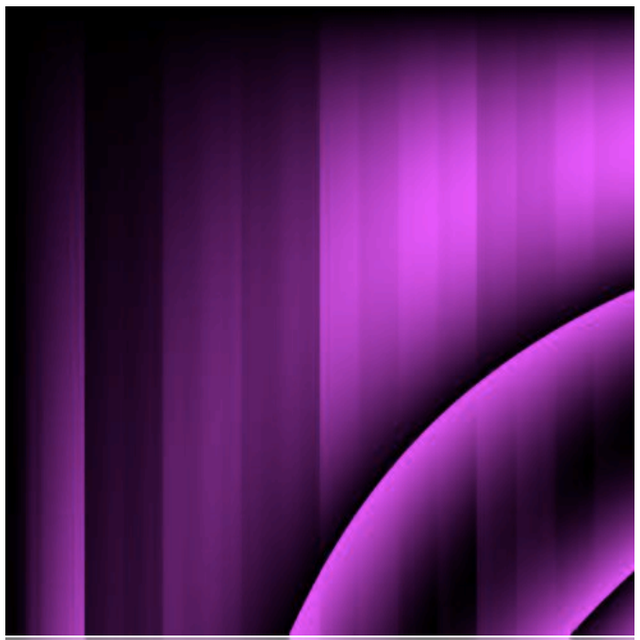
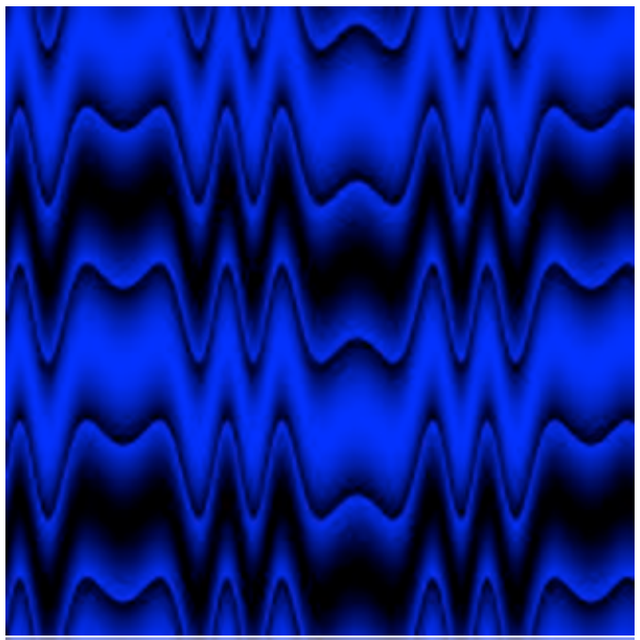
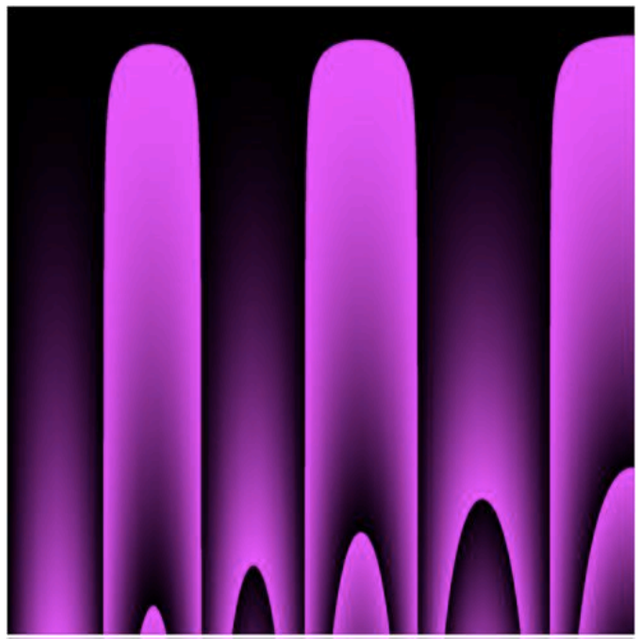
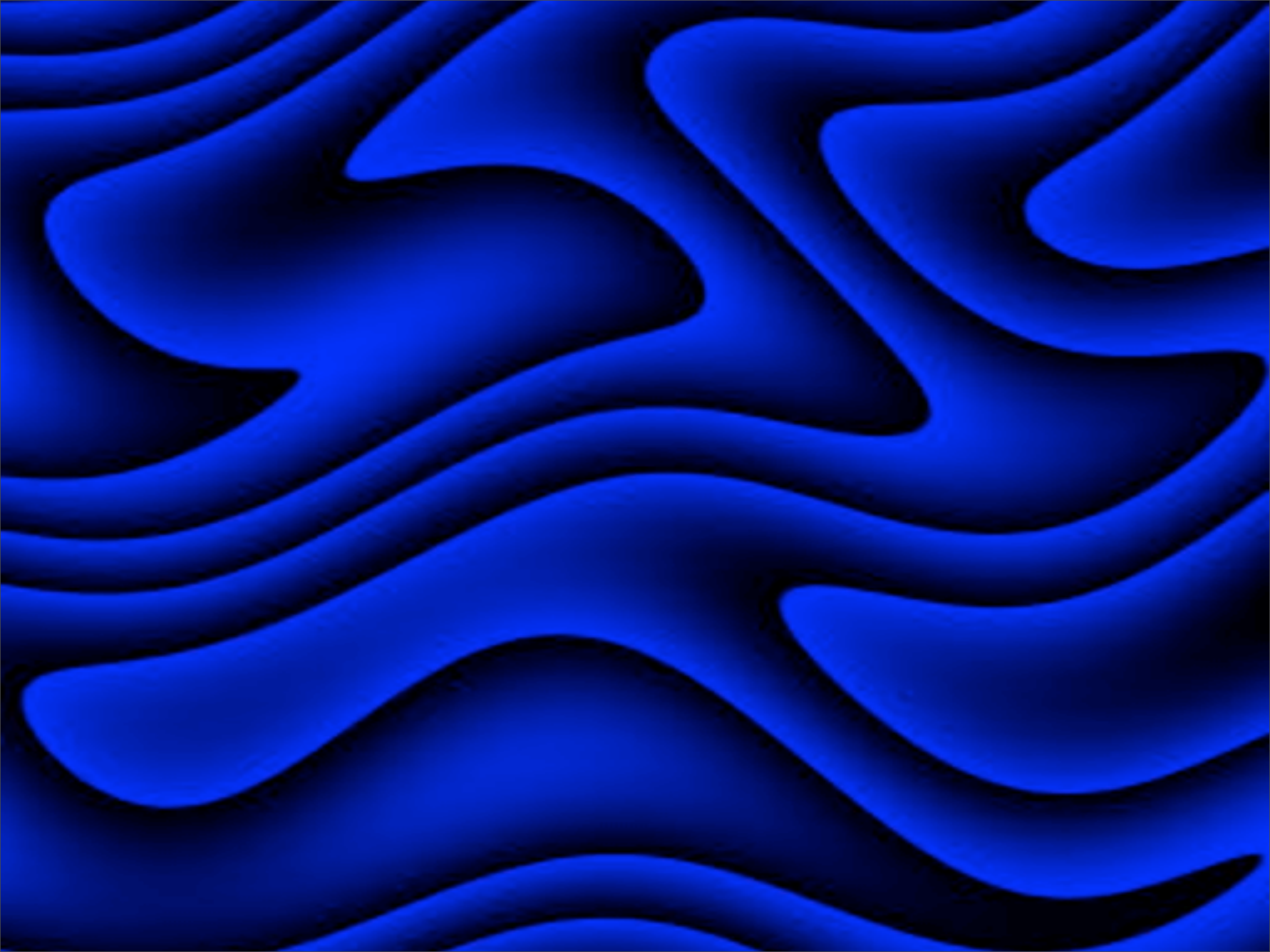
Artwork Evolution

