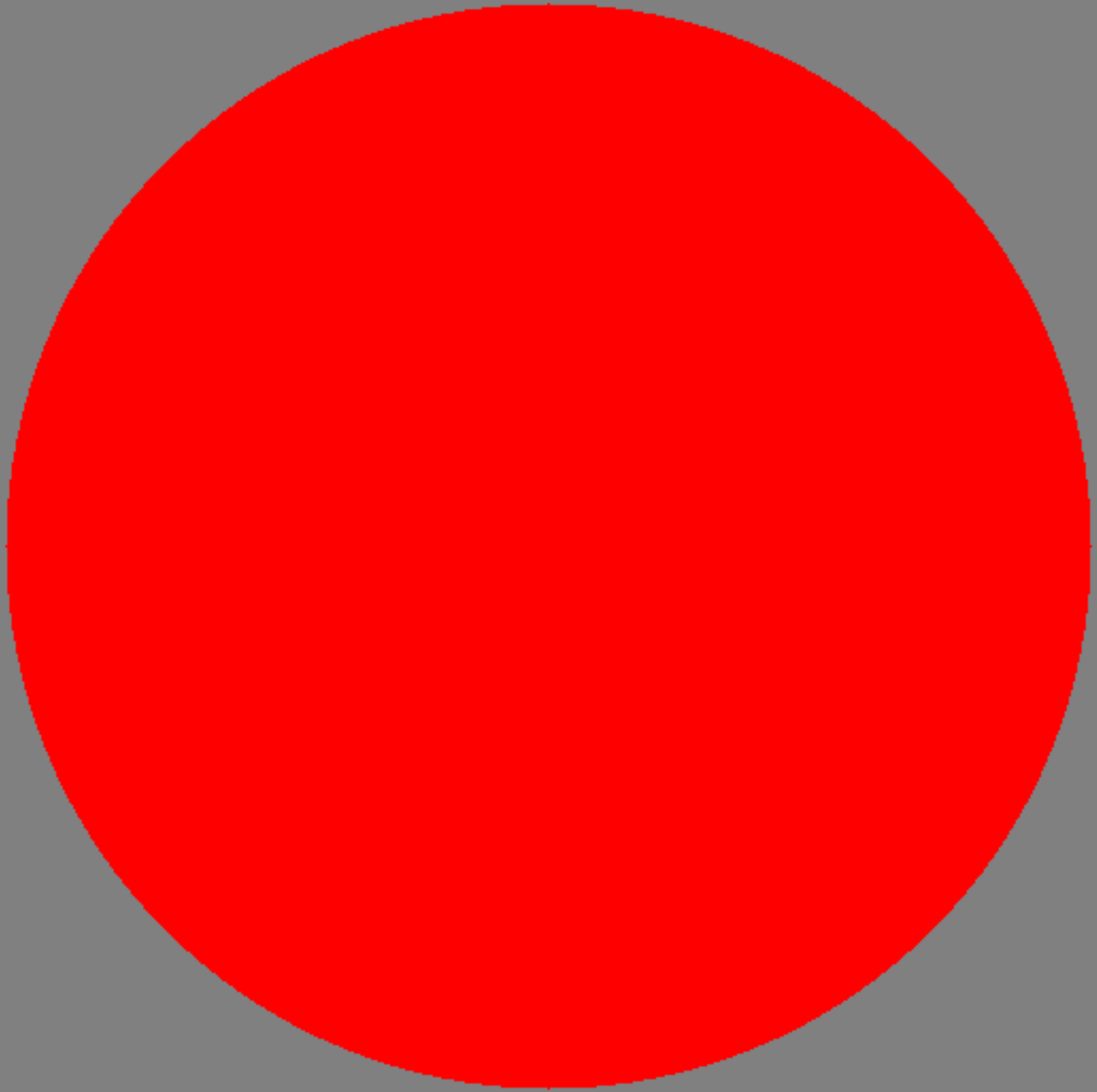
TexSyn

new library for procedural texture synthesis

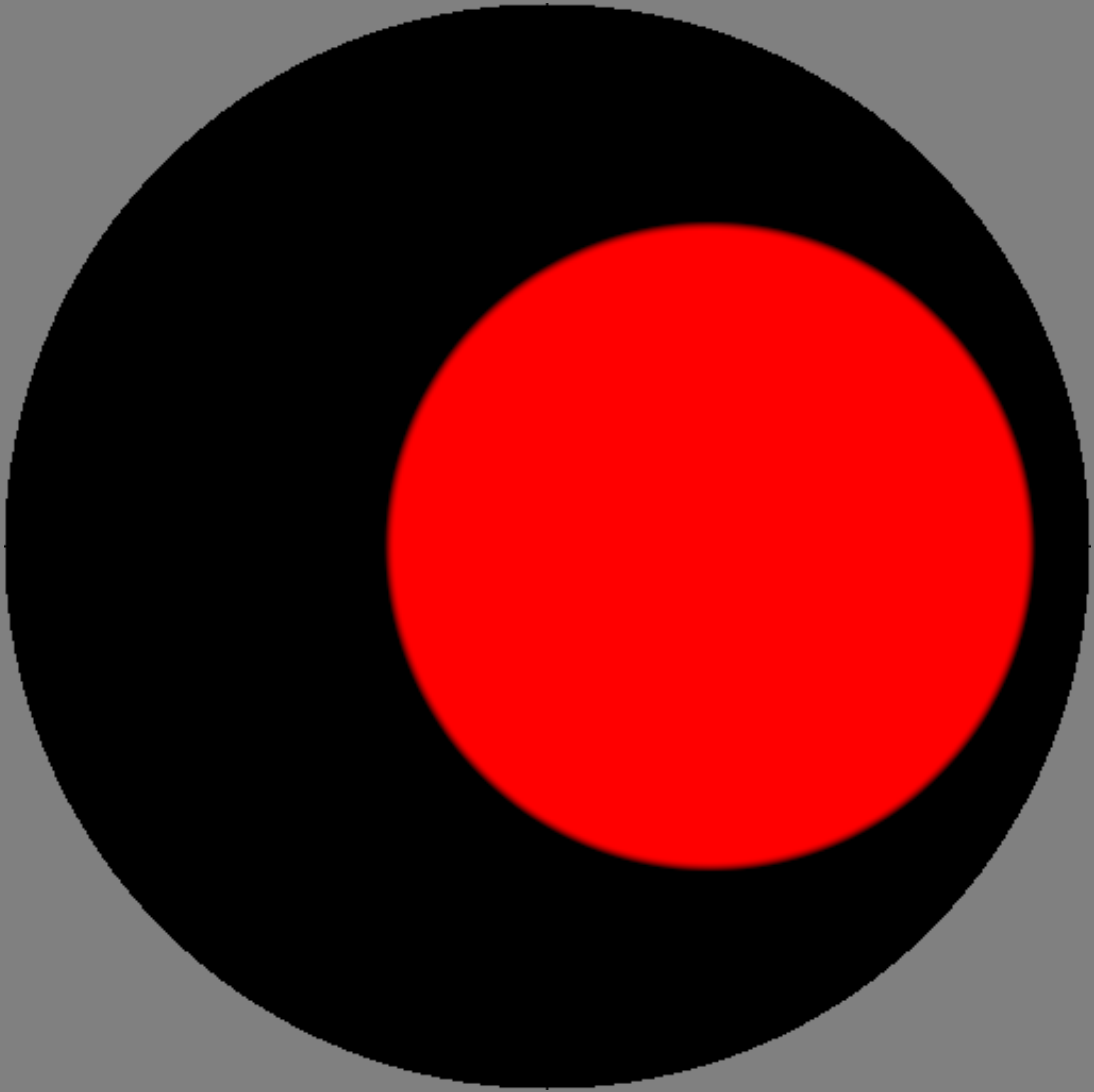
```
Uniform white(1);
```



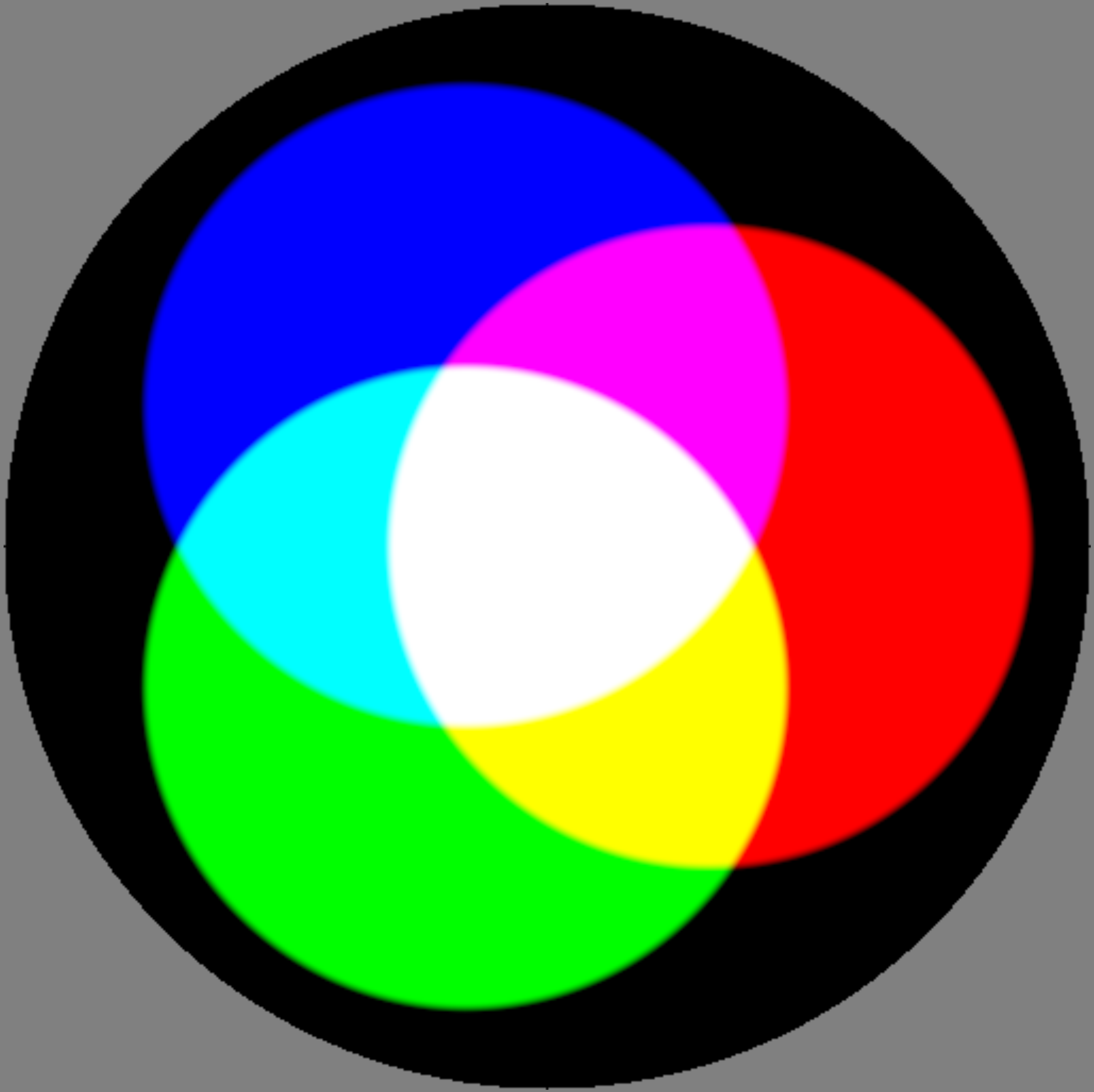
```
Uniform red(1, 0, 0);
Uniform green(0, 1, 0);
Uniform blue(0, 0, 1);
```



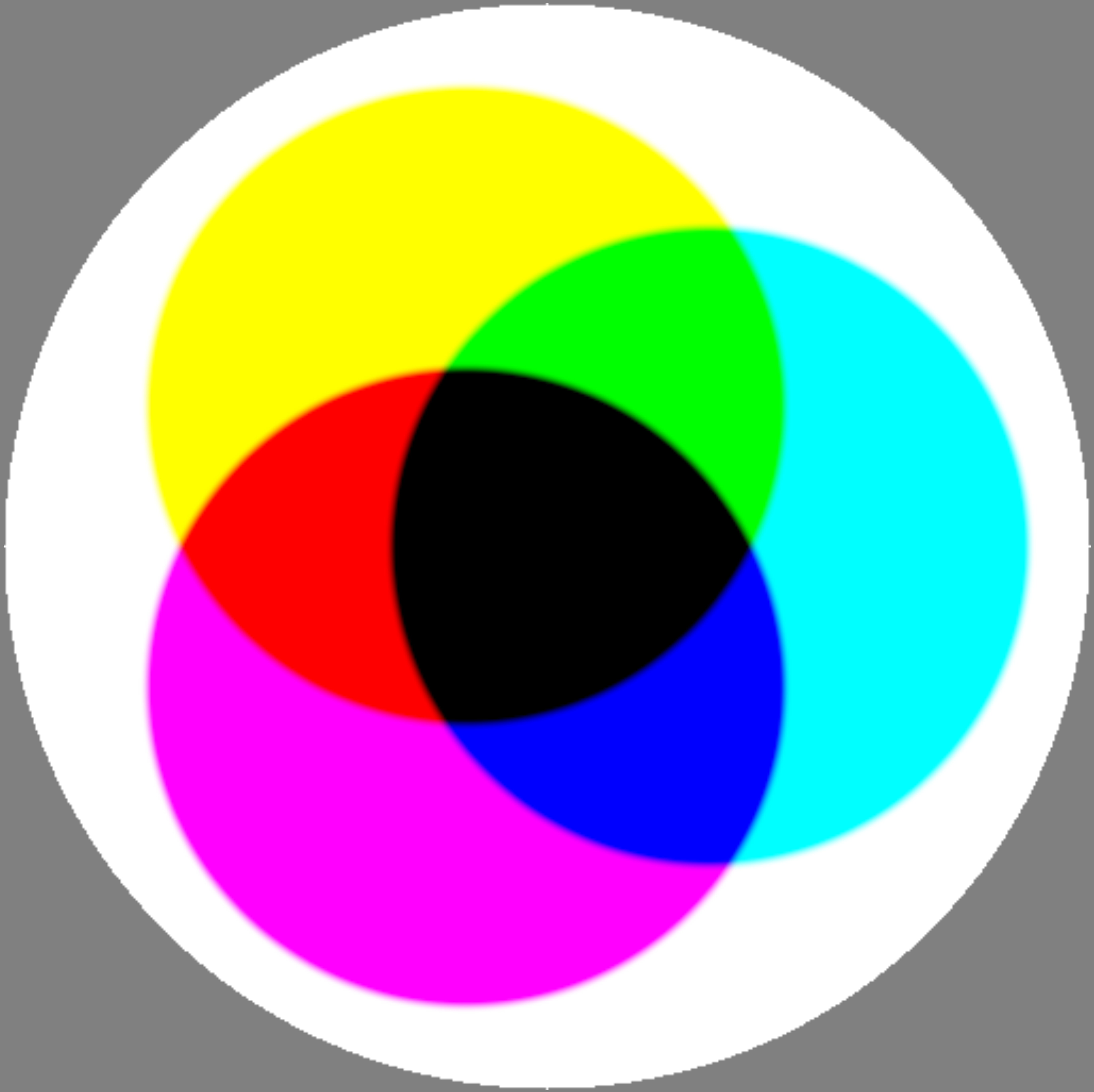
```
Vec2 p1(0.3, 0);
Vec2 p2 = p1.rotate(2 * pi / 3);
Vec2 p3 = p2.rotate(2 * pi / 3);

float ro = 0.6;
float ri = ro - 0.02;

Spot rs(p1, ri, red, ro, black);    →
Spot gs(p2, ri, green, ro, black);
Spot bs(p3, ri, blue, ro, black);
```

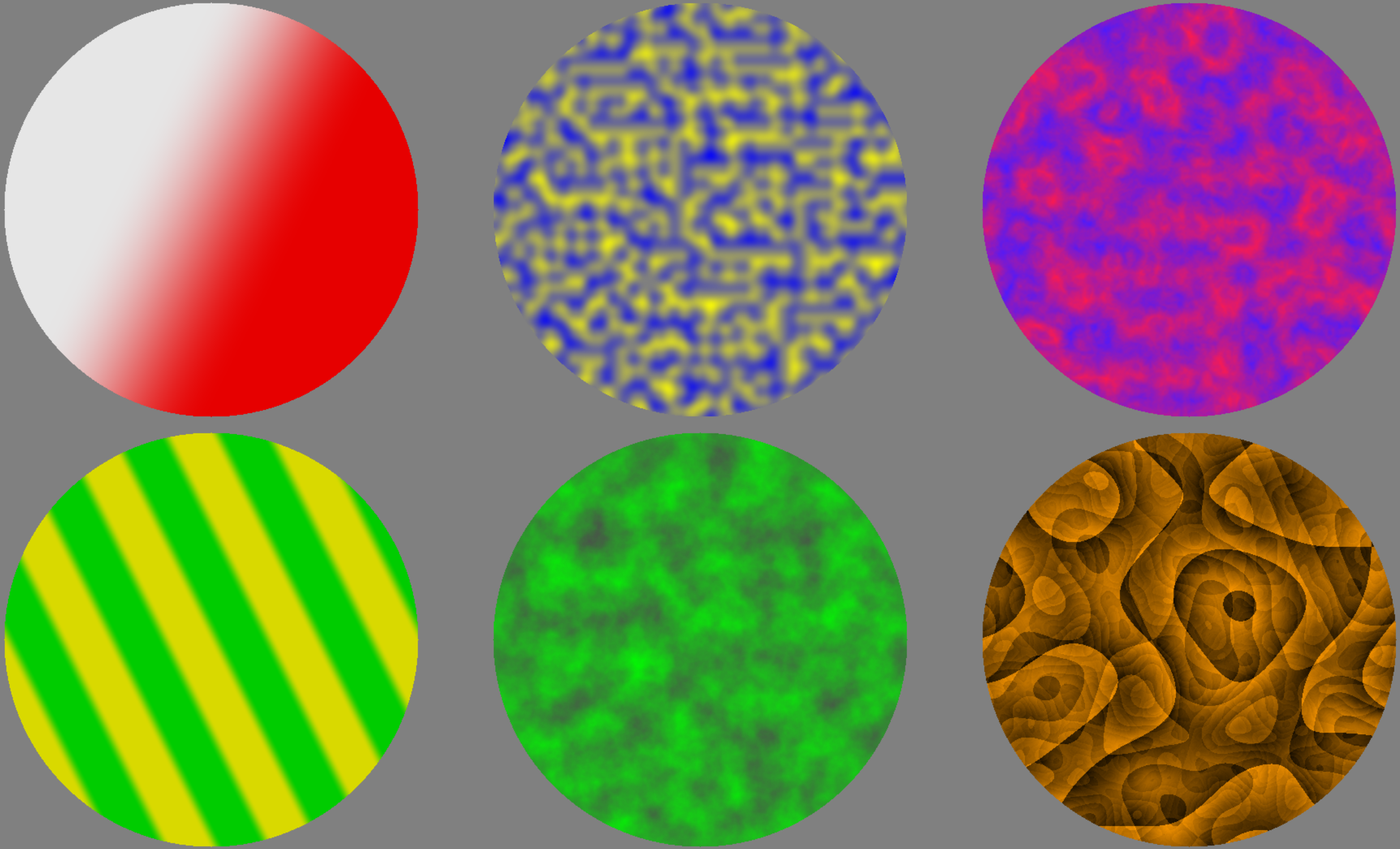


```
Add(rs, Add(gs, bs))
```

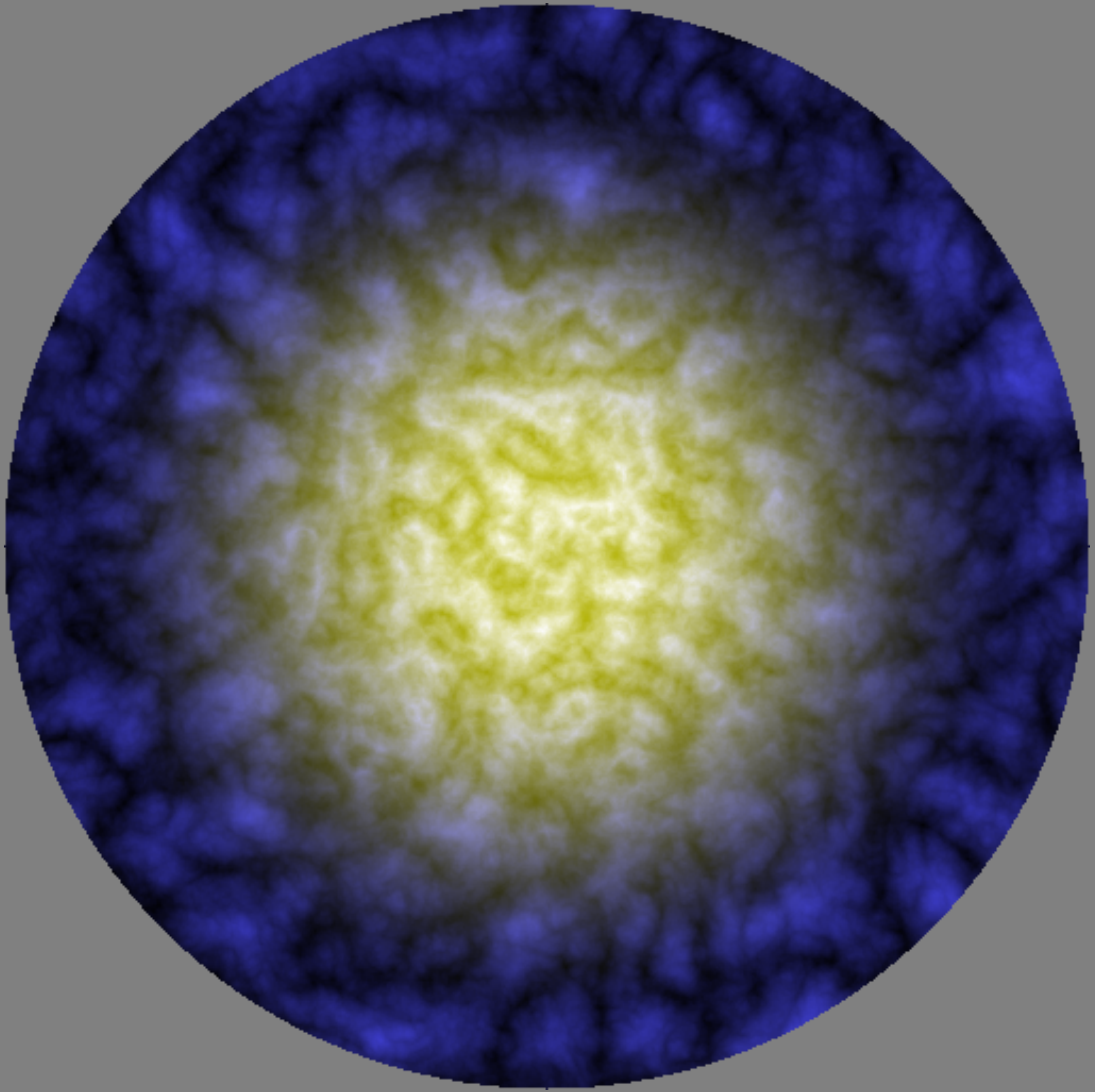


```
Multiply(Subtract(white, rs),
         Multiply(Subtract(white, gs),
                  Subtract(white, bs)))
```

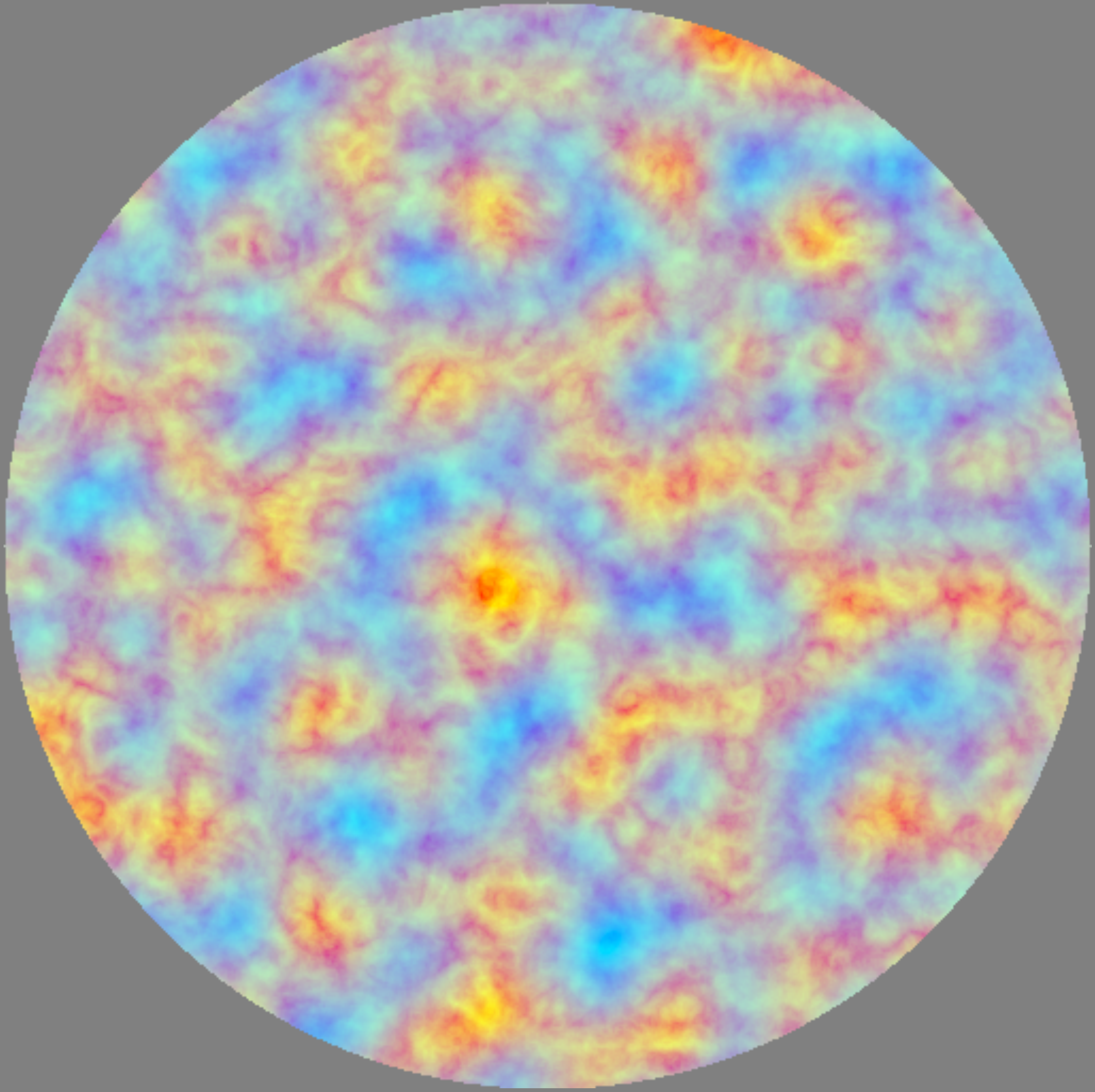




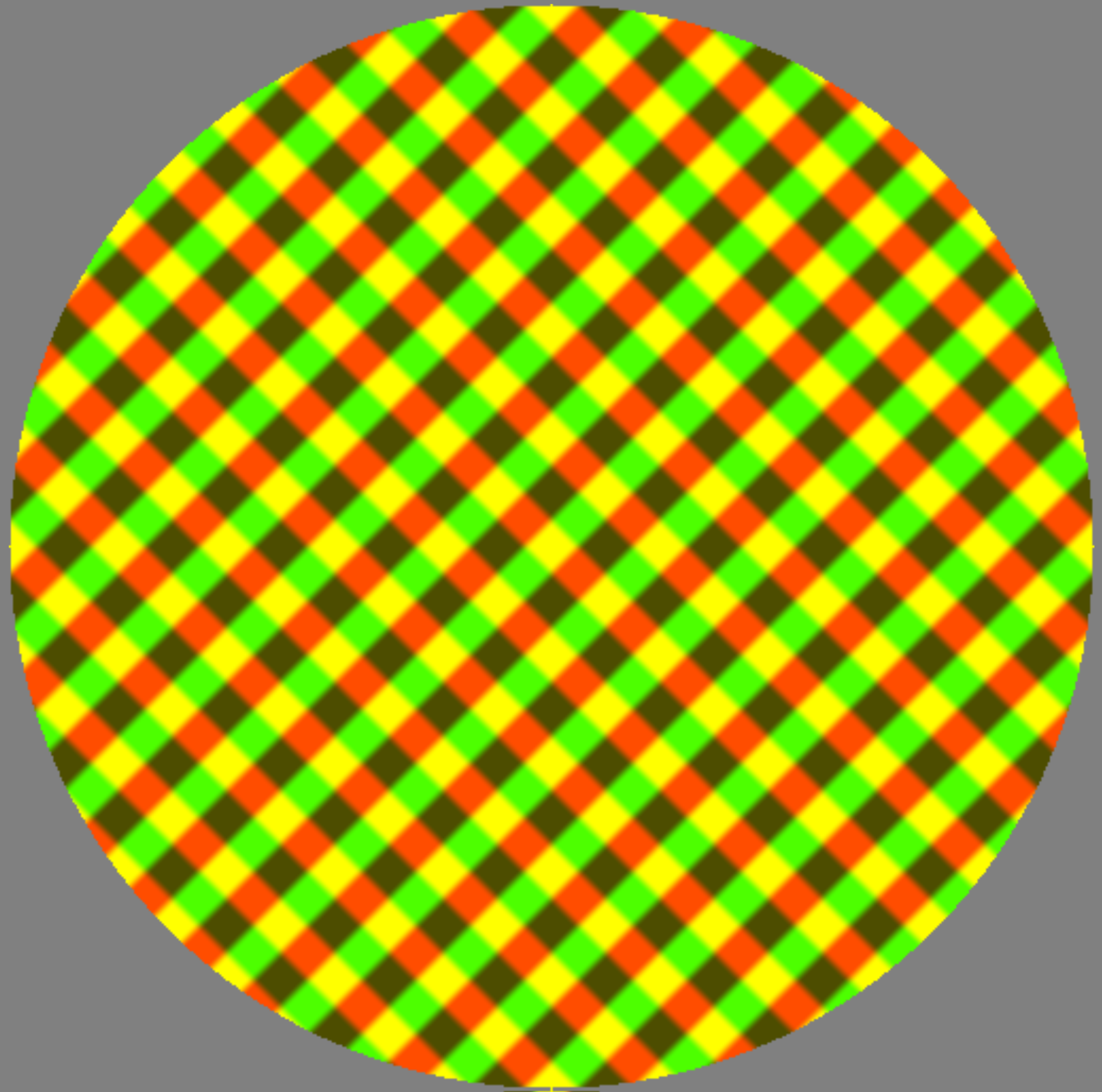
```
Spot(Vec2(0, 0),
     0.1,
     Furbulence(0.1,
                Vec2(1, 2),
                Uniform(1, 1, 1),
                Uniform(0.7, 0.7, 0)),
     0.9,
     Turbulence(0.2,
                Vec2(-5, 7),
                Uniform(0, 0, 0),
                Uniform(0.3, 0.3, 1))))
```



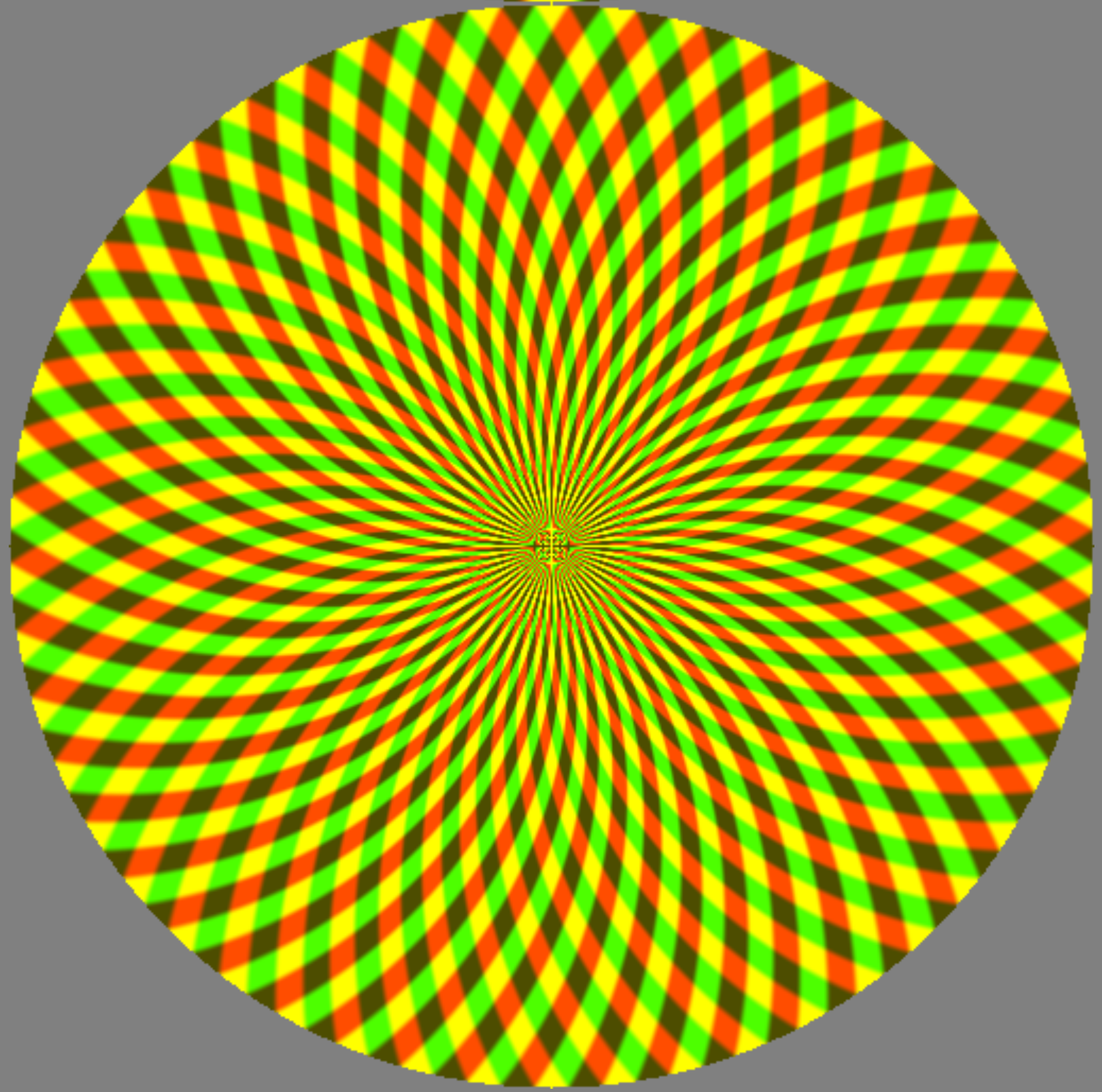
```
Noise(0.2,
      Vec2(1, 2),
      Furbulence(0.1, Vec2(), yellow, red),
      Brownian(0.1, Vec2(), blue, cyan))
```



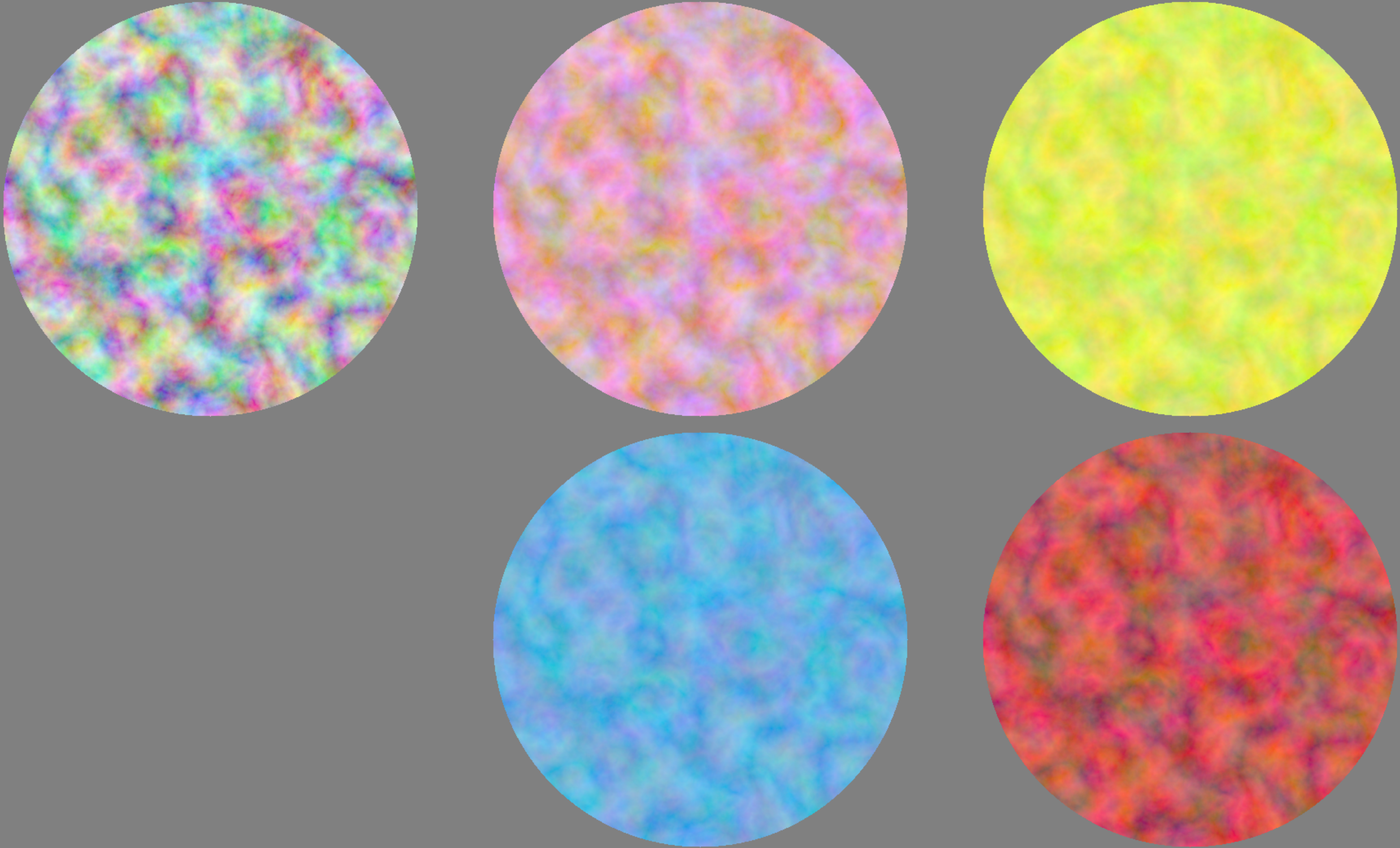
```
plaid = Add(Grating(Vec2(0, 0),
                    Color(1, 0, 0),
                    Vec2(0.1, 0.1),
                    Color(0.3, 0, 0), 0.3),
            Grating(Vec2(0, 0),
                    Color(0, 1, 0),
                    Vec2(-0.1, 0.1),
                    Color(0, 0.3, 0), 0.3))
```