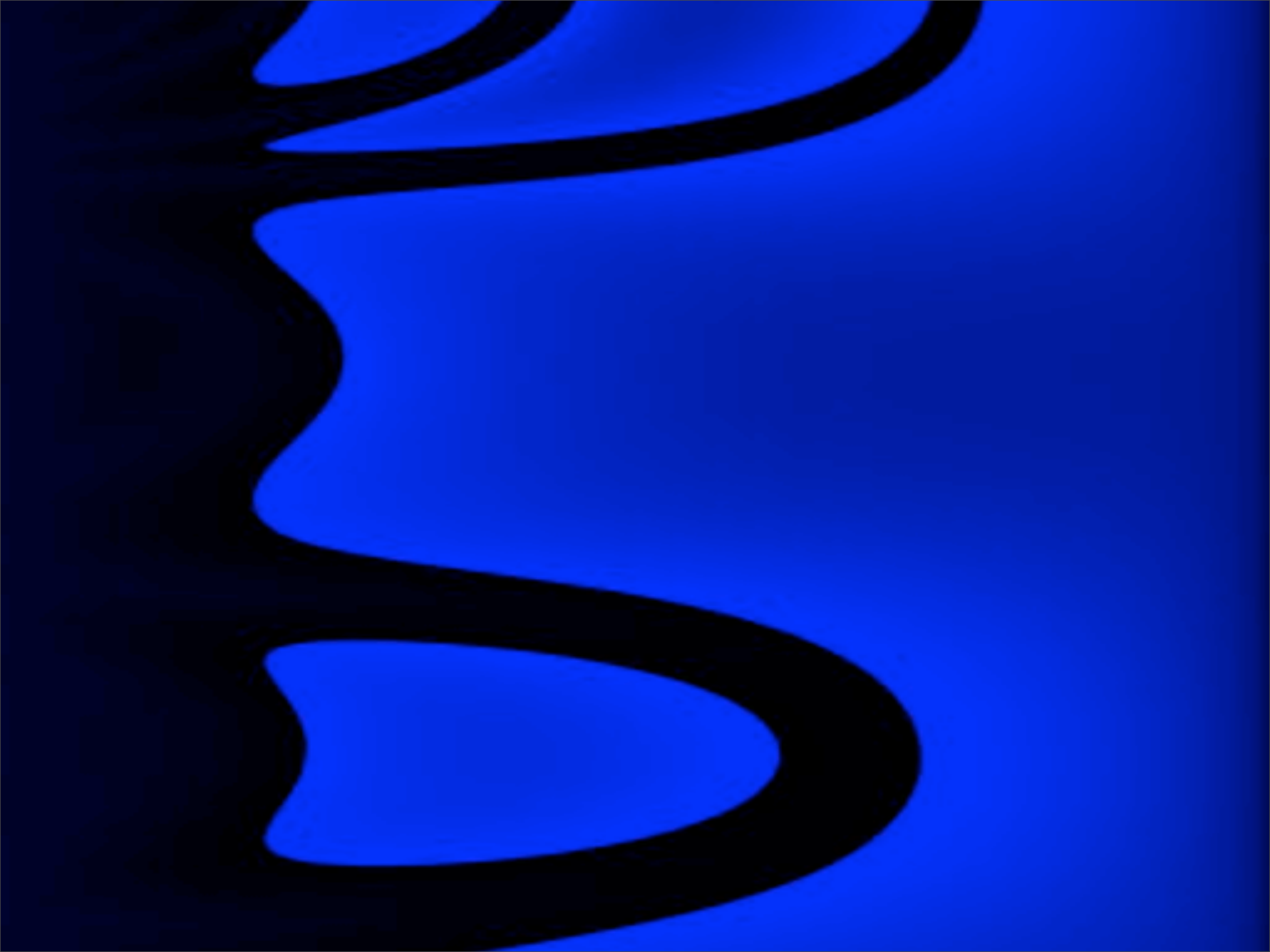
Paul Solt

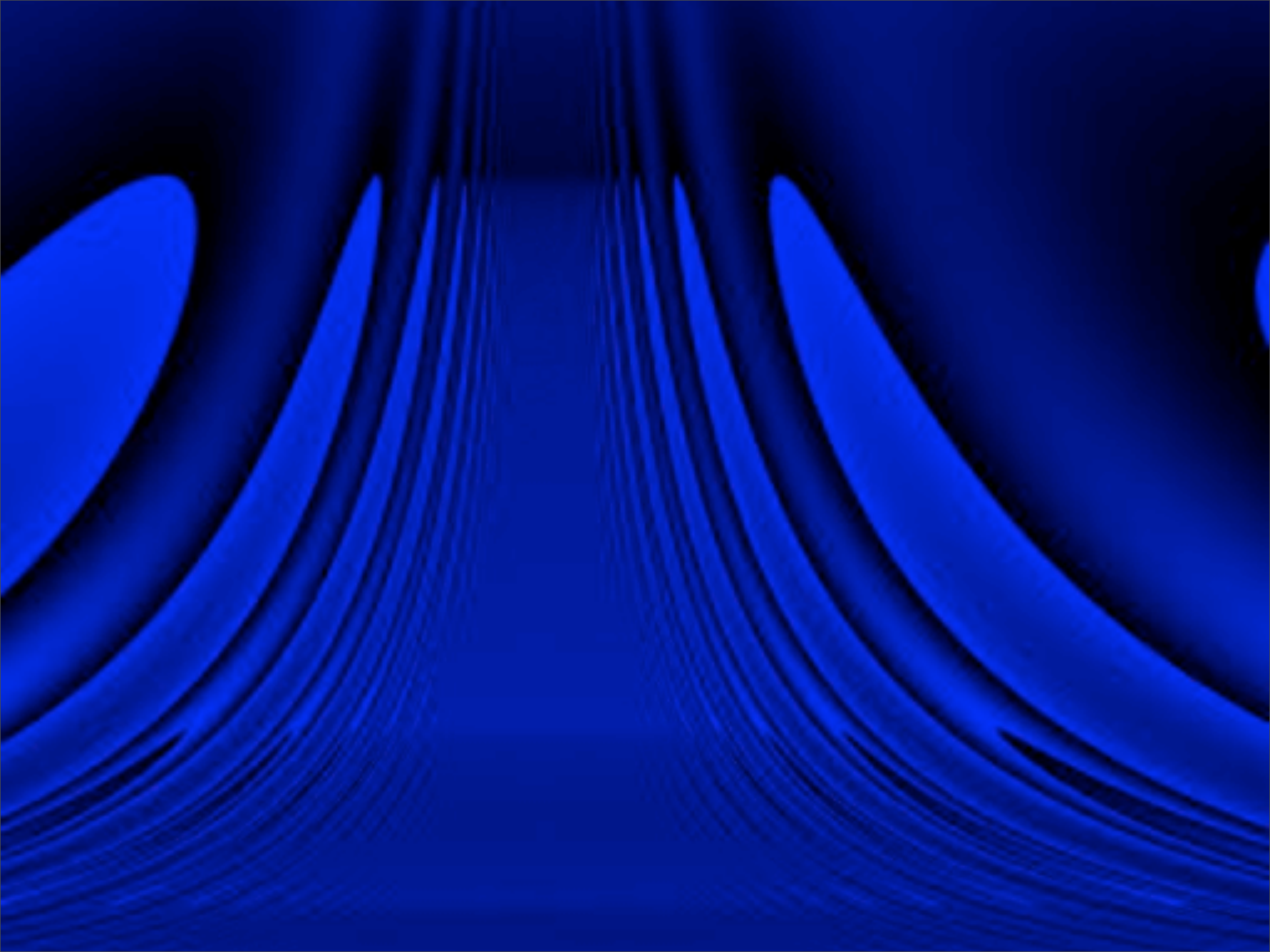






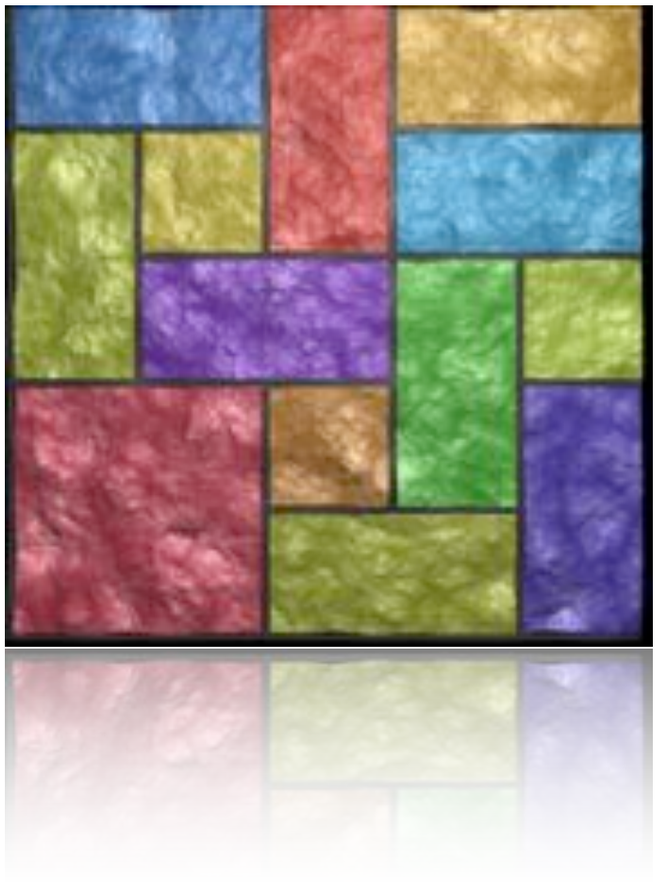






Problem

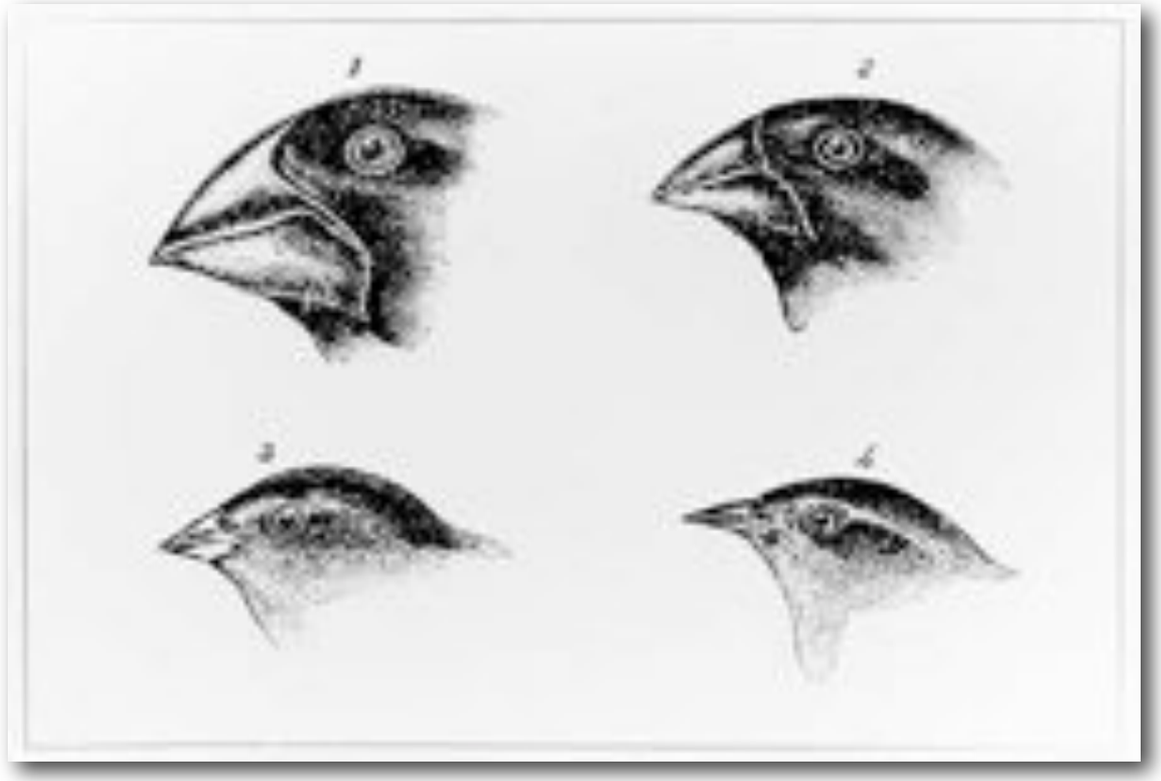
- Digital artwork creation is hard
  - Digital textures == Artist
  - Procedural textures == Programmer



```
// A tile 5x5 grid shader using fBm noise to create displacement perturbations
// and turbulence noise for color alterations.
// Paul Solt
// 4-26-09

#define snoise(x) (2*noise(x)-1)

float fBm(point p; uniform float octaves, lacunarity, gain) {
        varying float sum = 0, amp = .1;
        varying point pp = point "shader" p;
        uniform float i;

        for(i = 0; i < octaves; i+= 1) {
                sum += amp * snoise(pp);
                amp *= gain; pp *= lacunarity;
        }
        return sum;
}

float turbulence(point p; uniform float octaves, lacunarity, gain) {
        varying float sum = 0, amp = 1.2;
        varying point pp = point "shader" p;
        uniform float i;

        for(i = 0; i < octaves; i+=1) {
                sum += amp * abs(snoise(pp));
                amp *= gain; pp *= lacunarity;
        }
        return sum;
}

color varyTileColor(color Cin; float index, varyHue, varySat, varyLum;) {
        color Chsl = ctransform ("hsl", Cin);
        float h = comp(Chsl, 0), s = comp(Chsl,1), l = comp(Chsl,2);

        h += varyHue * (cellnoise(index+3)-0.5);
        s *= 1 - varySat * (cellnoise(index-14)-0.5);
        l *= 1 - varyLum * (cellnoise(index+37)-0.5);
        Chsl = color(mod(h,1), clamp(s,0,1), clamp(l,0,1));

        return ctransform("hsl", "rgb", clamp(Chsl, color(0), color(1) ));
}

// Determines if the position is brick or mortar
color createBrick(float ss, tt, x, y, width, height;
                                        color brickColor, mortarColor;
                                        float mortarThickness;) {
        color cOut = mortarColor;
        extern point P;
        extern normal N;
        vector Nn = normalize(N);
        point Psh = point "shader" (P);

        float sBrick = ss - x;
        float tBrick = tt - y;

        if( sBrick < mortarThickness || tBrick < mortarThickness) {

                float amp = fBm(Psh, 1, 2, .5);
                cOut = mortarColor * amp + color(.2);
                P += Nn * (amp / length(vtransform("shader",Nn)));
                N = calculatenormal(P);

        } else {

                float amp = fBm(Psh, 4, 2, .5);
                float t = turbulence(Psh, 5, 2, .6); // 4, 2, .5);
```

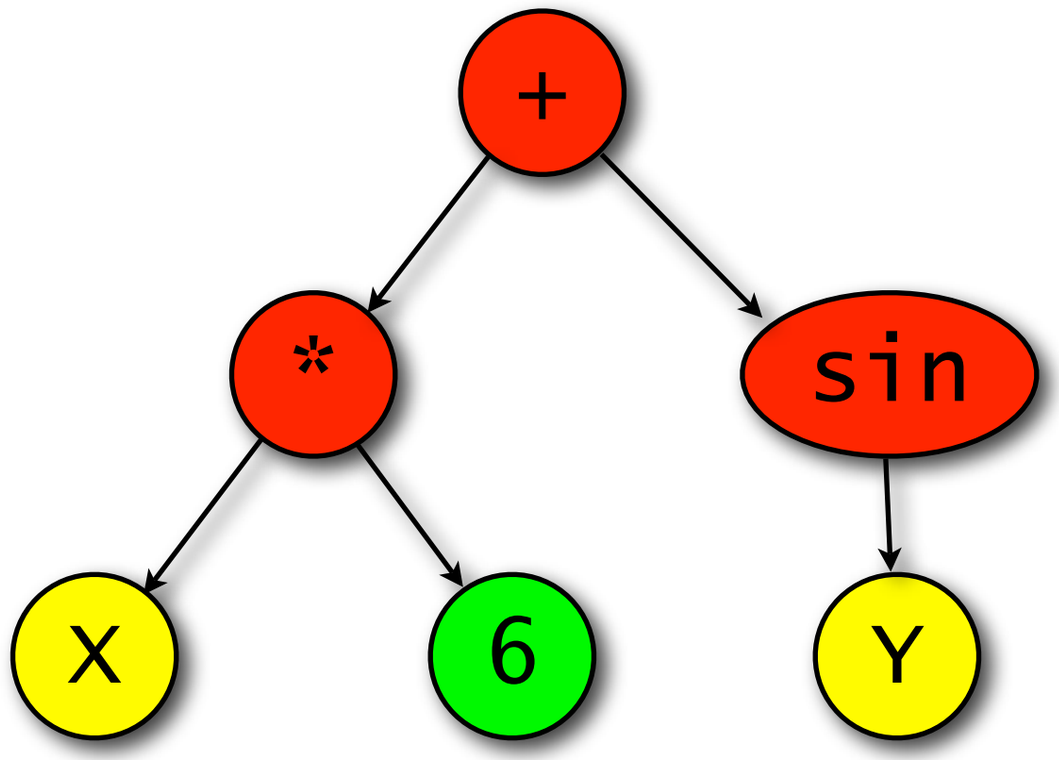
Solution

- Evolutionary computing
  - Create digital **Individuals**
    - **Genotype** - DNA
    - **Phenotype** - Organism
  - Mate and **evolve**
    - **Crossover** - trade DNA
    - **Mutation** - change DNA