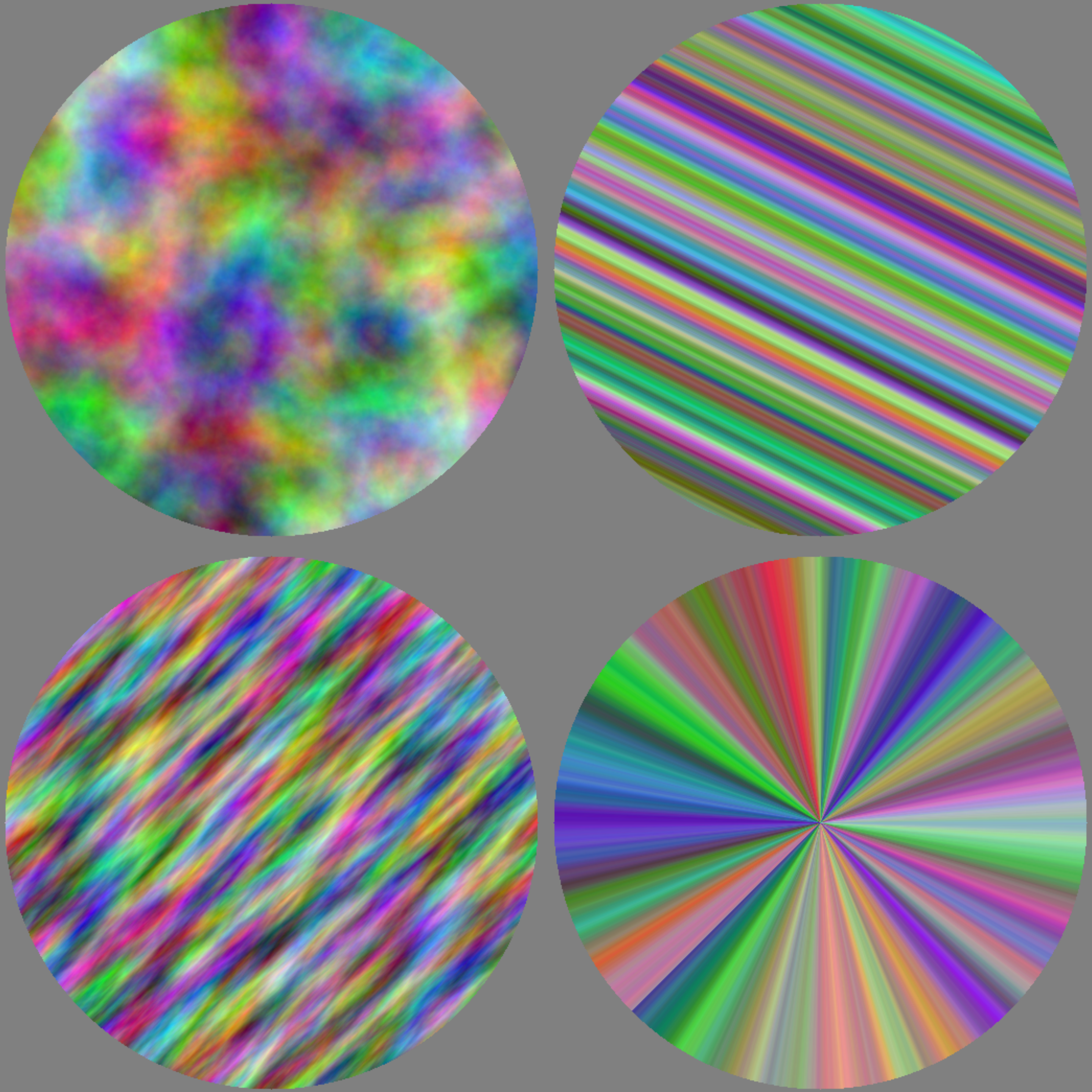


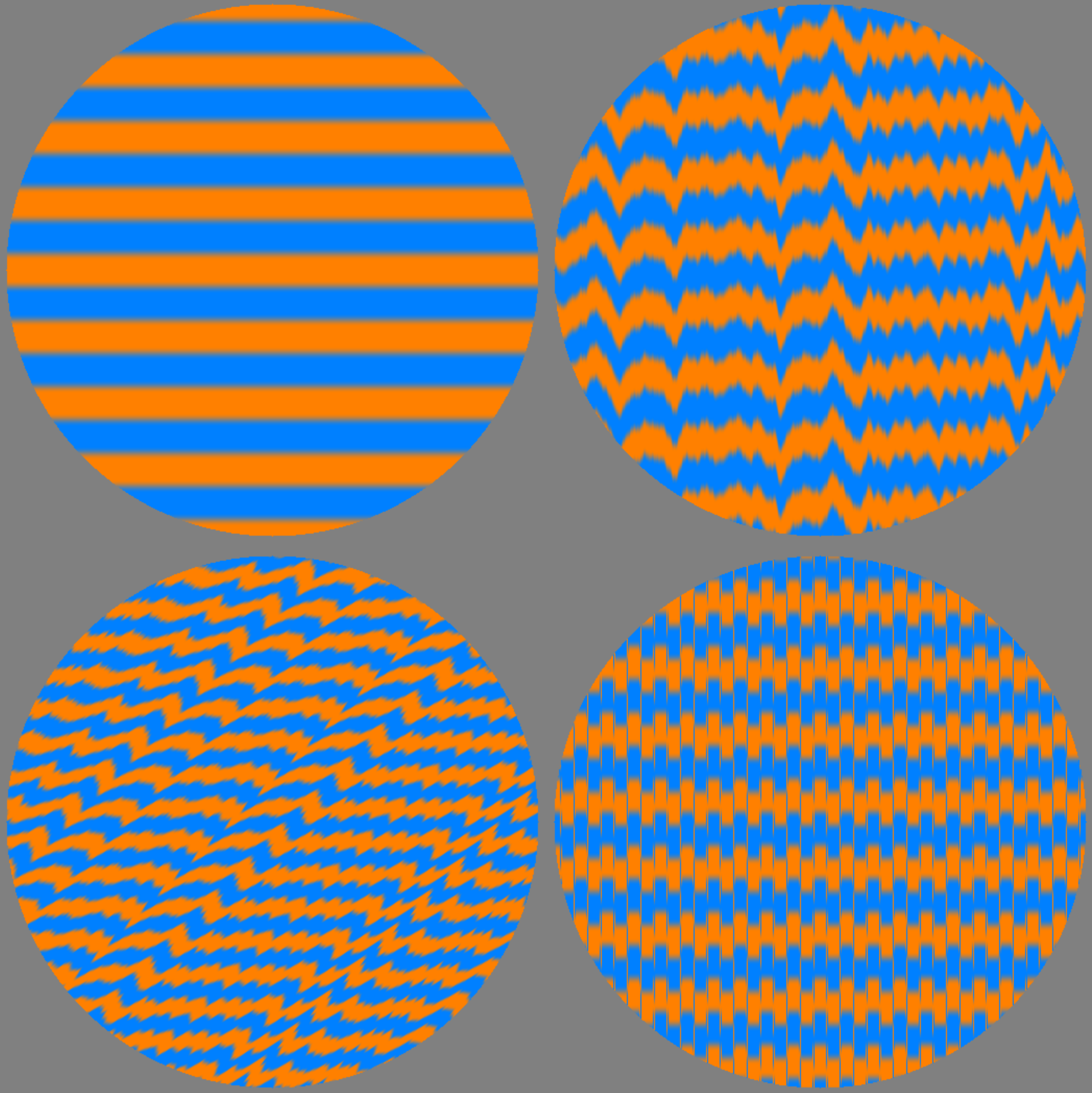
```
Wrap(5, Vec2(0, 0), Vec2(0, 1), plaid)
```

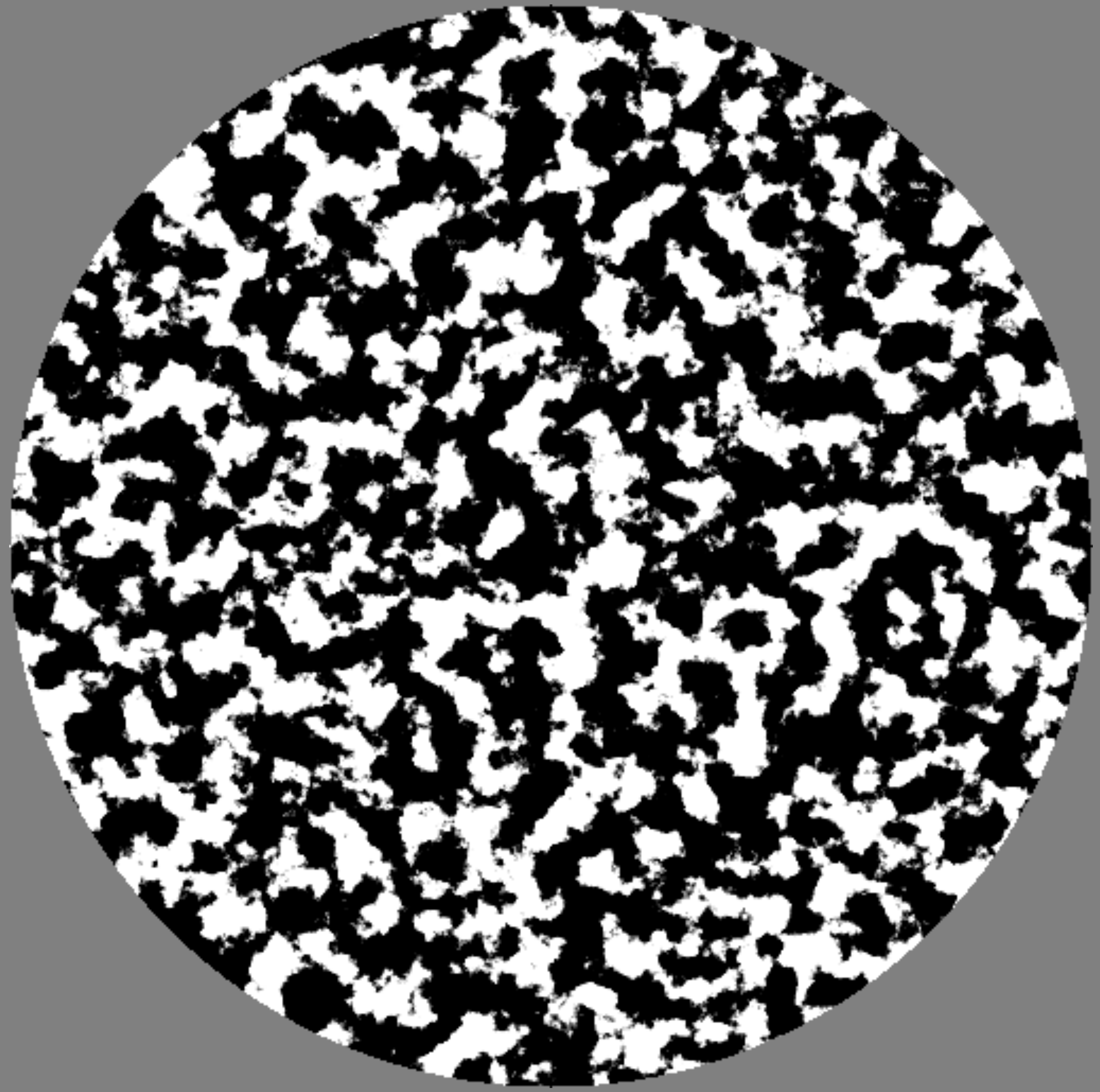
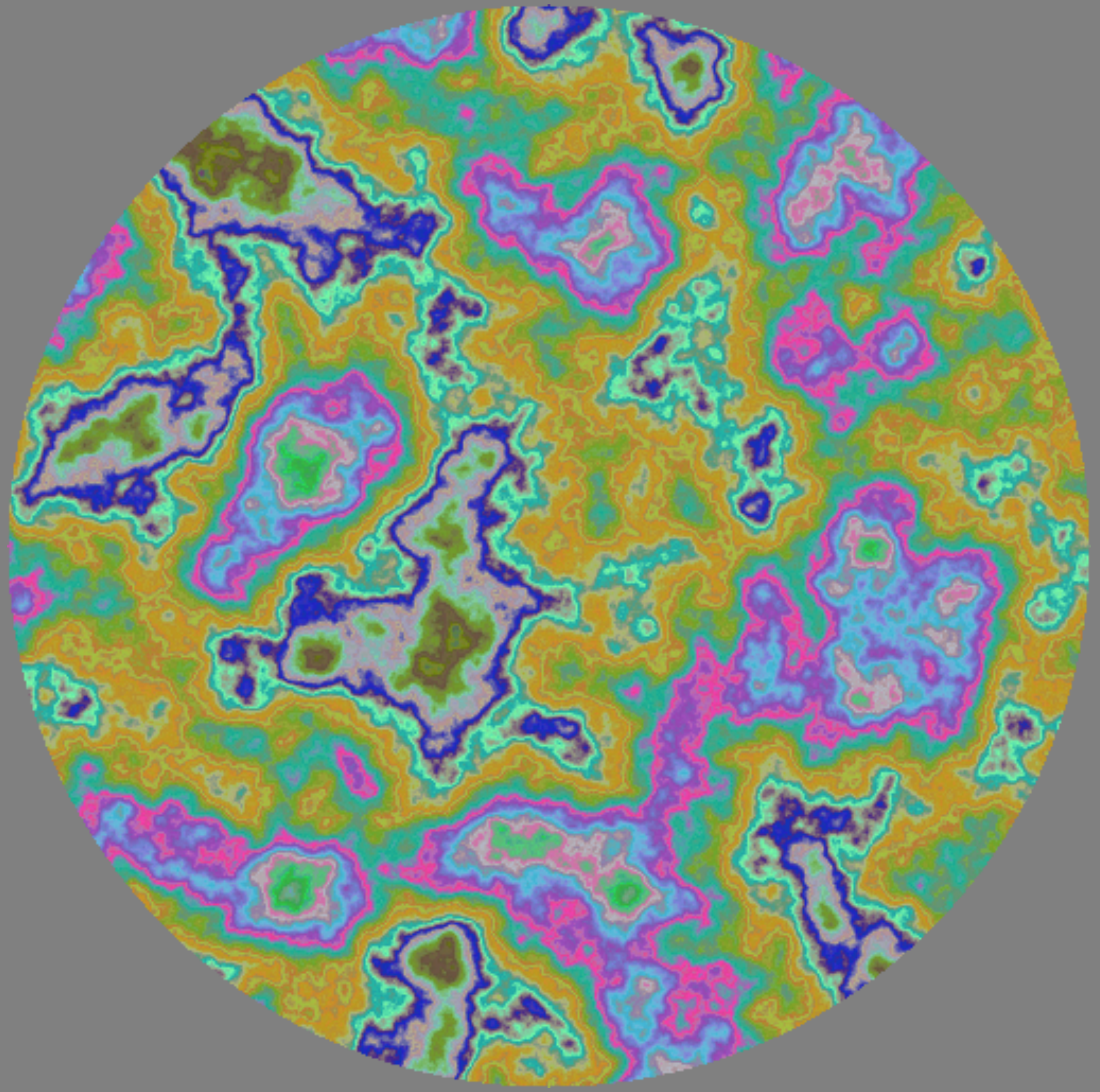


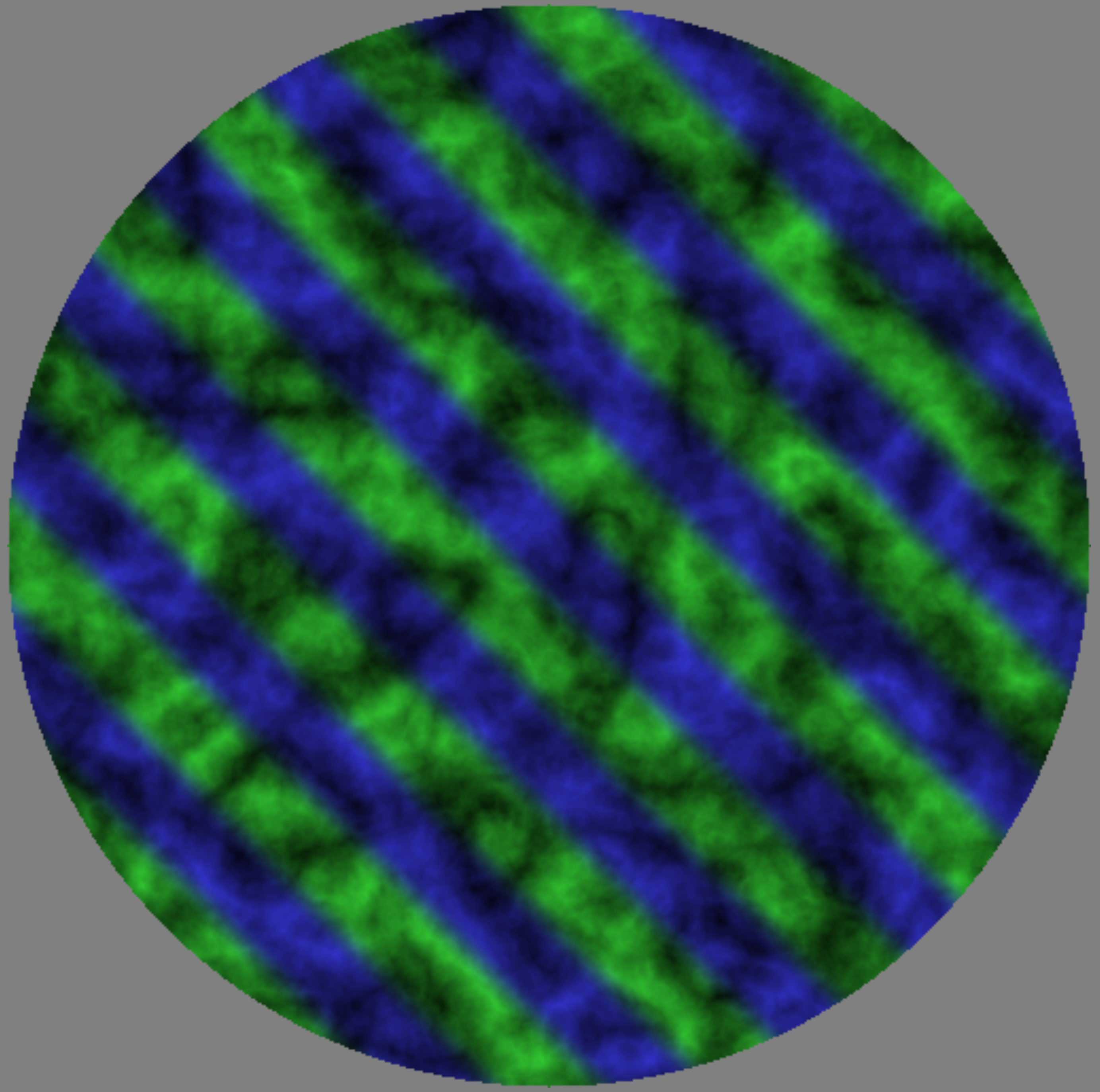
eye candy:
examples of operators

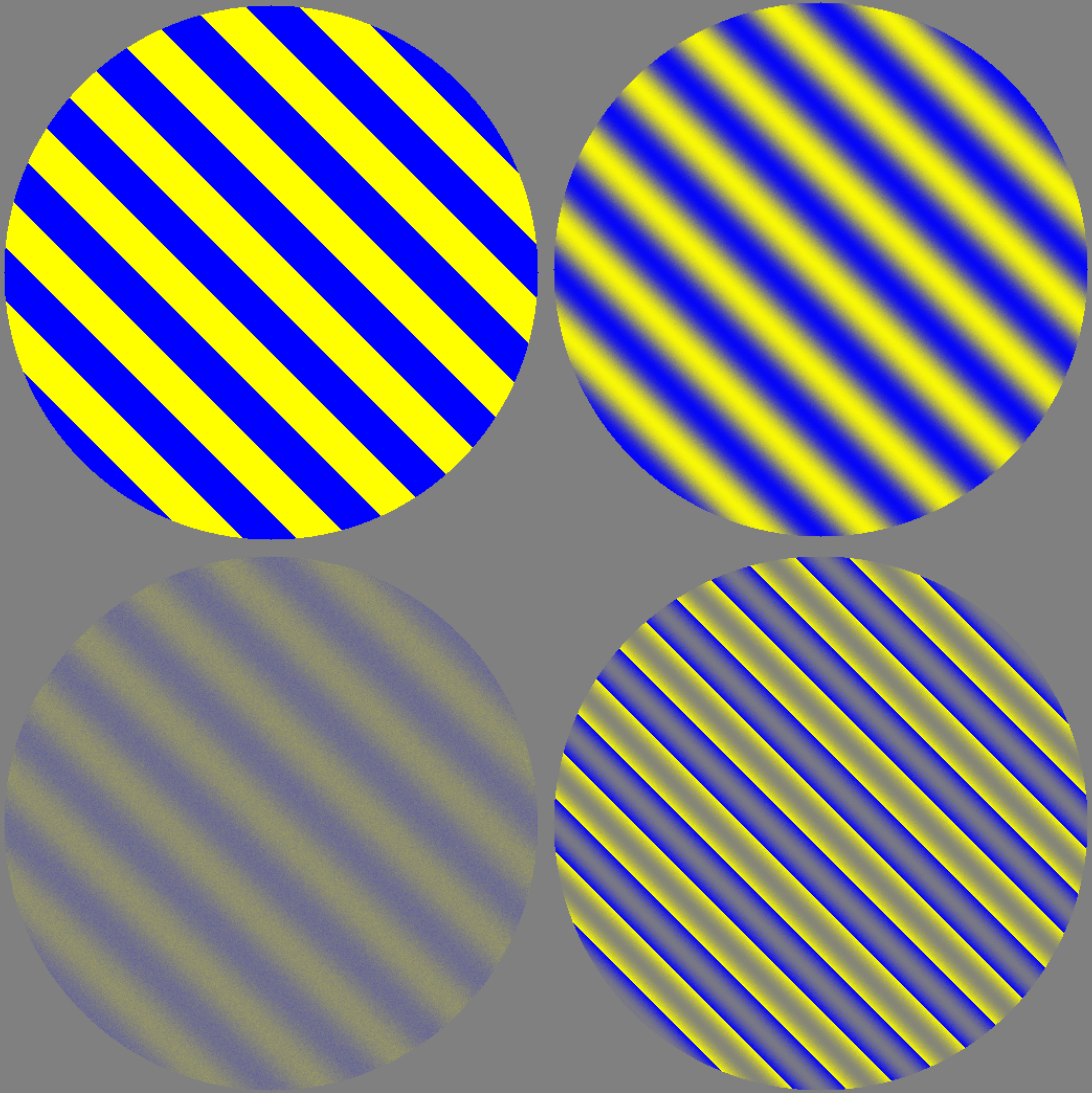


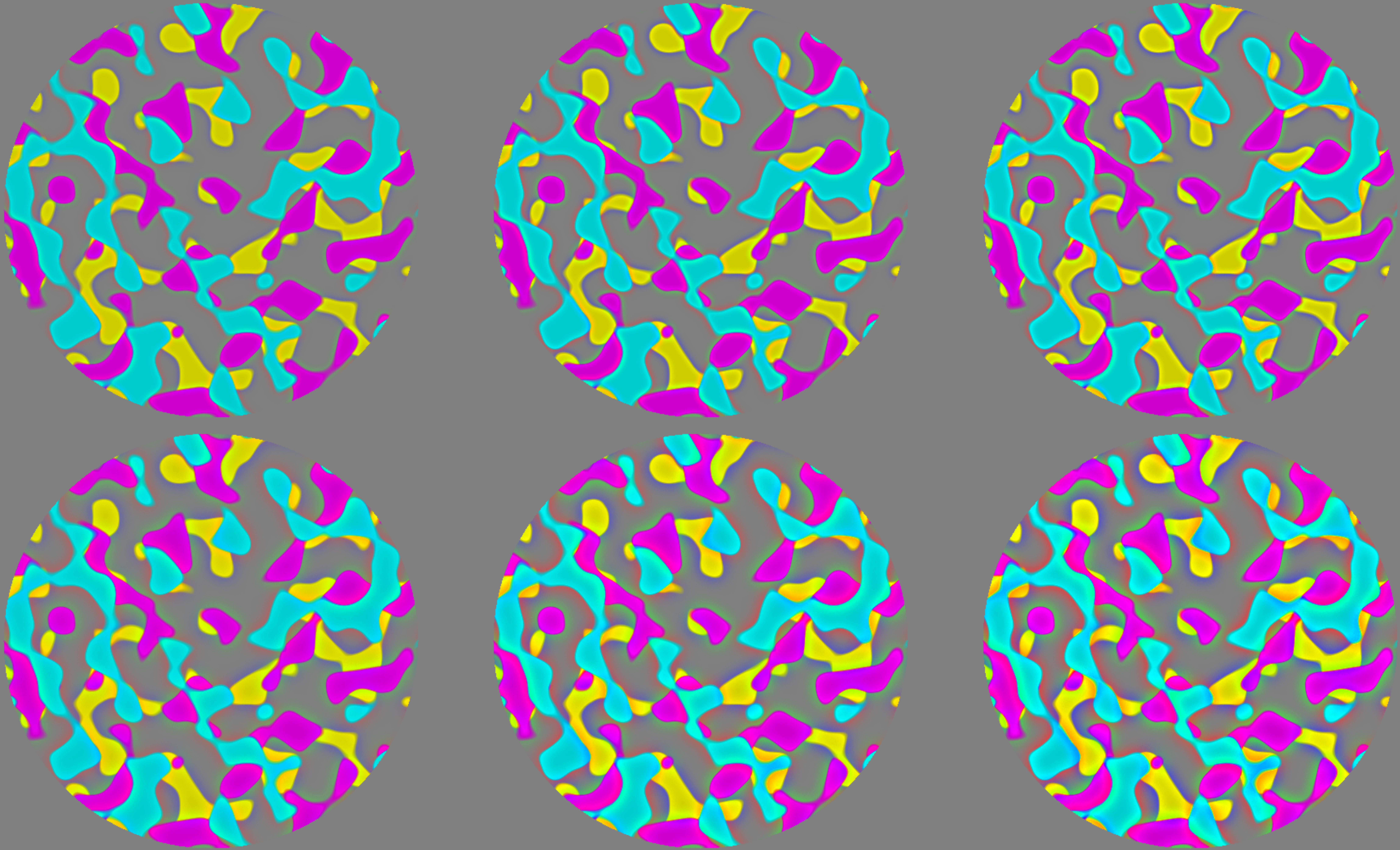


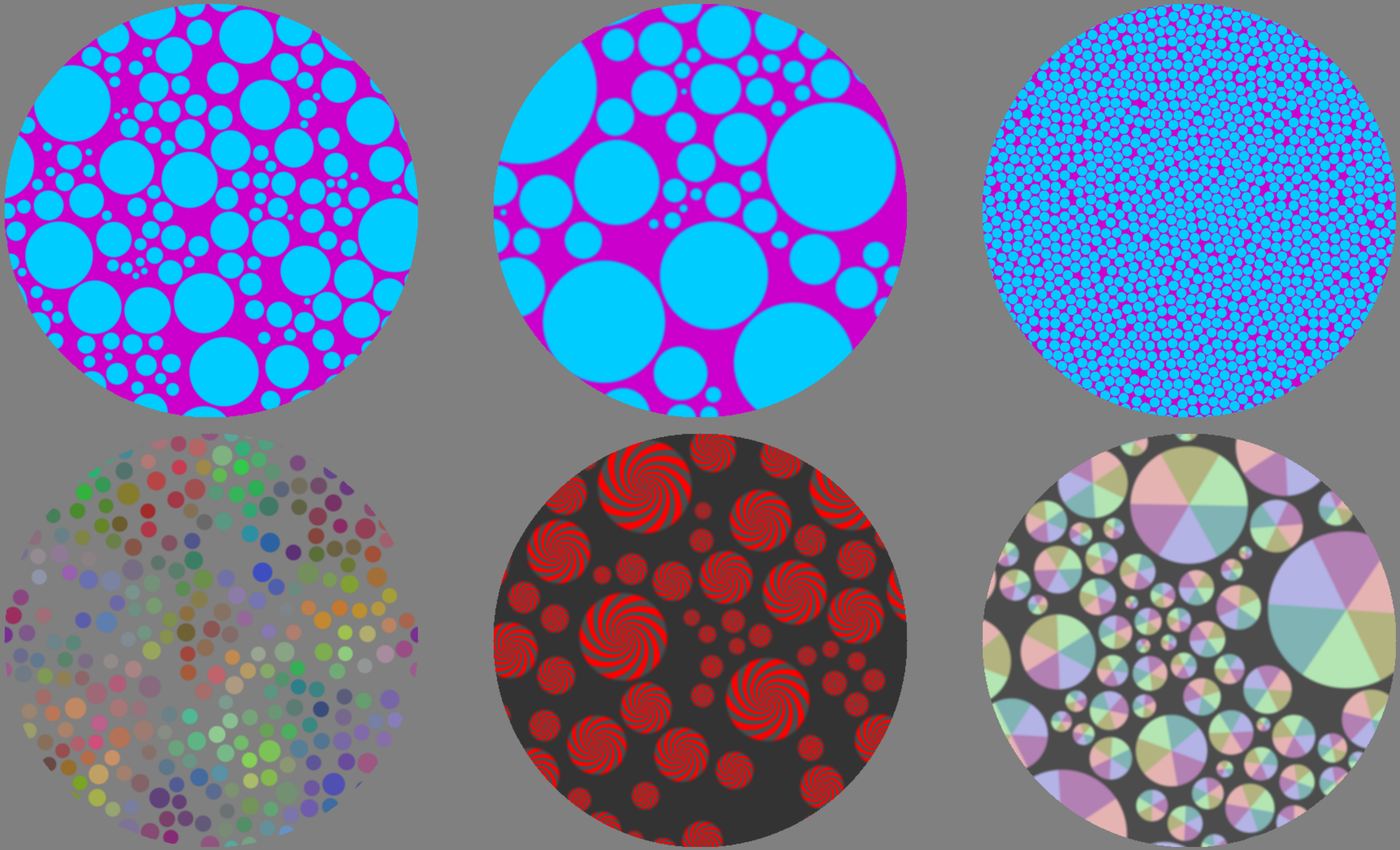


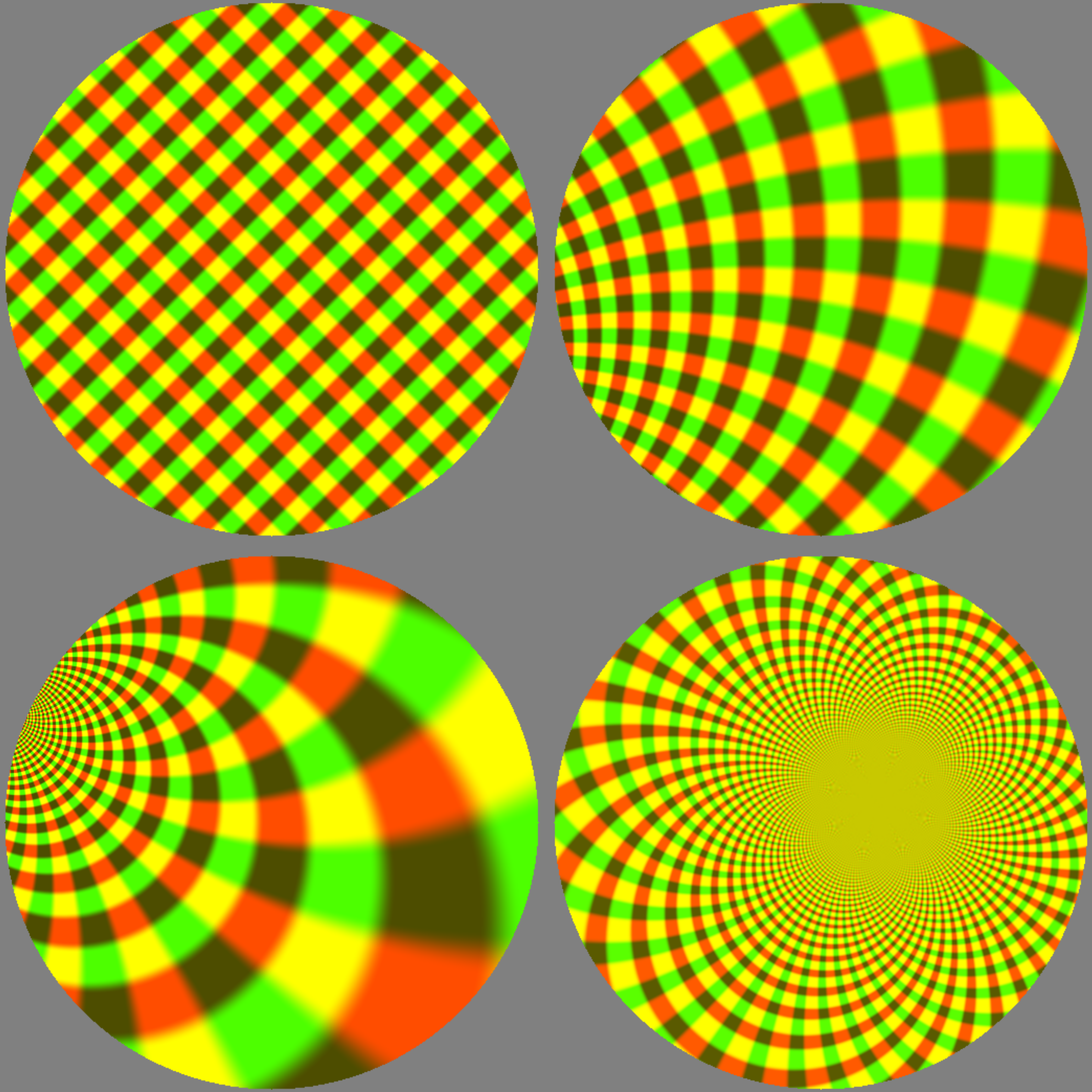



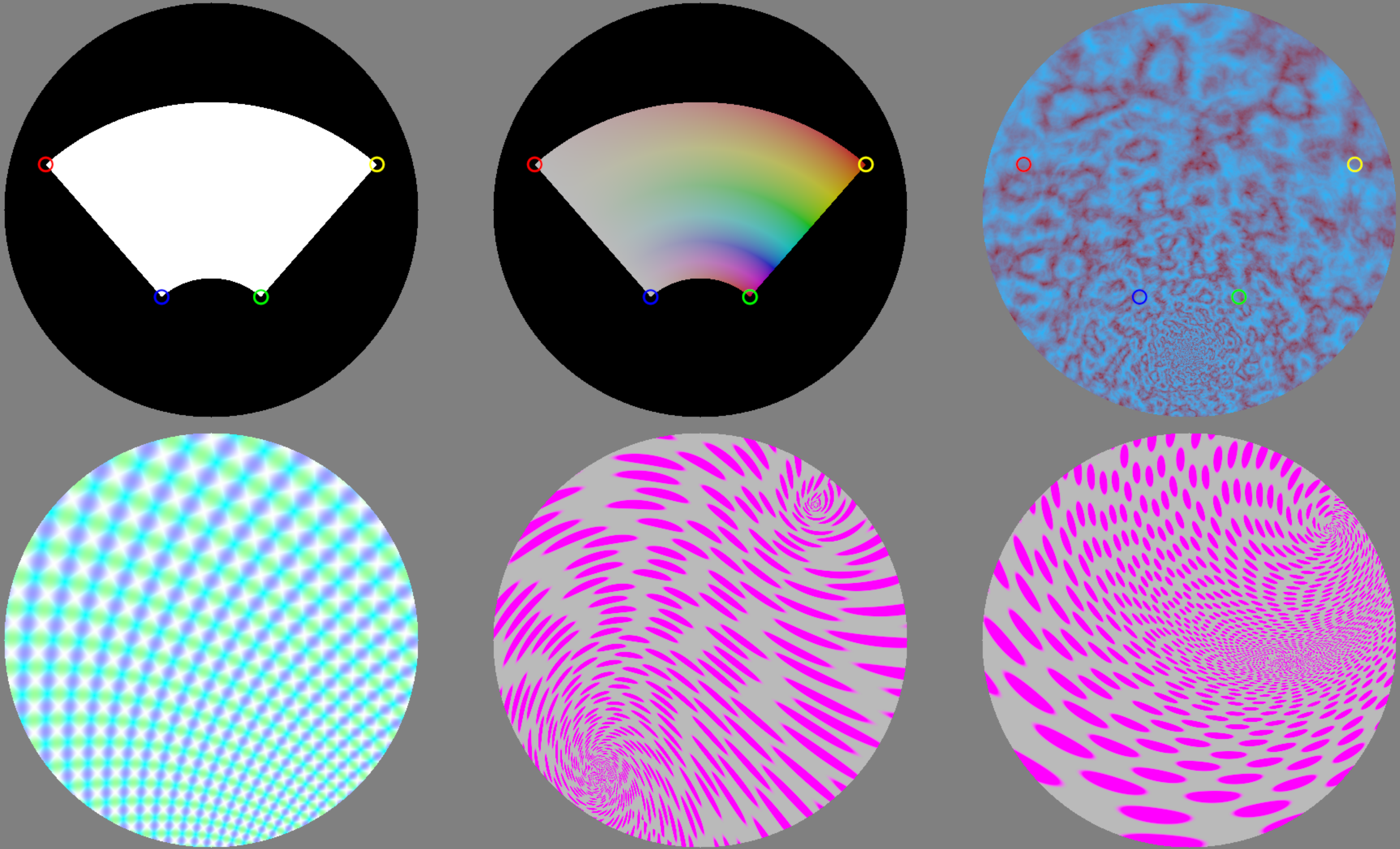