Genotype

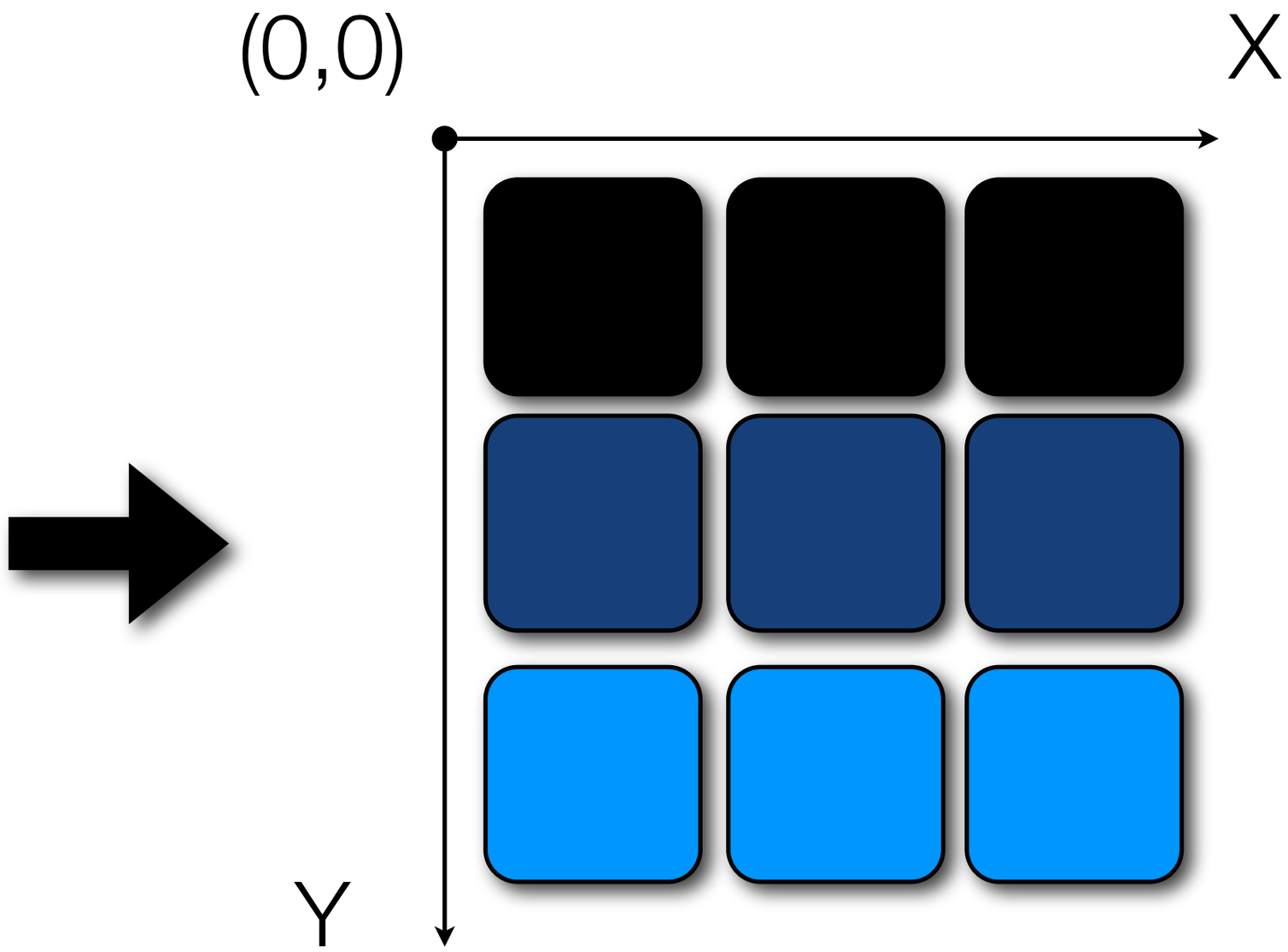
- Math expression to calculate color at a given pixel (X, Y)
    - `(+ (* X 6) (sin Y))`



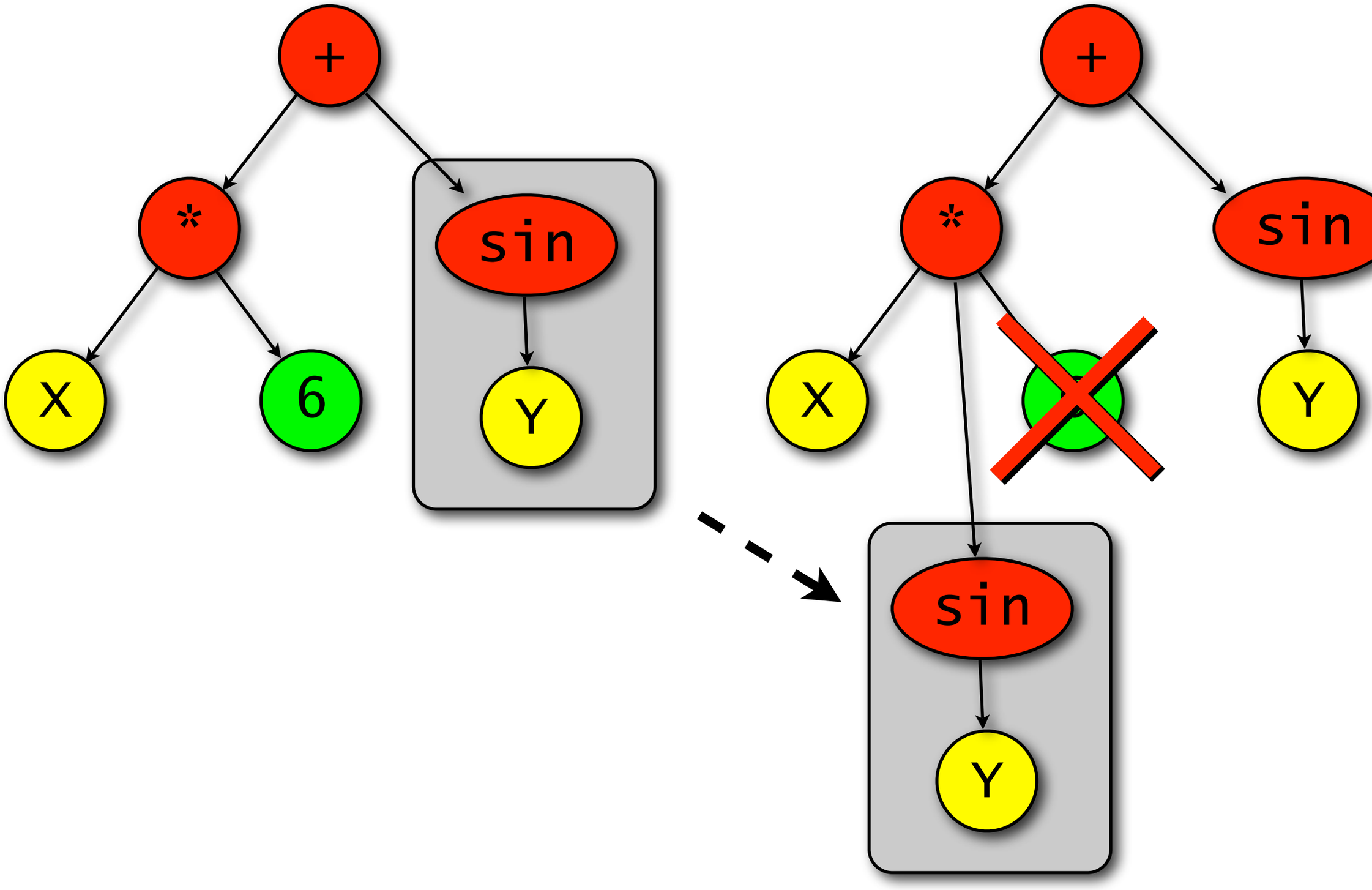
Phenotype

- 2D image output

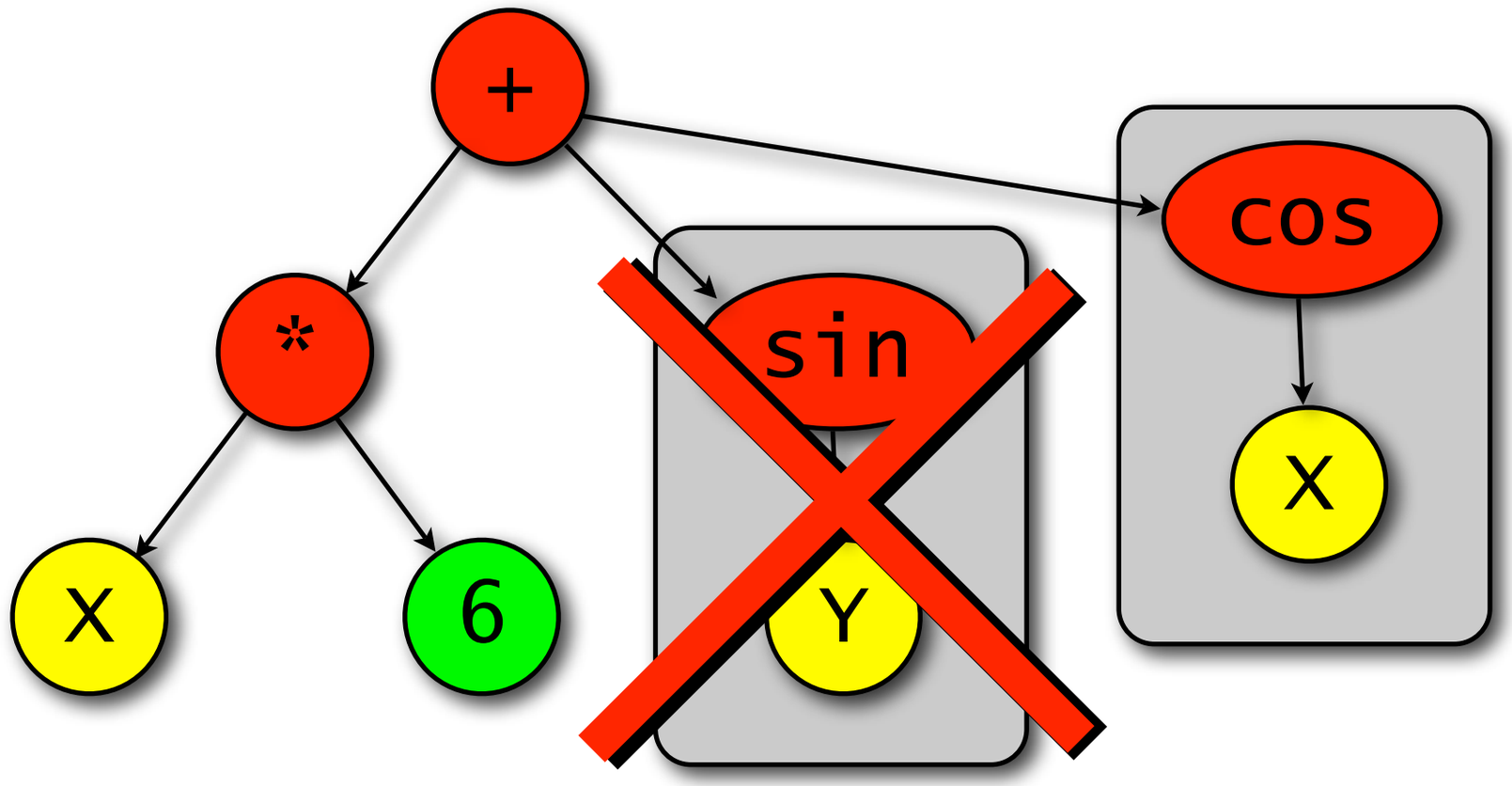
Genotype
(Y)