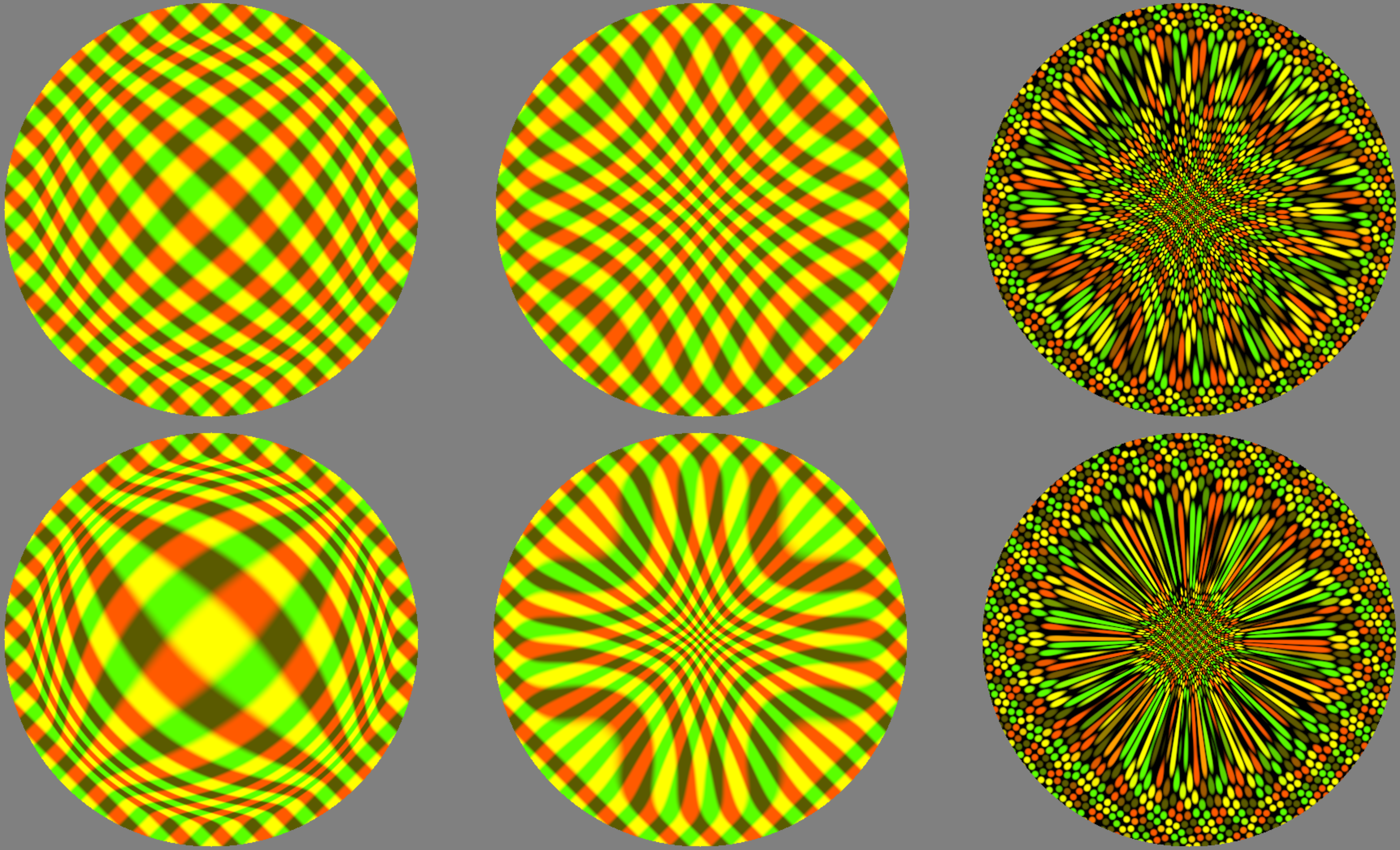


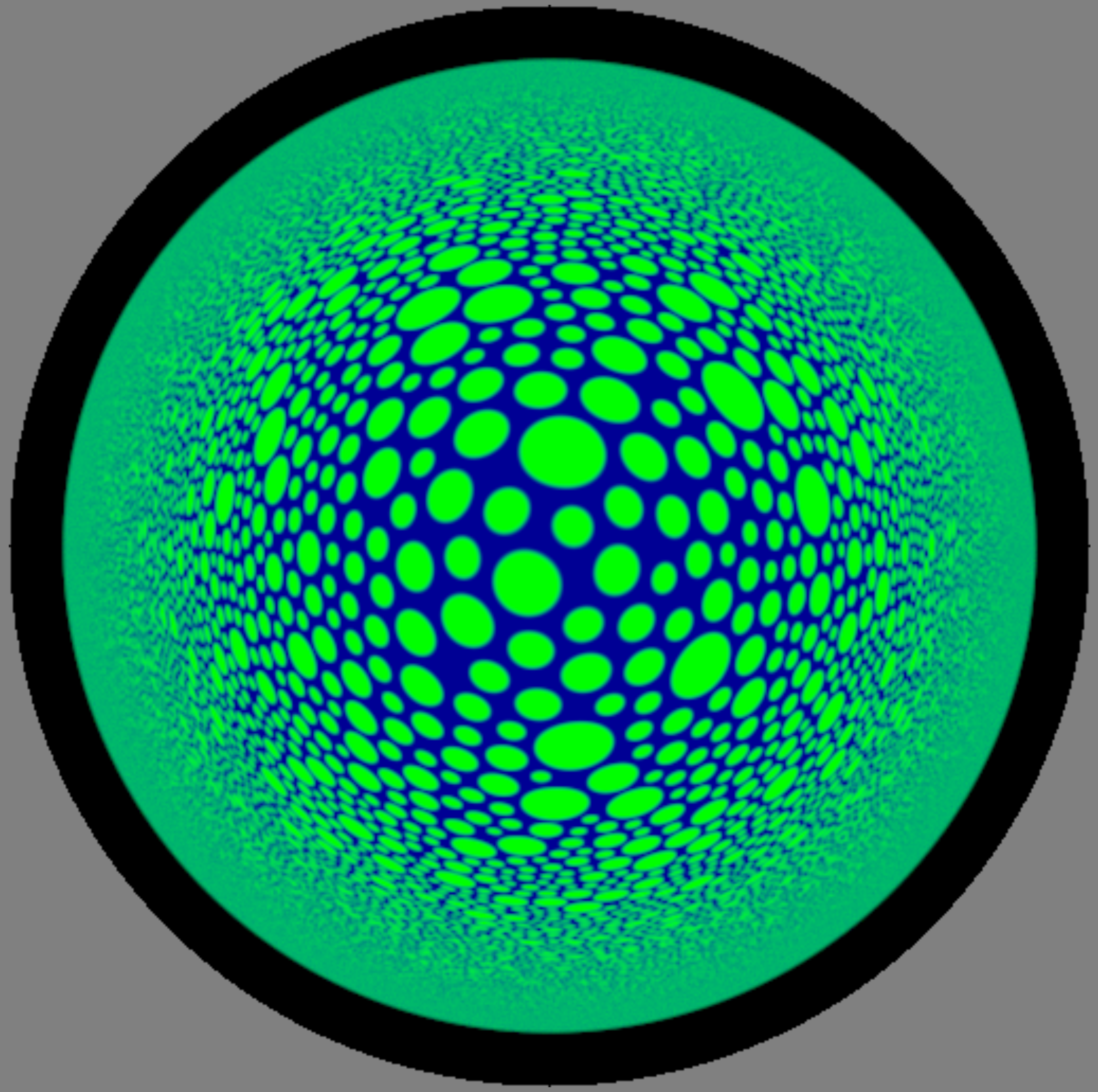
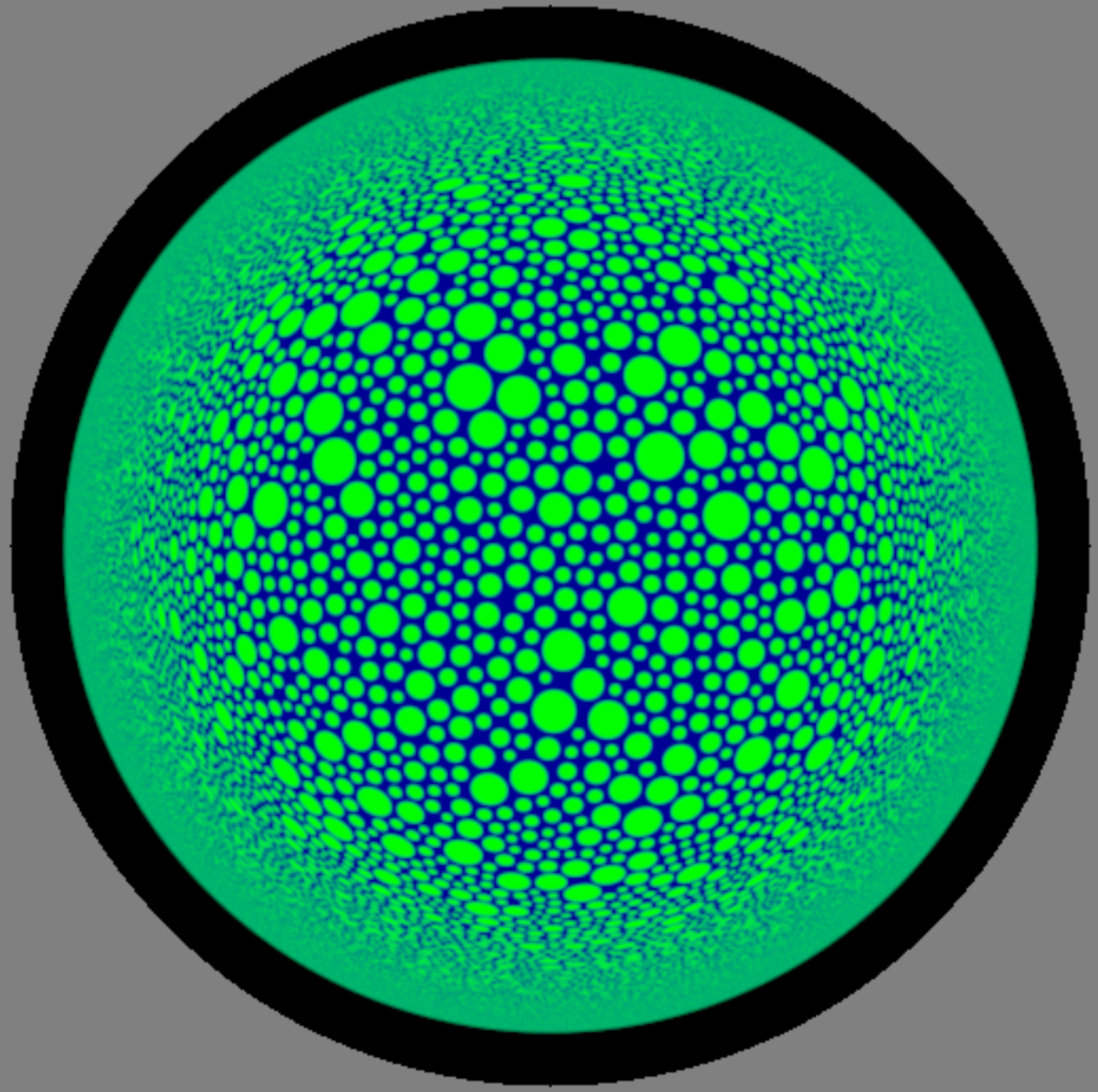
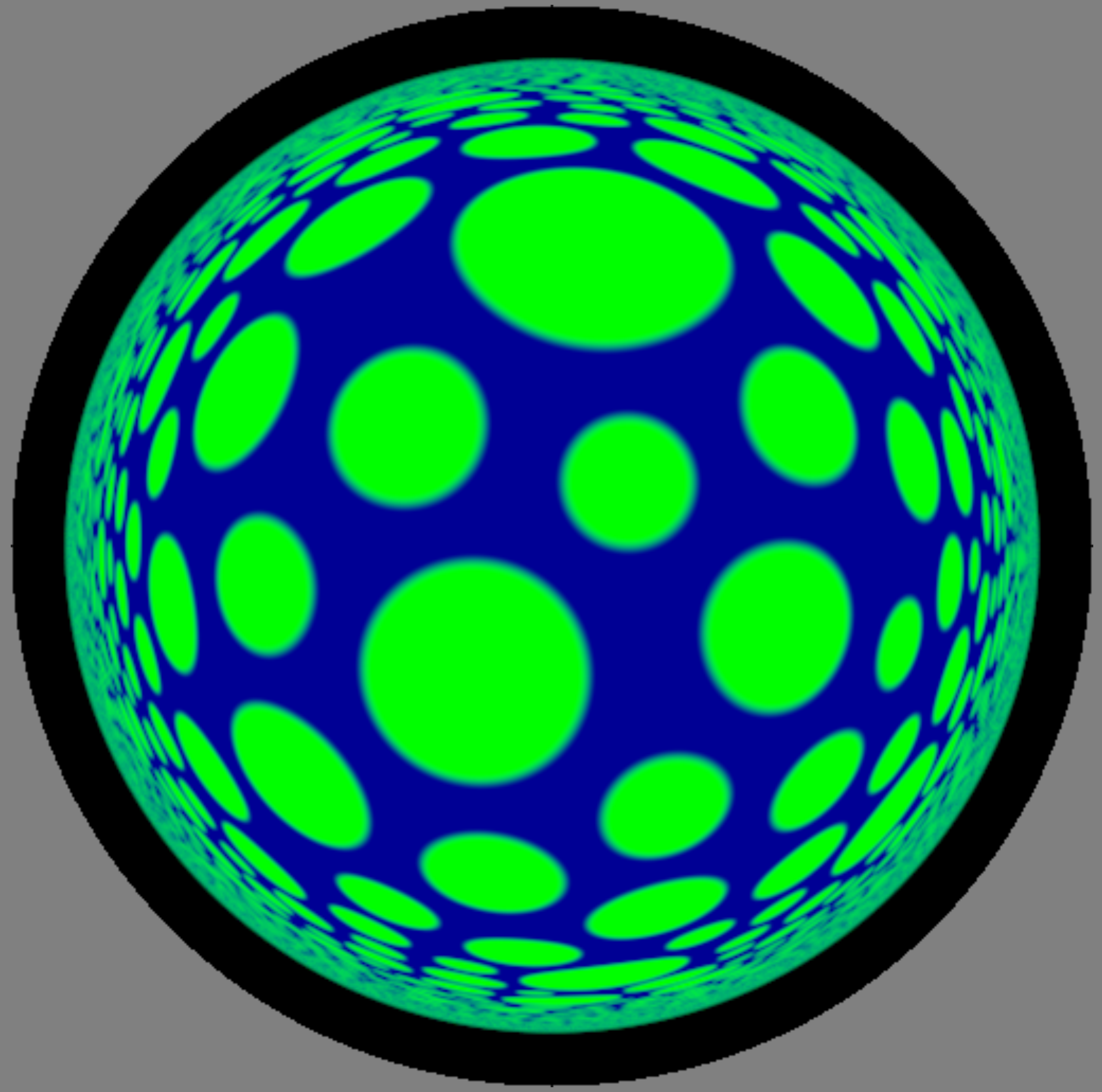
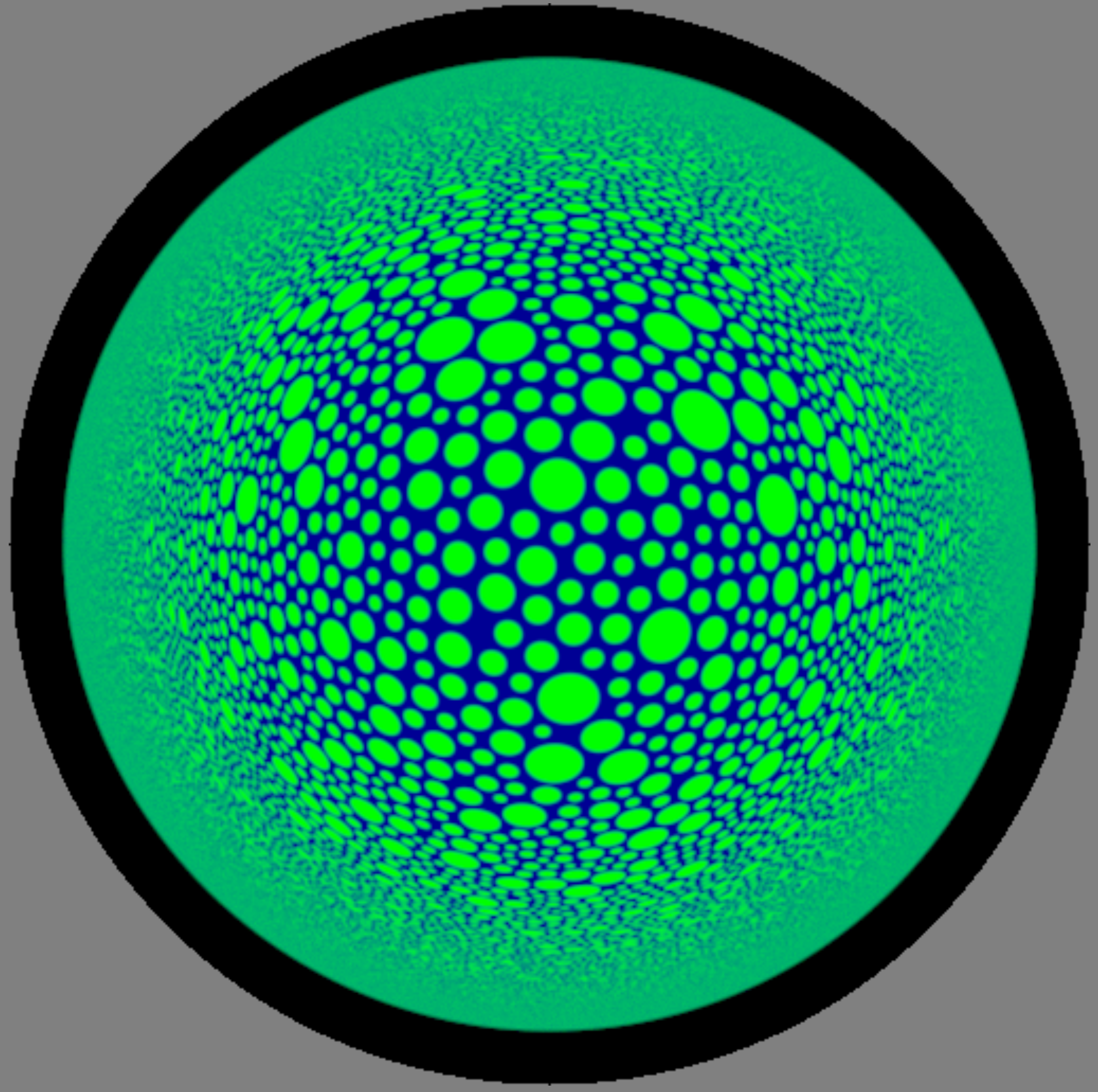
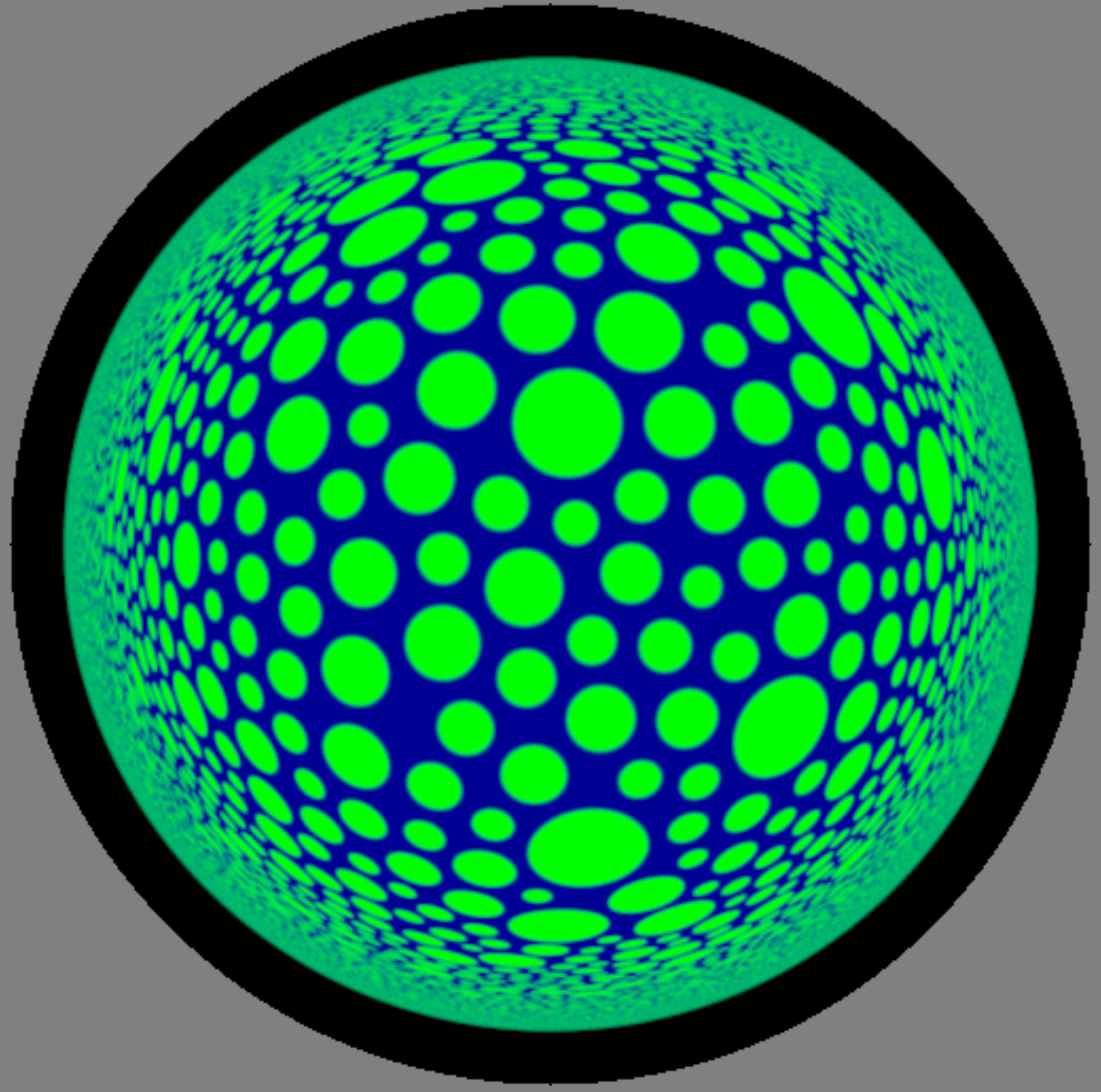










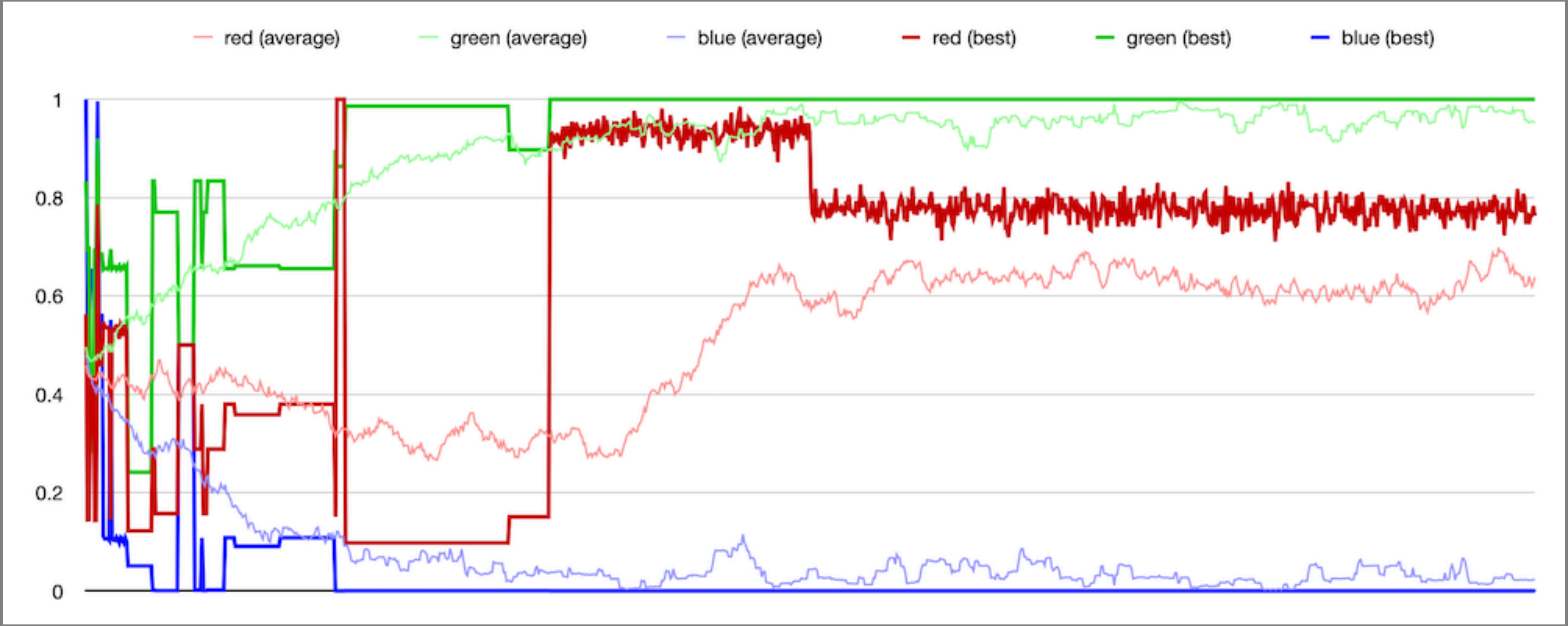
LazyPredator

library for evolutionary computation

LazyPredator

- evolutionary computation
- genetic programming
- steady state population
  - absolute fitness
    - elitism
  - tournament relative fitness
    - negative selection
    - genetic drift

Early simple texture evolution test:
push green up, blue down, red unconstrained.



TexSyn with LazyPredator

Random GP program generation

```
This(Other(9, 0),
     That(This(This(This(Other(8, 2),
                         This(Other(5, 9),
                              Other(3, 8))),
                    Other(3, 1)),
               That(This(This(Other(6, 1),
                              Other(8, 1)),
                         Other(4, 0)),
                    This(Other(3, 2),
                         That(This(Other(9, 4),
                                   Other(6, 9)),
                              This(Other(6, 4),
                                   Other(1, 9)))))),
          Other(9, 6)))
```

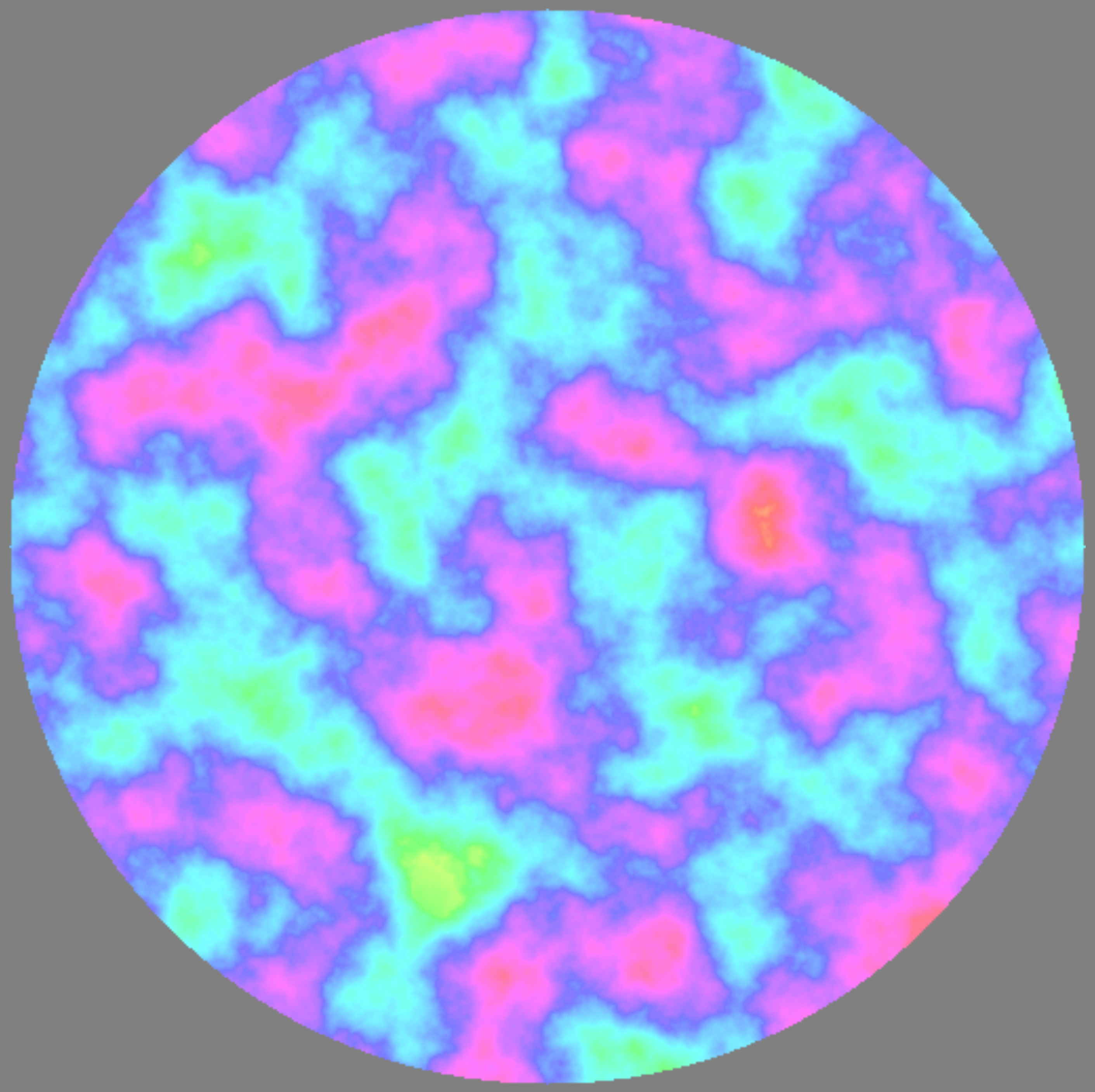
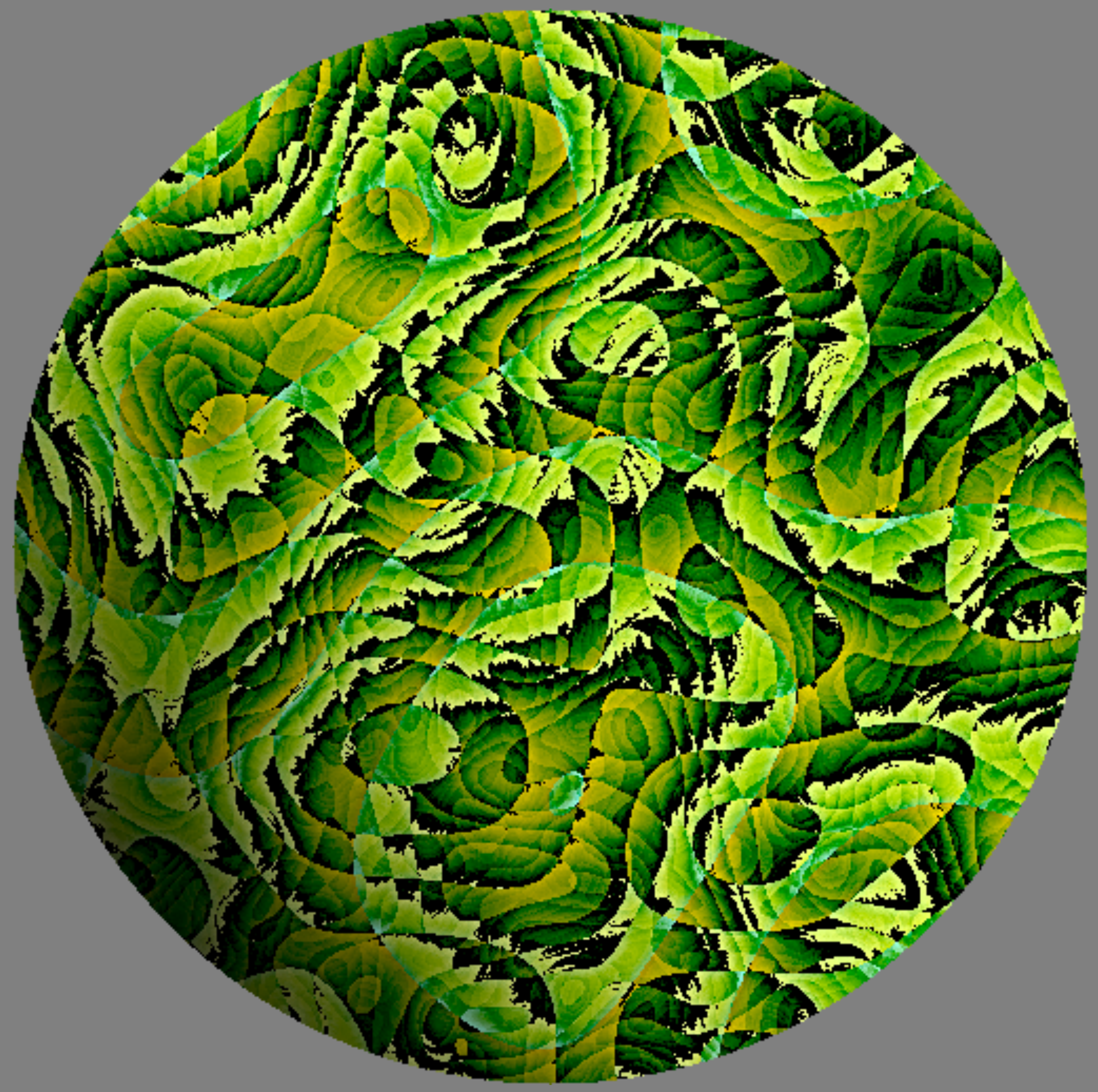
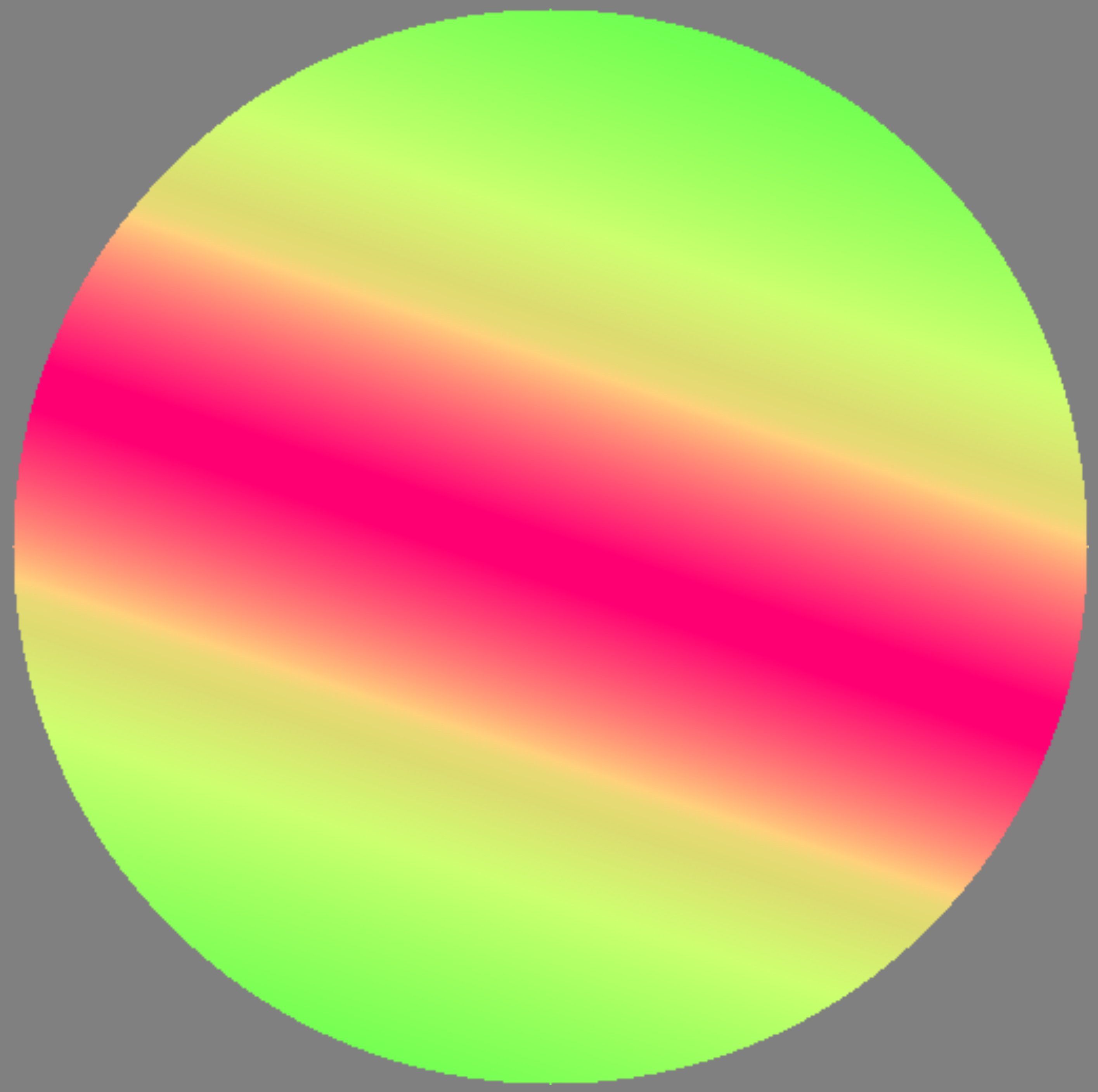
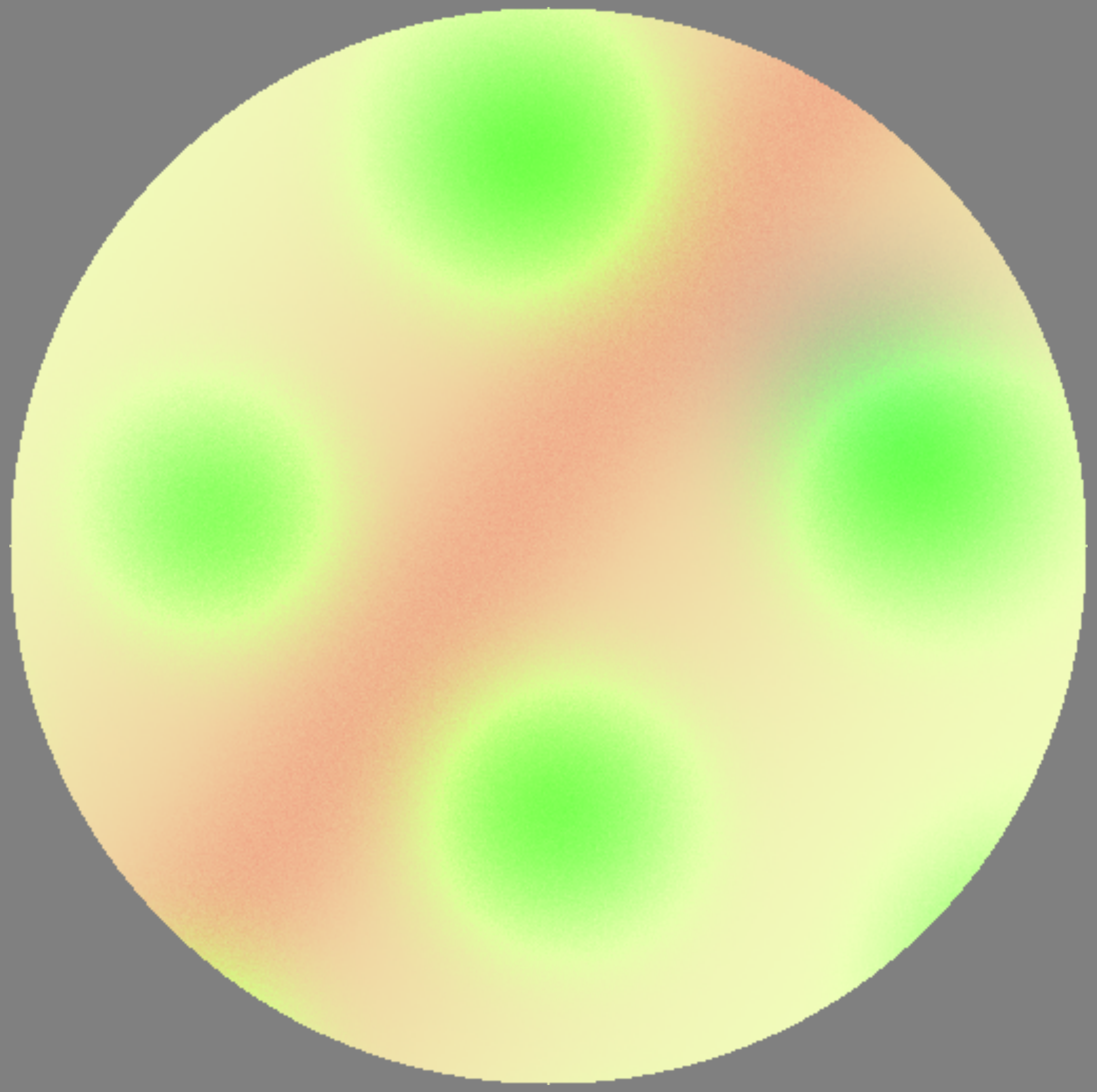
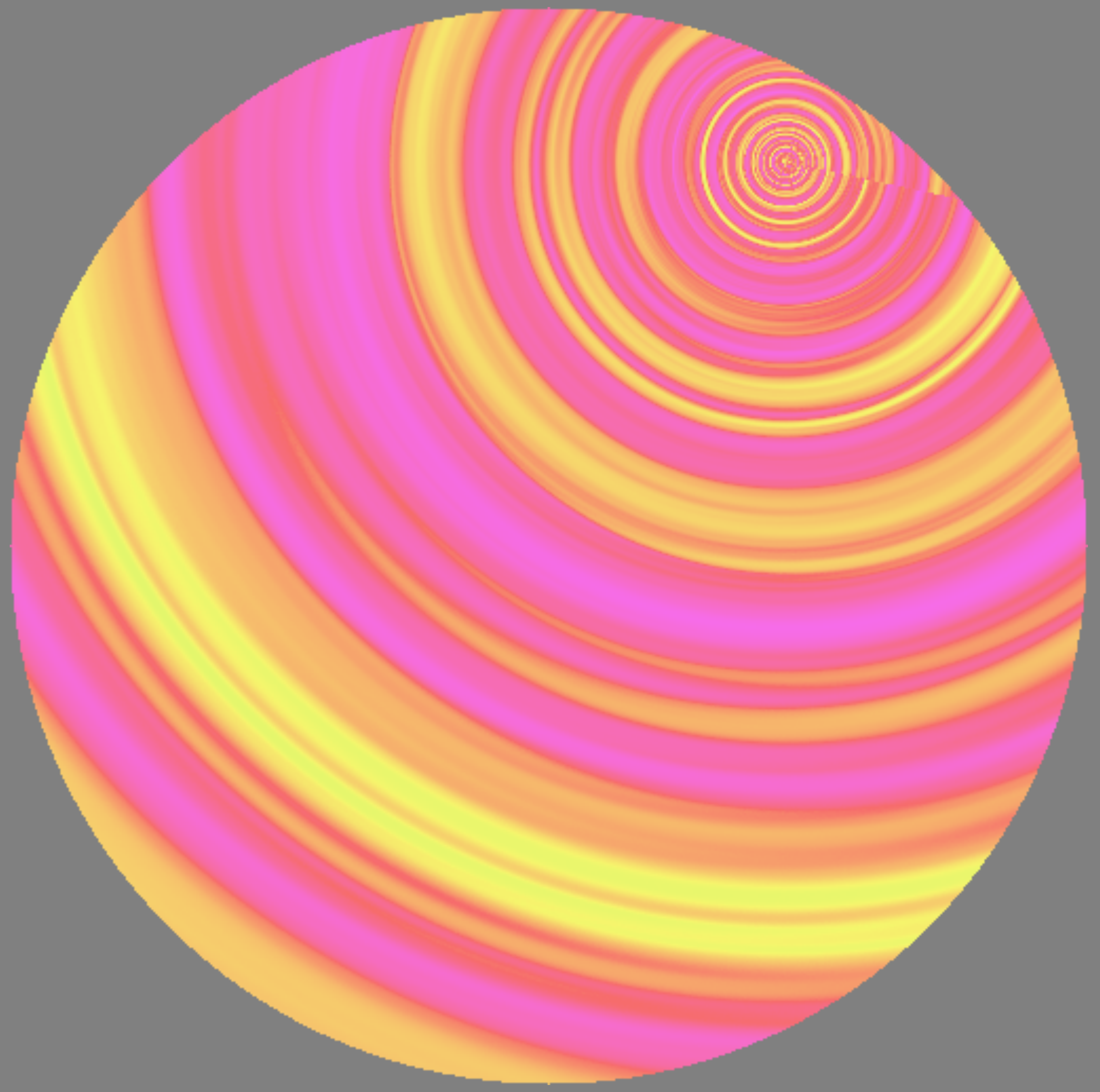
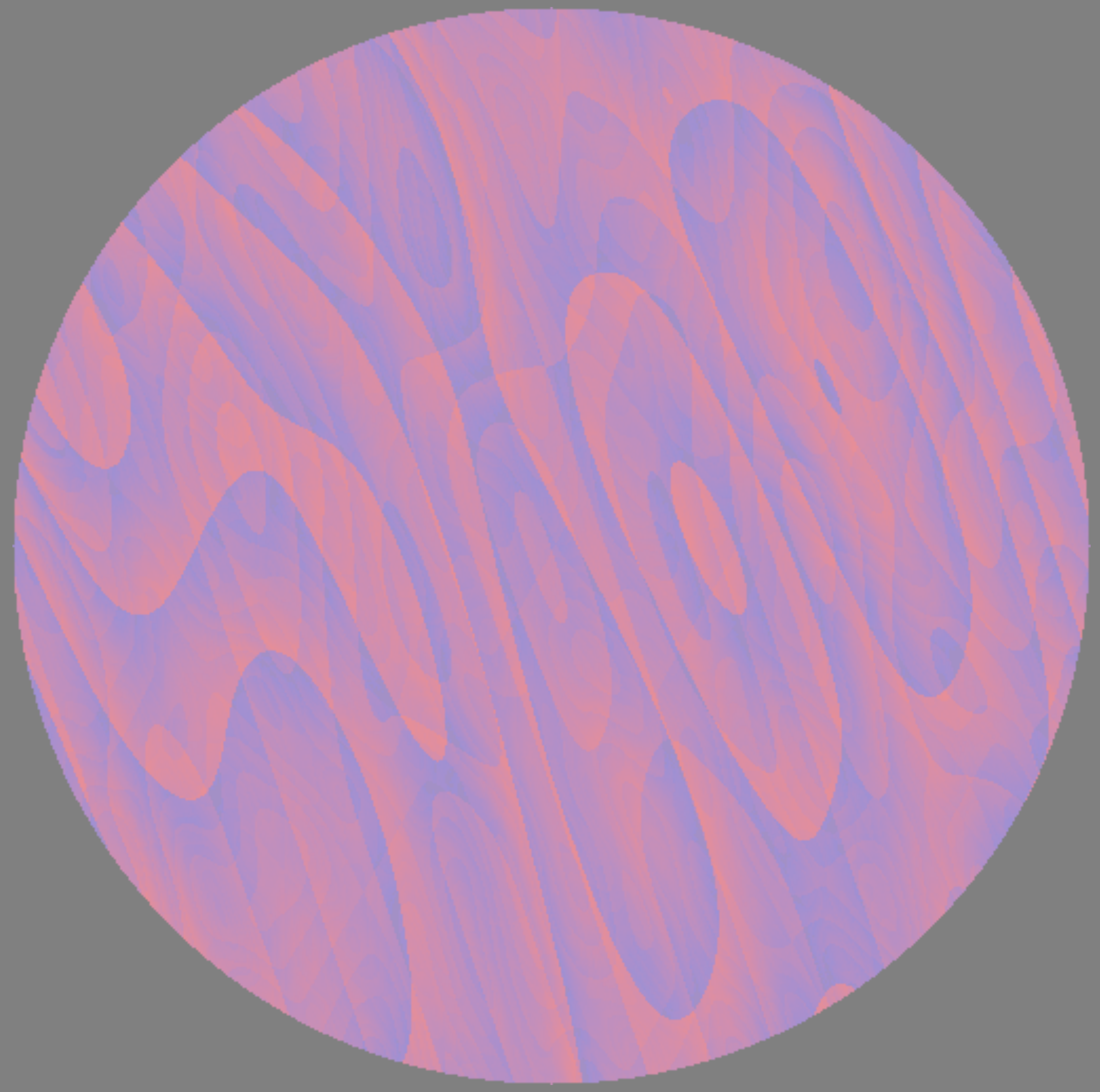
```
Mirror(Vec2(-2.56139, 4.19621), Vec2(2.89003, 3.0138), BrightnessToHue(0.564665, Spot(Vec2(-4.74602, -0.973344),
    1.37896, Max(SoftMatte(Uniform(0.615928, 0.28928, 0.571328), Uniform(0.282334, 0.51935, 0.754801),
    Uniform(0.100931, 0.370136, 0.269515)), Gamma(3.73964, Uniform(0.205834, 0.869023, 0.862499))), 1.86182,
    MultiNoise(Vec2(-1.43561, -4.09413), Vec2(-0.894367, -0.546391), AdjustBrightness(0.197535, Uniform(0.619363,
    0.851894, 0.816023)), AdjustSaturation(0.849886, Uniform(0.930344, 0.951282, 0.177565)), 0.0469612))))
```