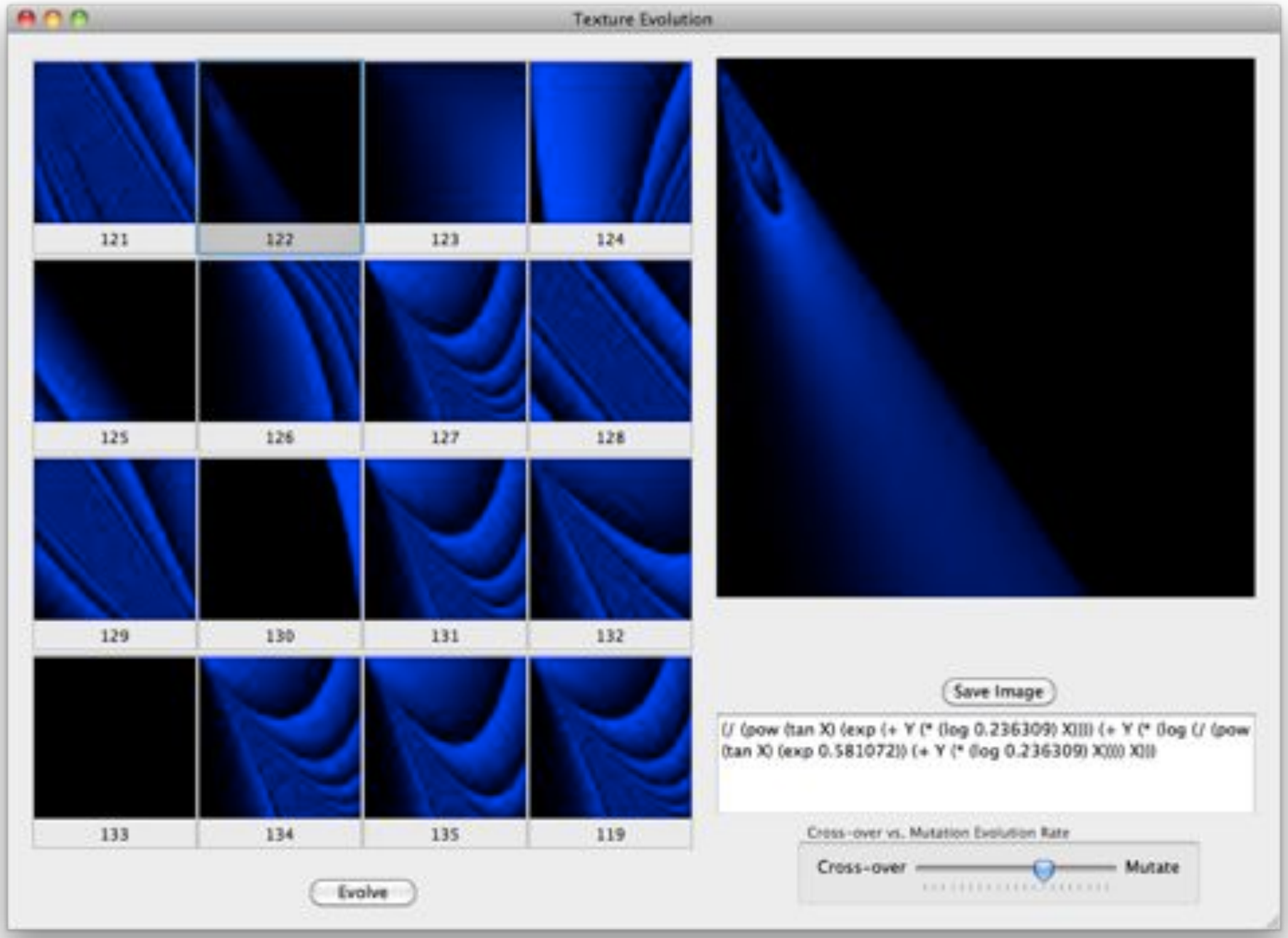
Crossover



Mutation

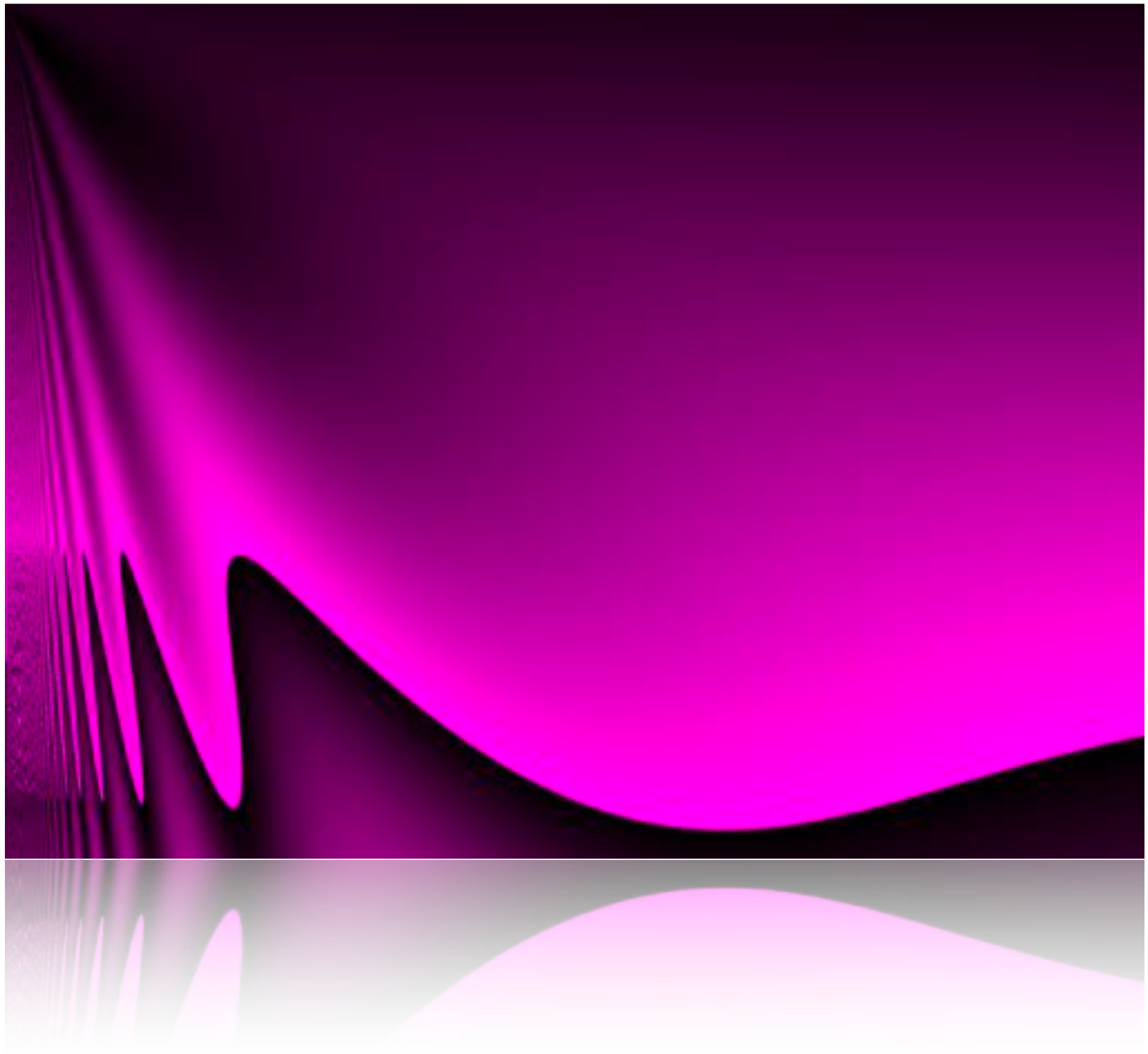


Selection

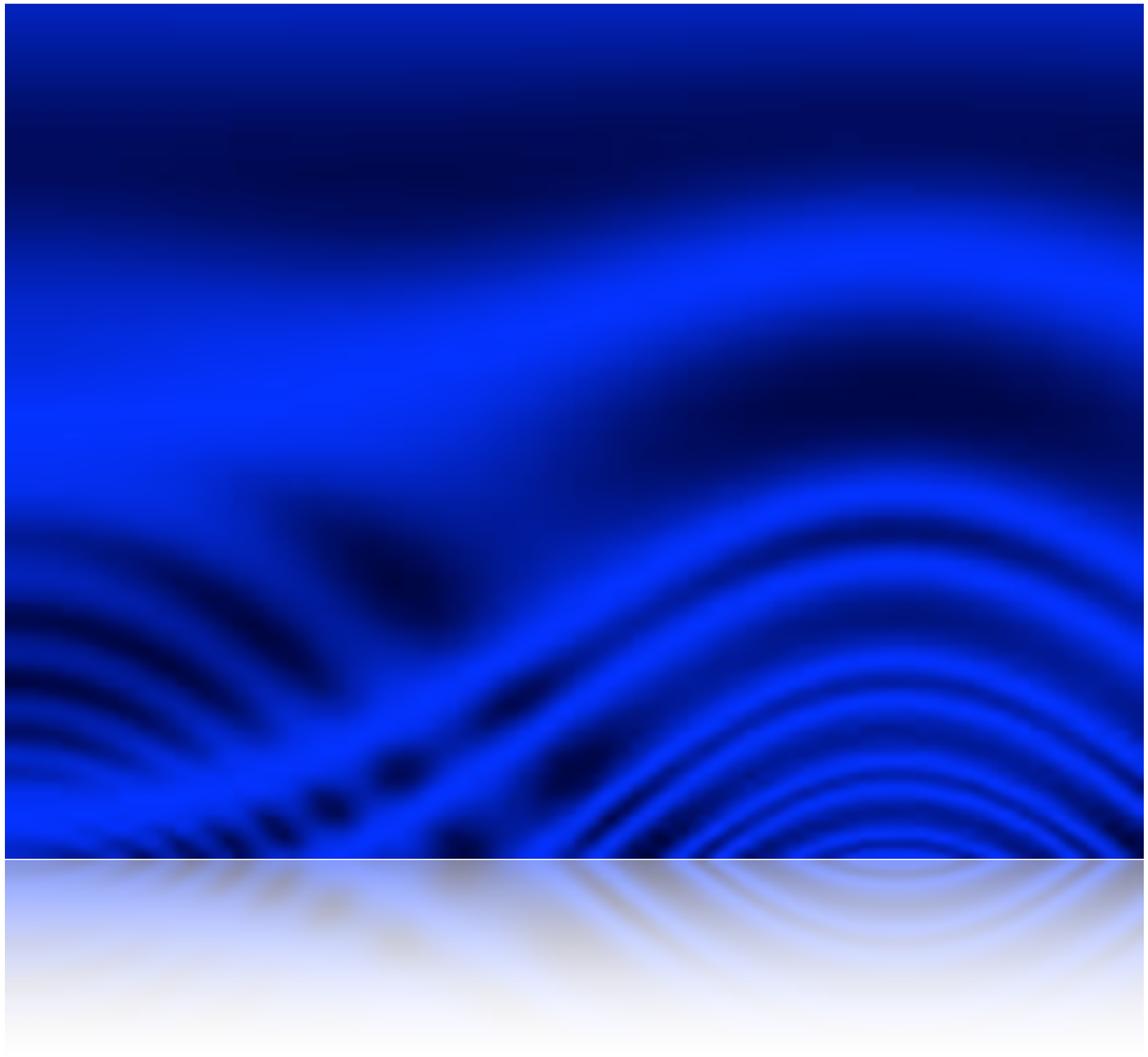

Texture Evolution
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
119
Save Image
(/ (pow (tan X) (exp (+ Y (* (log 0.236309) X)))) (+ Y (* (log (/ (pow (tan X) (exp 0.581072)) (+ Y (* (log 0.236309) X)))) X)))
Cross-over vs. Mutation Evolution Rate
Cross-over
Mutate
Evolve

Demo

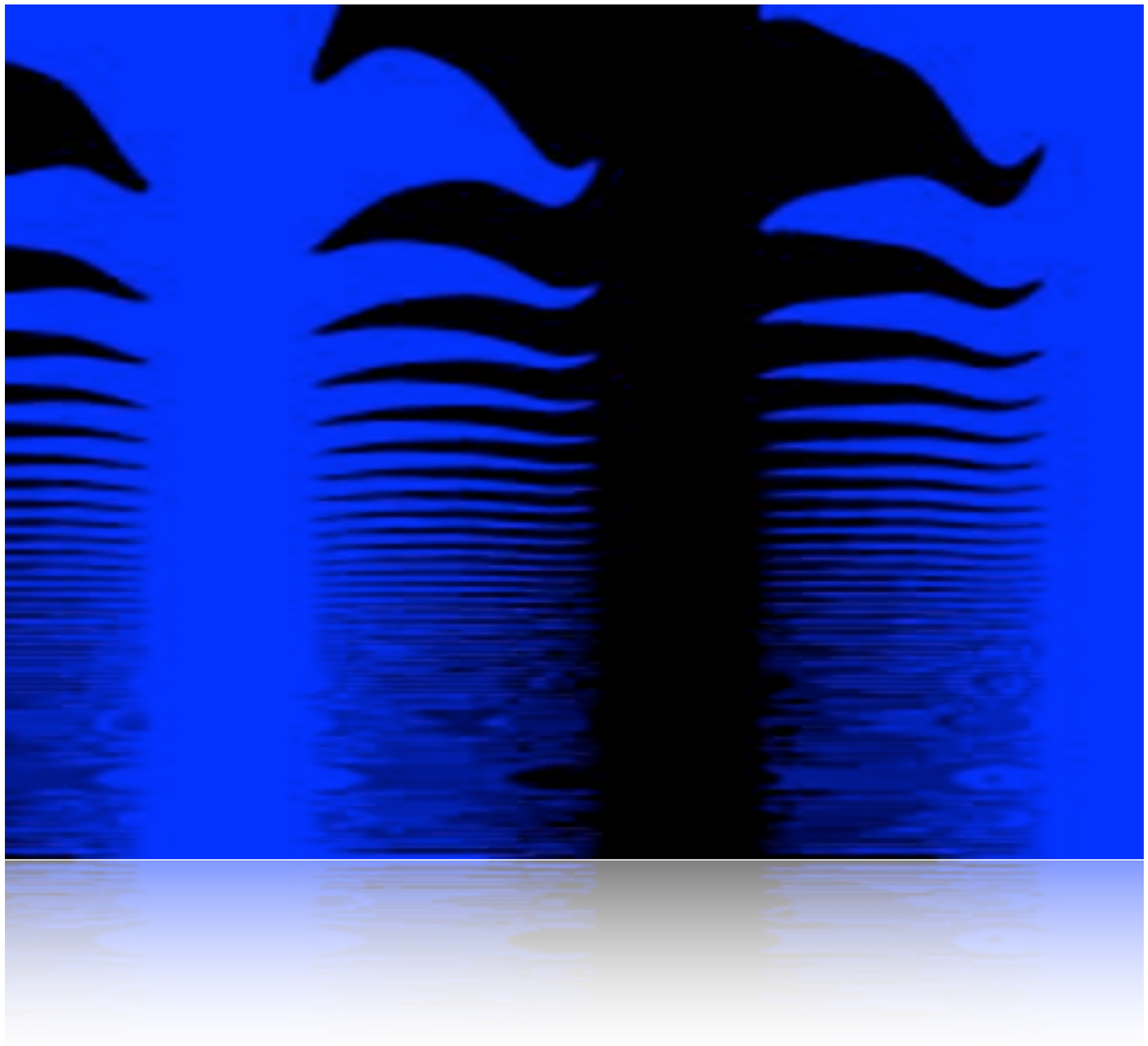
Artwork



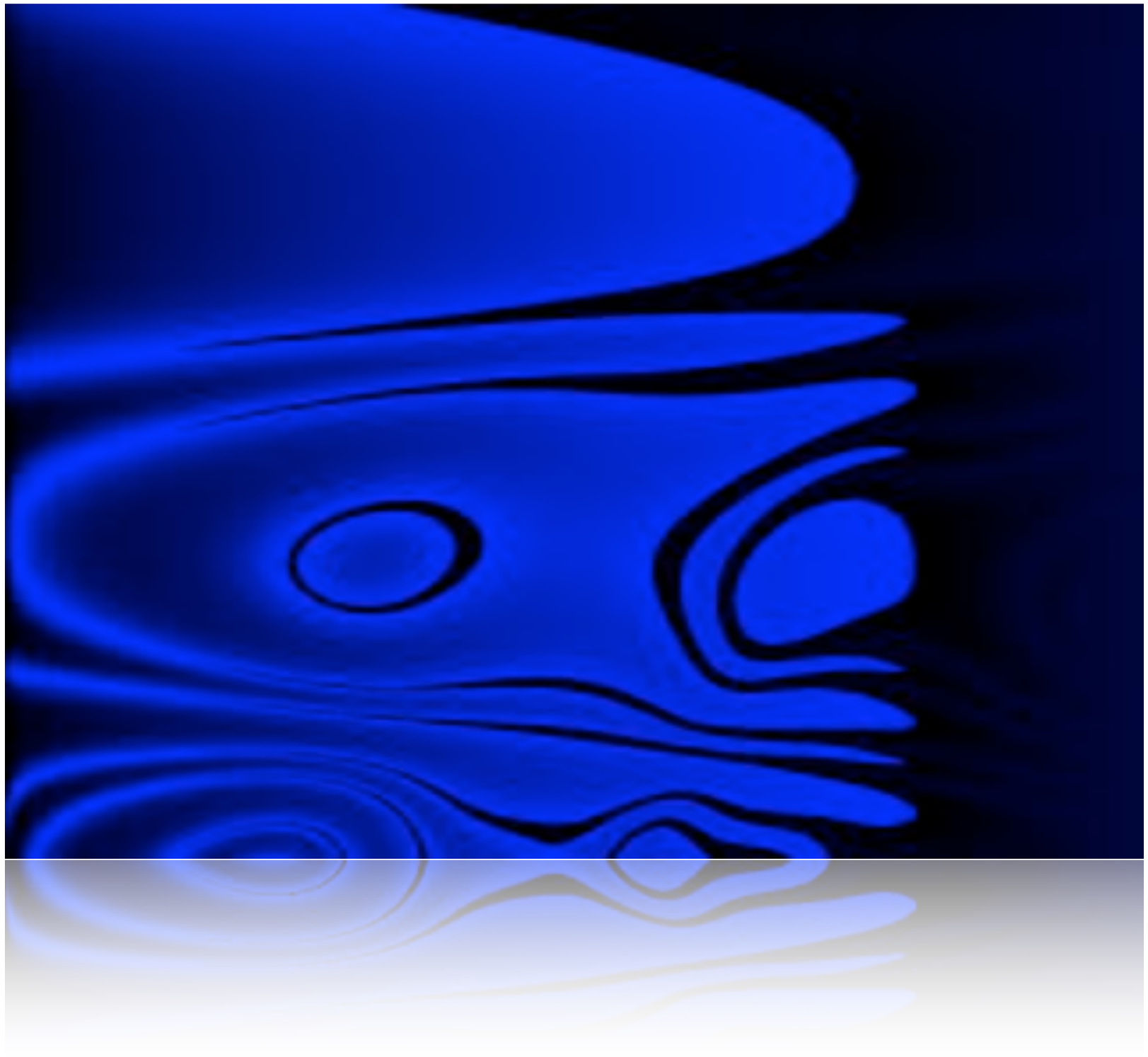
Artwork