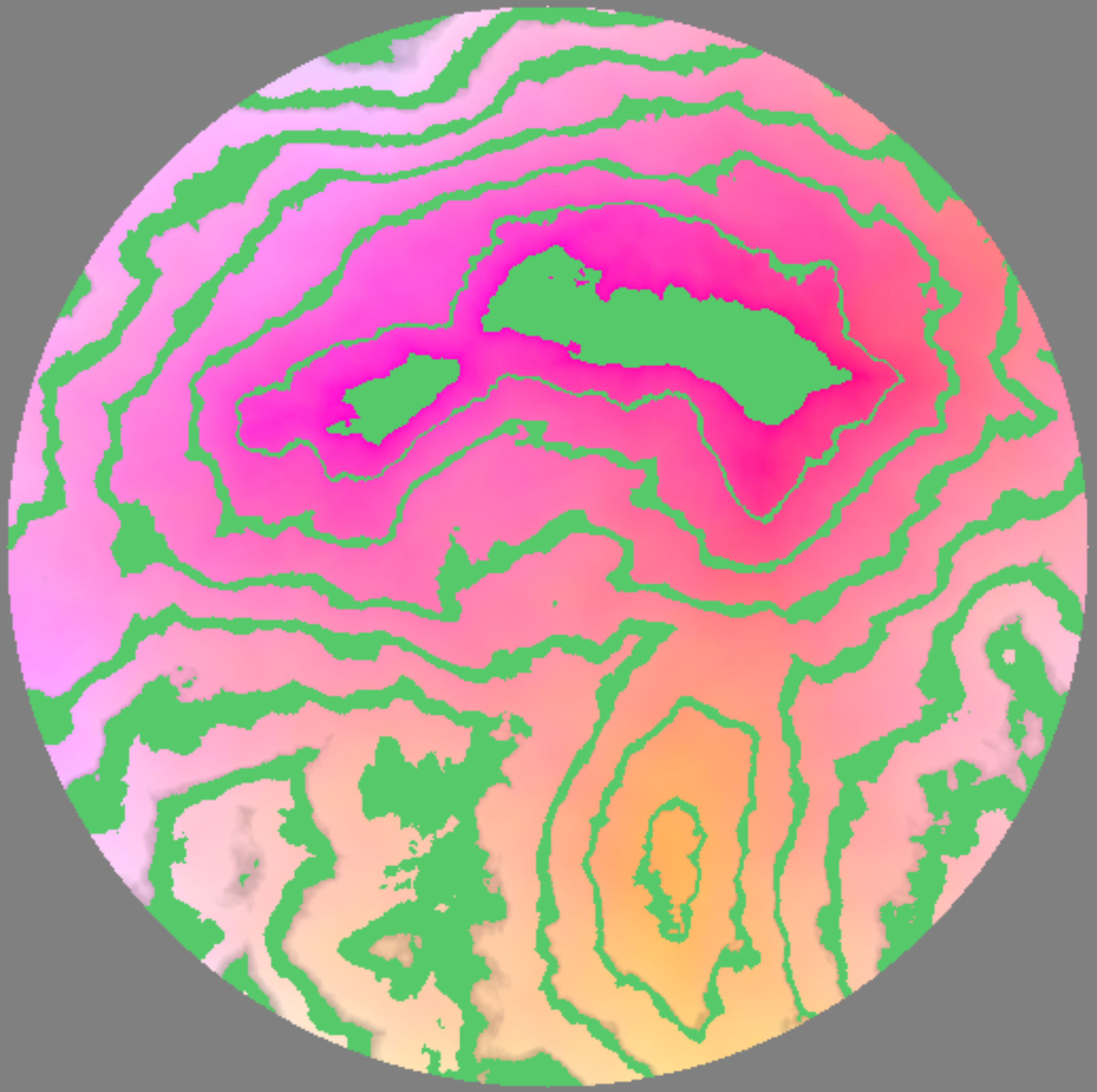
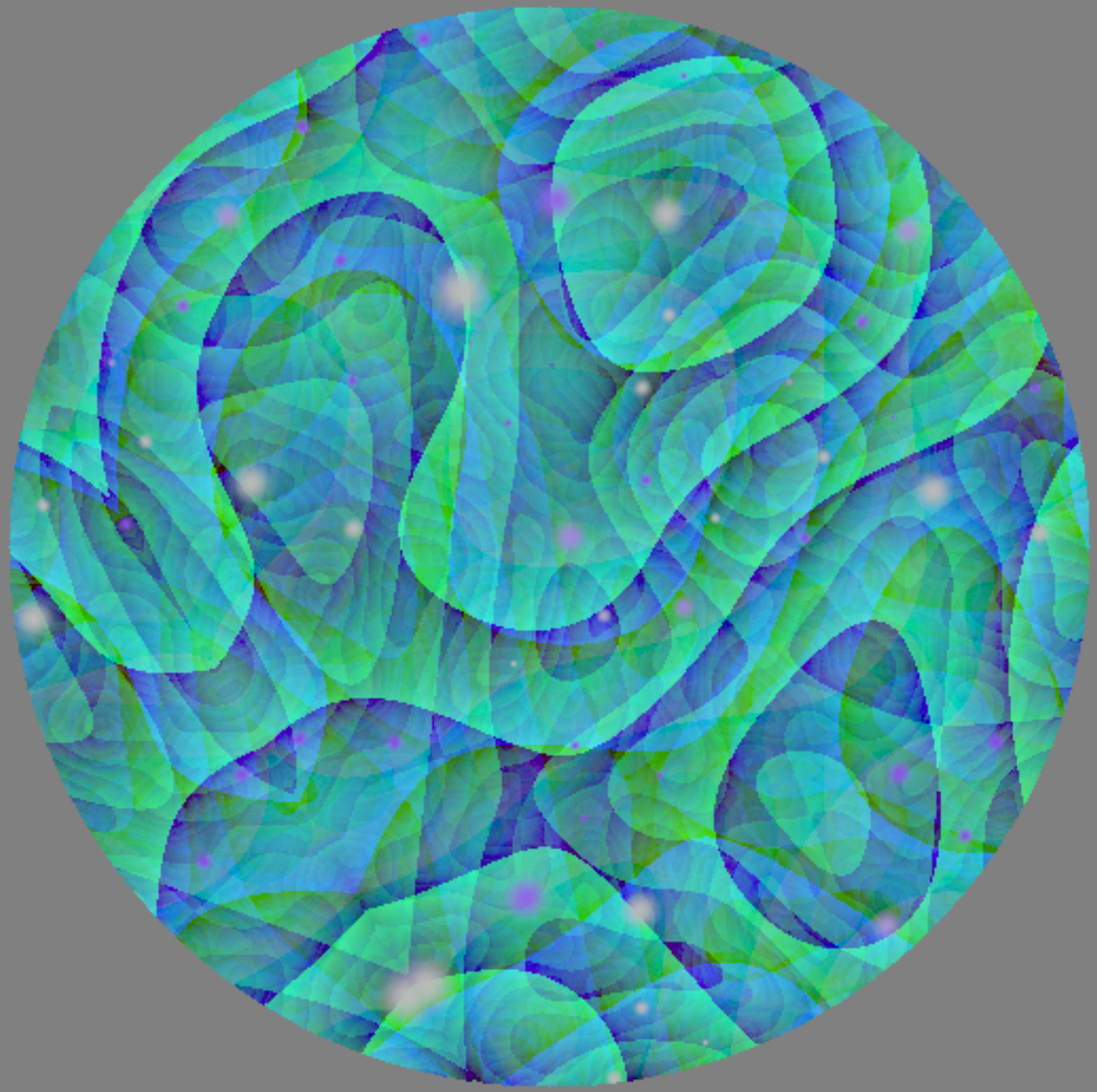
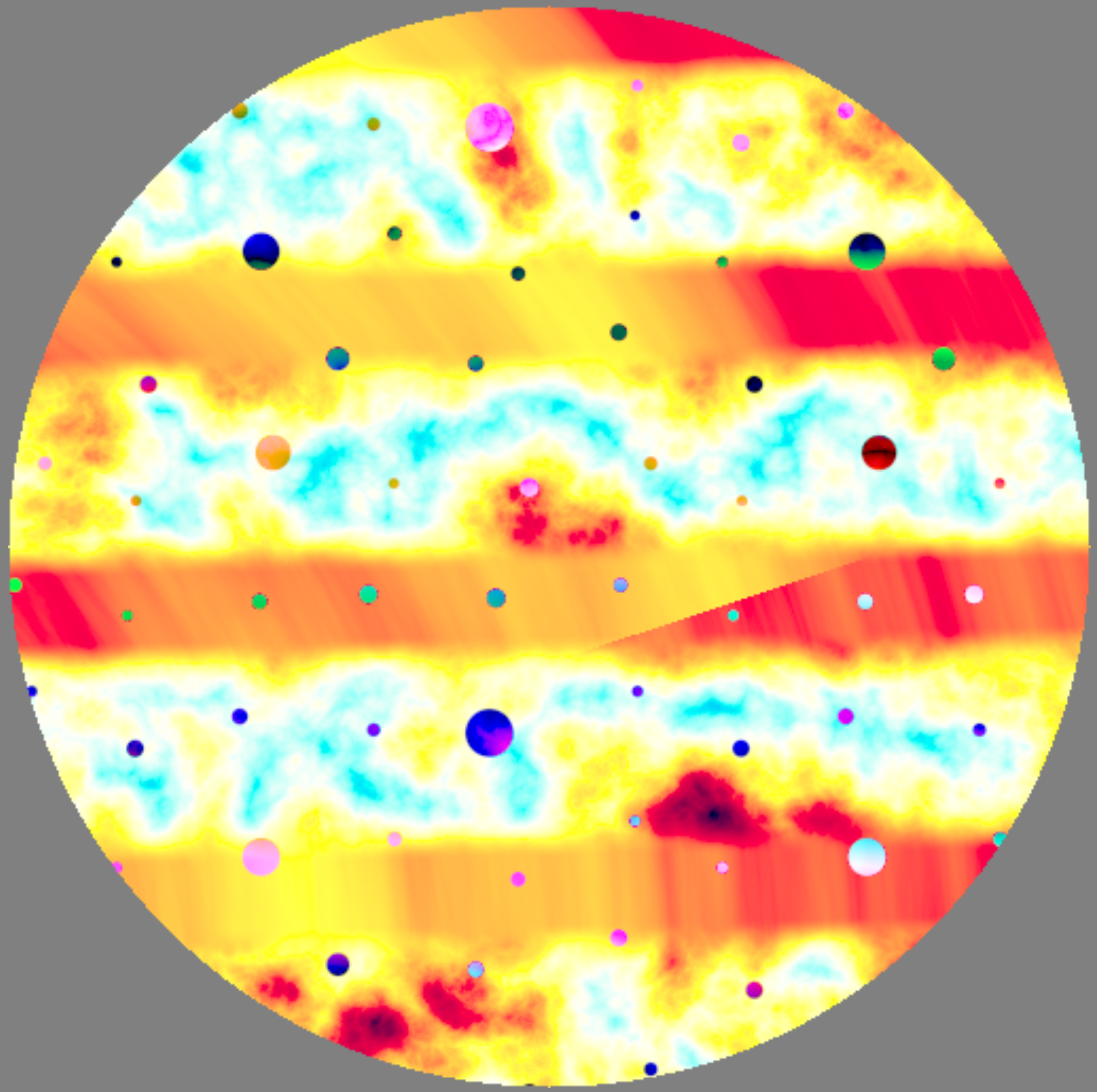
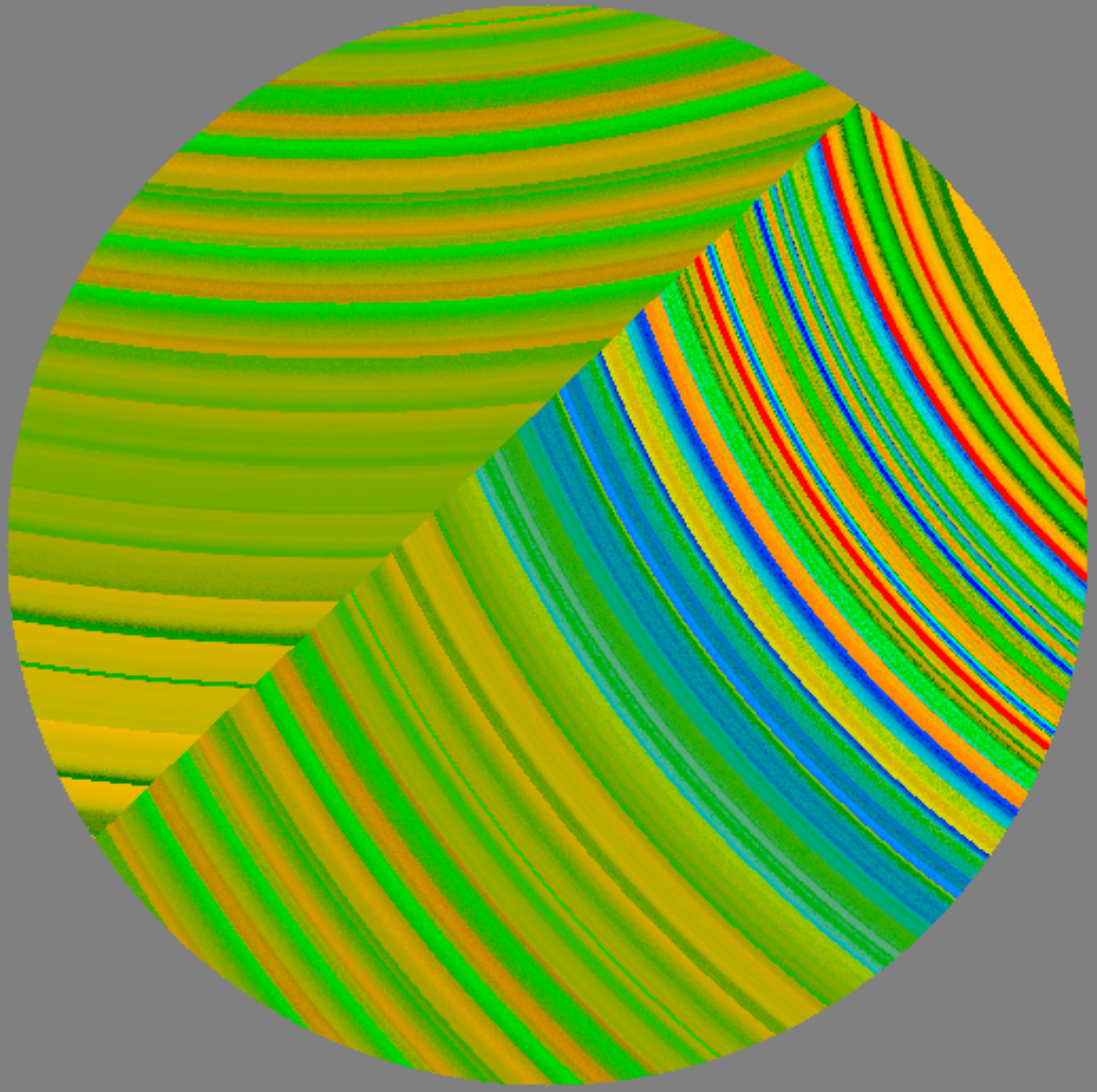
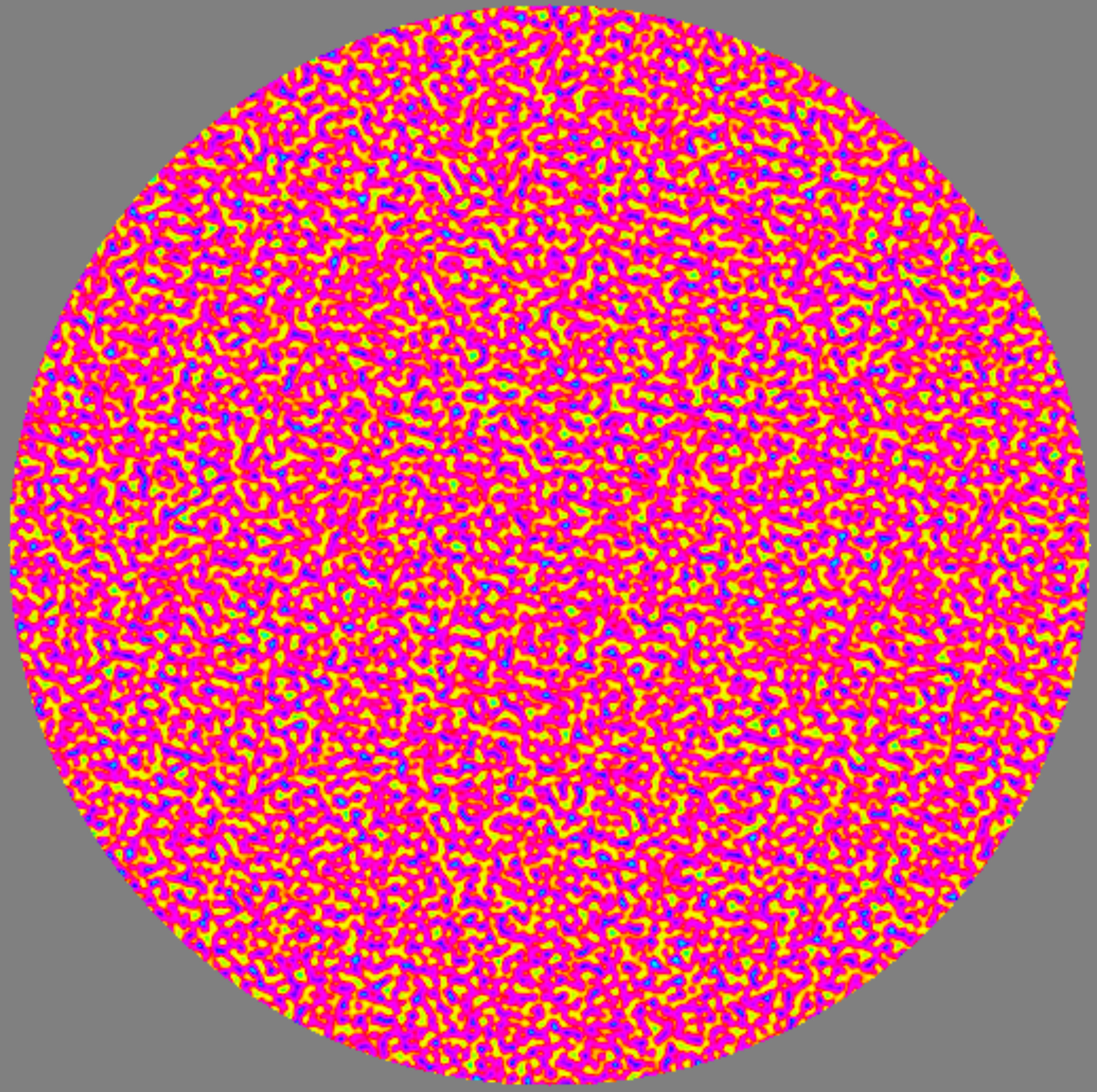
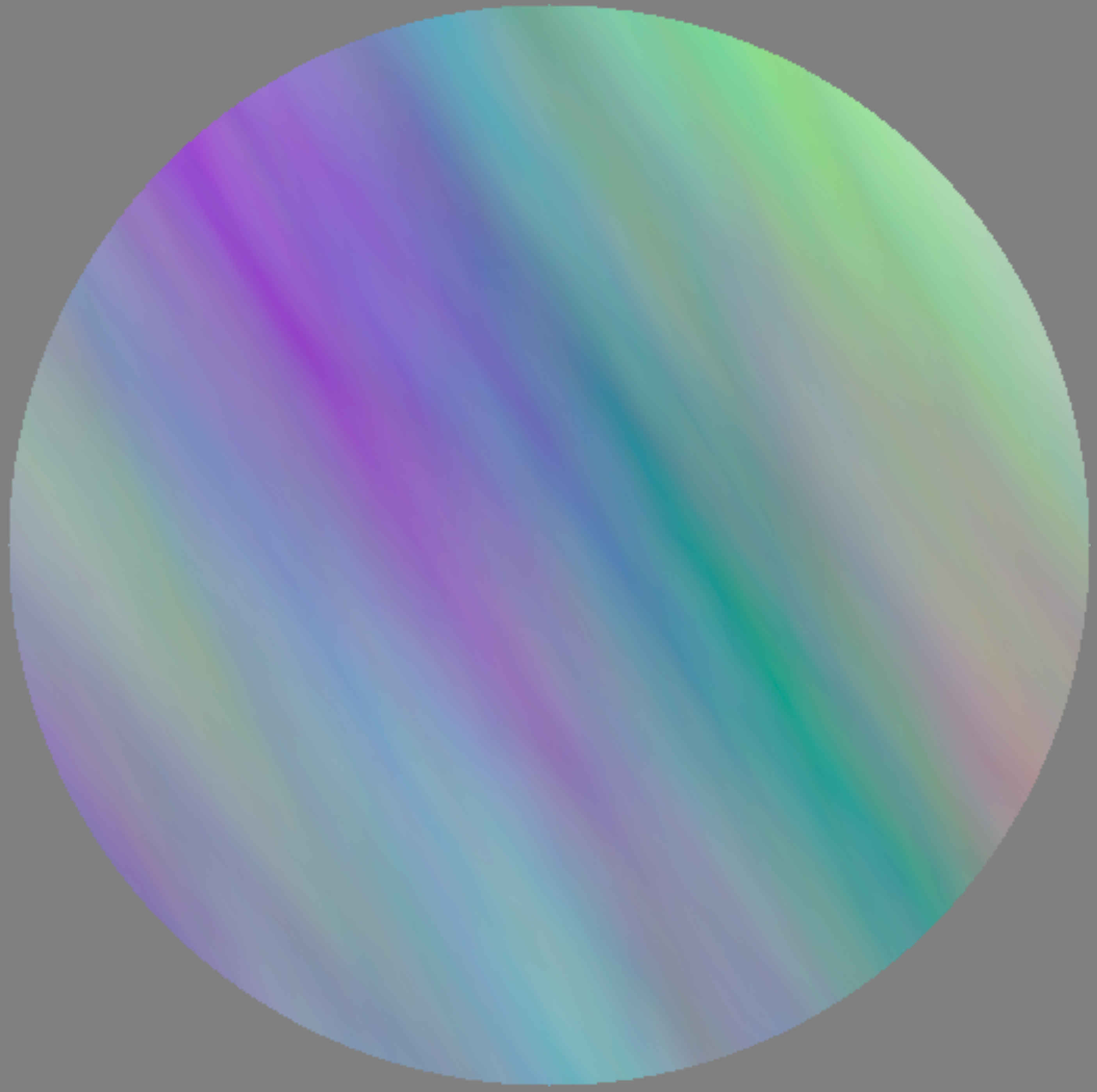
Random GP program crossover