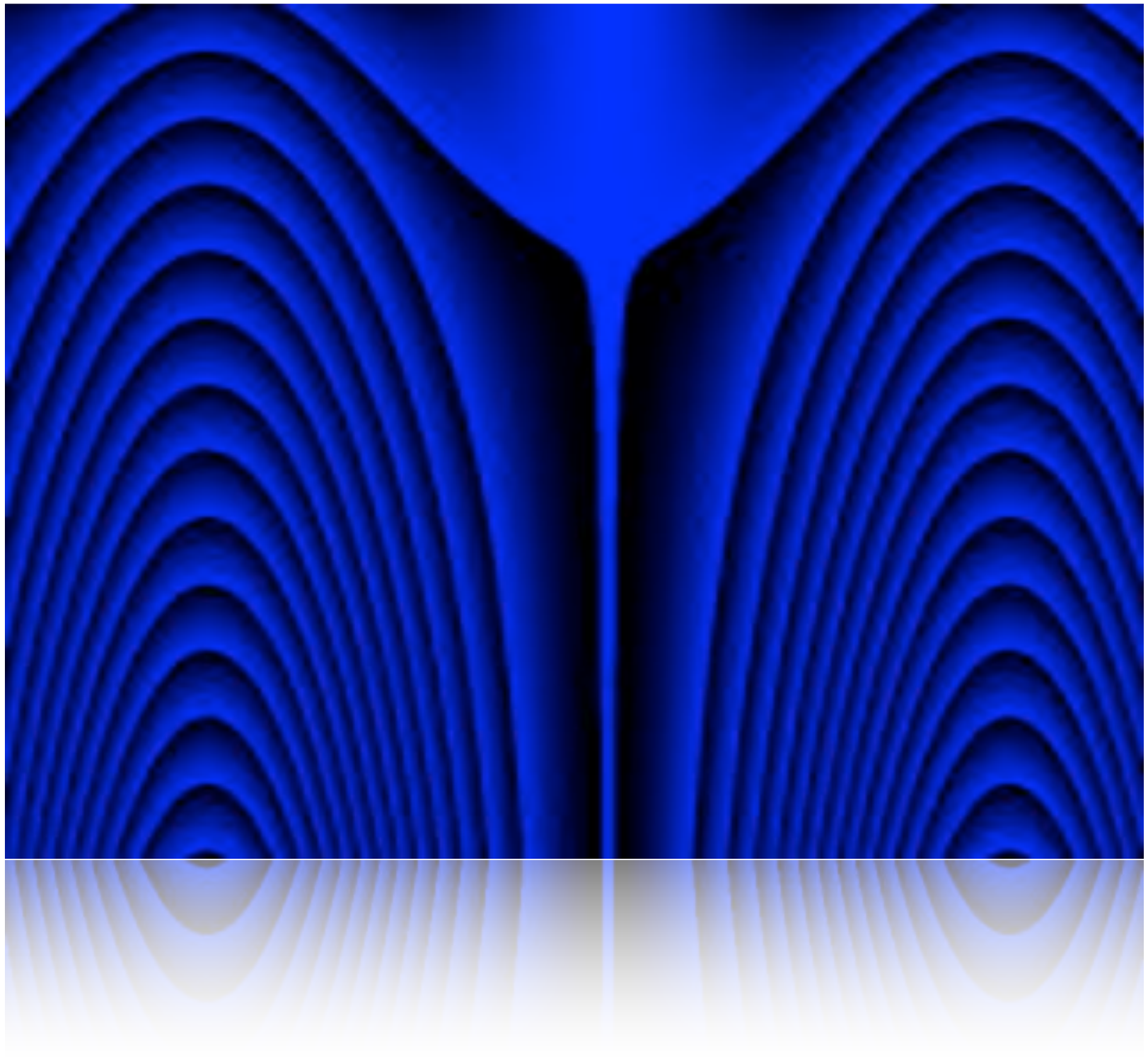
Artwork



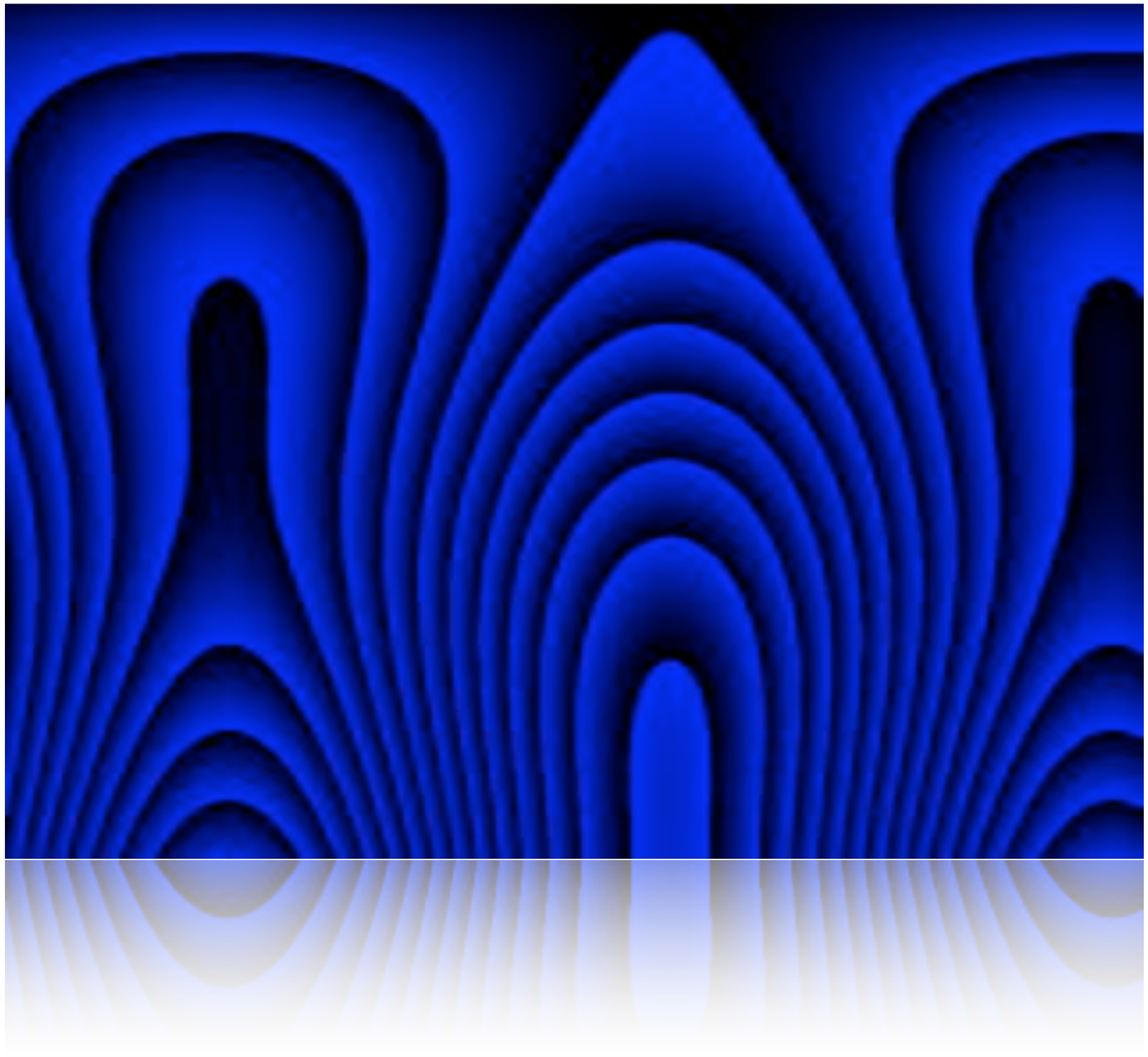
Artwork