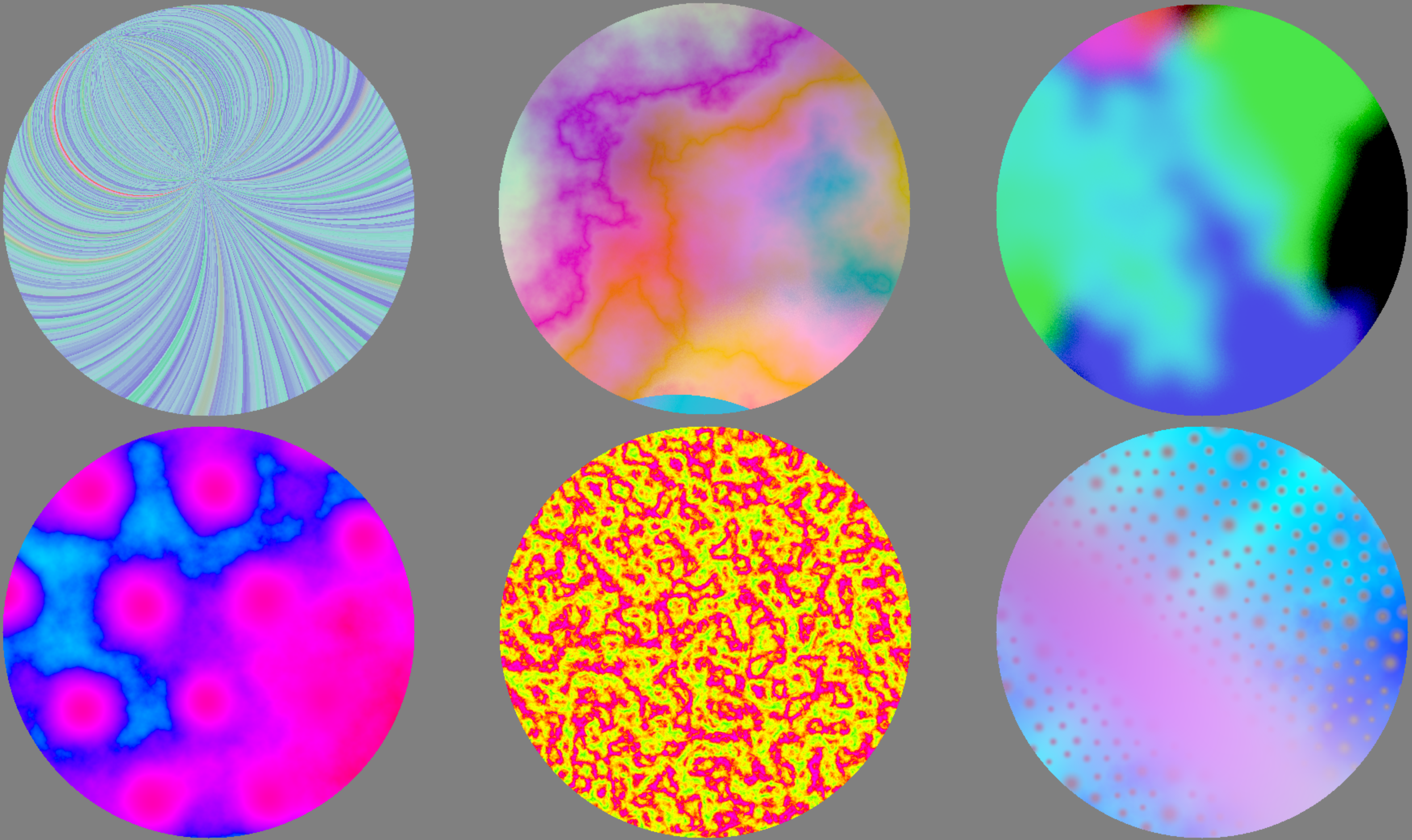
```
// The P tree with it first subtree selected:
PPP(PPP(P(7), P(P(1)), PP(4, 3)),
    P(PPP(P(8), P(P(0)), PP(8, 4))),
    PPP(P(P(1)), P(P(6)), P(P(7))))

// The Q tree with it third subtree selected:
QQQ(Q(QQQ(Q(7), Q(2), QQ(5, 8))),
    Q(QQQ(Q(5), QQ(5, 3), Q(Q(3)))),
    QQQ(QQ(3, 4), QQ(6, 1), QQ(8, 1)))

// The offspring tree with some of both:
PPP(QQQ(QQ(3, 4), QQ(6, 1), QQ(8, 1)),
    P(PPP(P(8), P(P(0)), PP(8, 4))),
    PPP(P(P(1)), P(P(6)), P(P(7))))
```

LimitHue test: absolute fitness

- Population size: 100
- Evolution steps: 1000
  - equivalent to 10 “generations”
- Initial tree size: 100
- Multiplicative components:
  - 4 of 12 “hue buckets”
  - average brightness near midrange
  - average saturation above 50%



Thank You!

these slides:

http://www.red3d.com/cwr/presentations/20201118_TexSyn_LazyPredator.pdf

git repositories:

https://github.com/cwreynolds/TexSyn

https://github.com/cwreynolds/LazyPredator