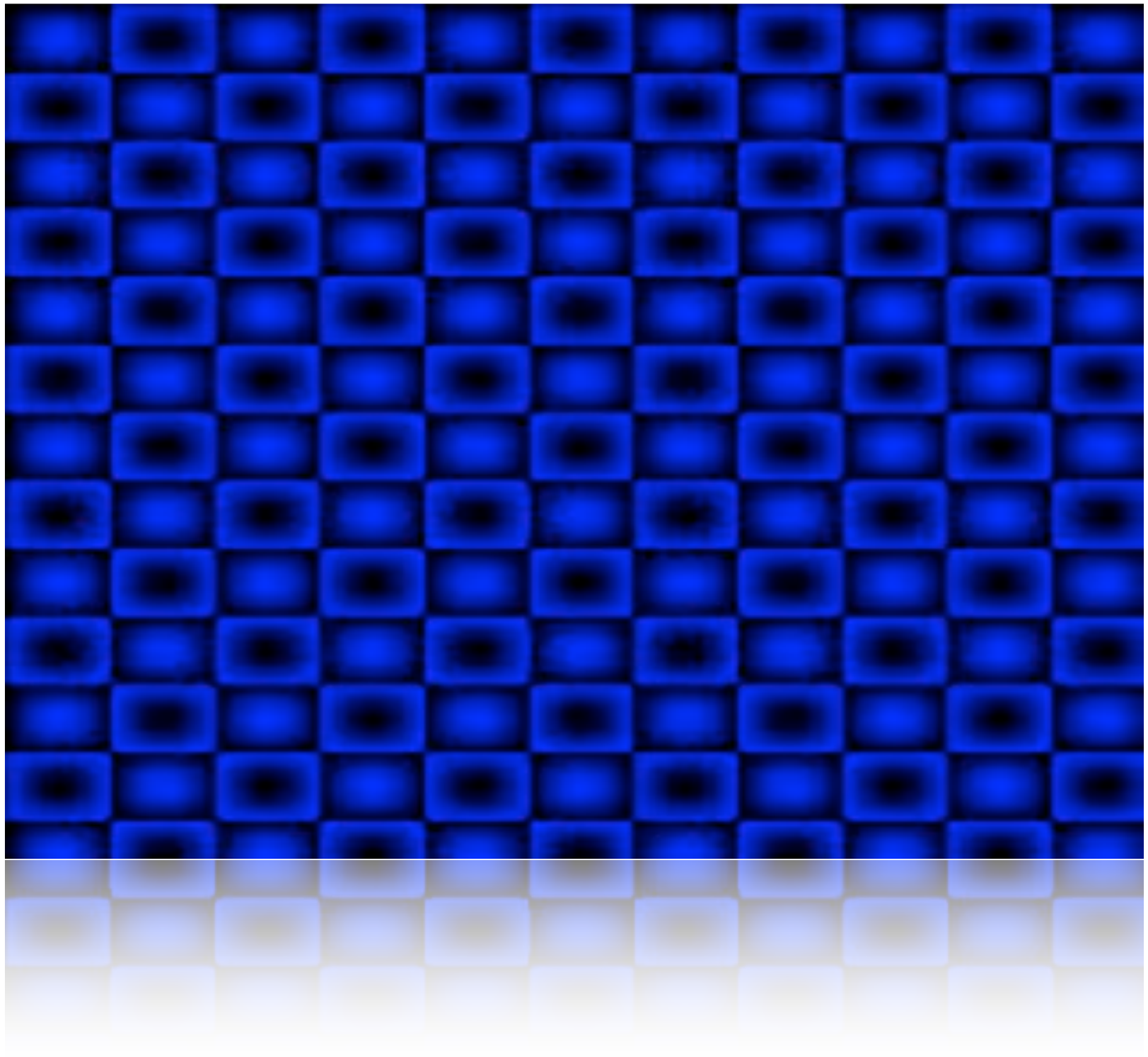
Artwork



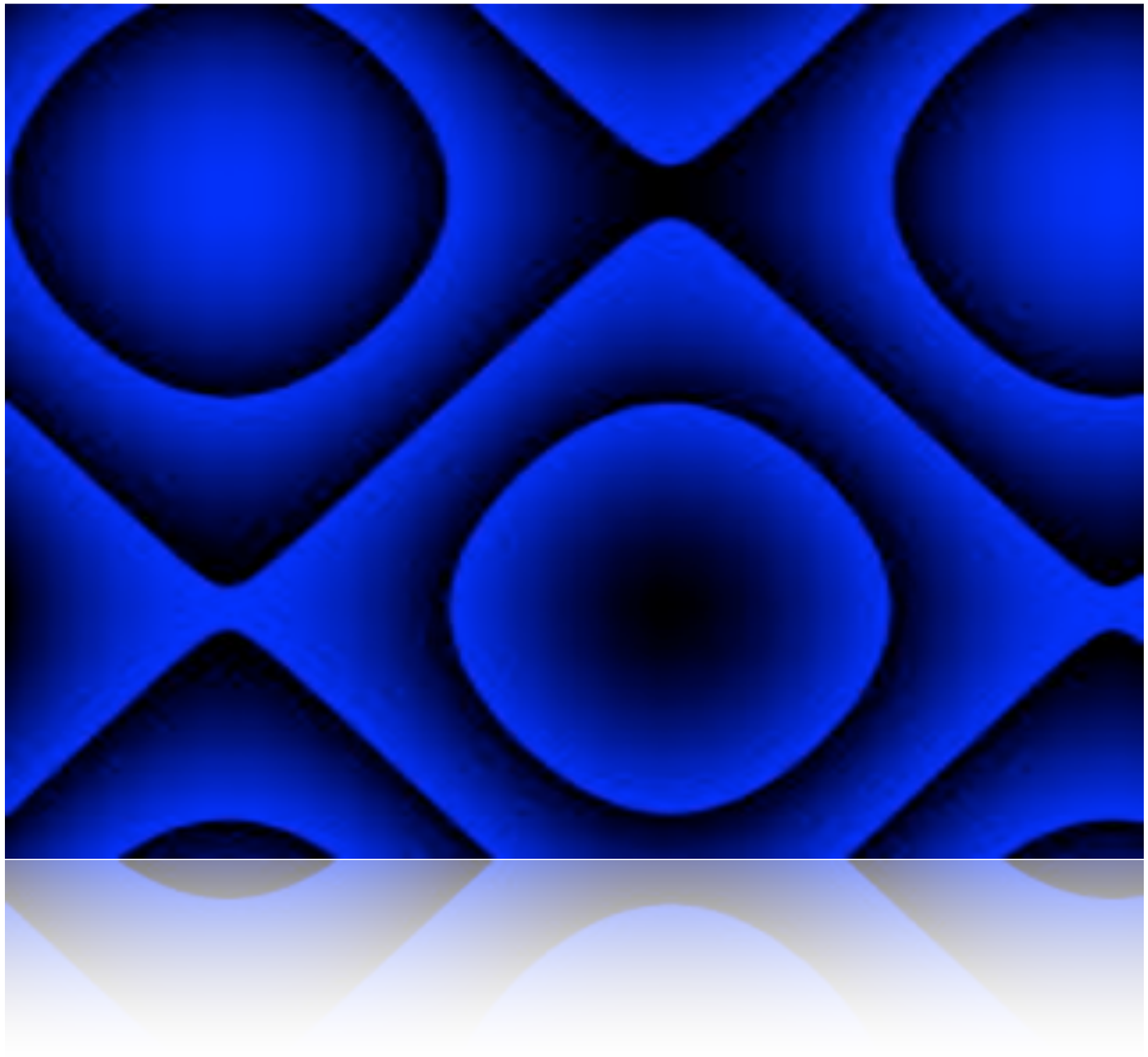
Artwork



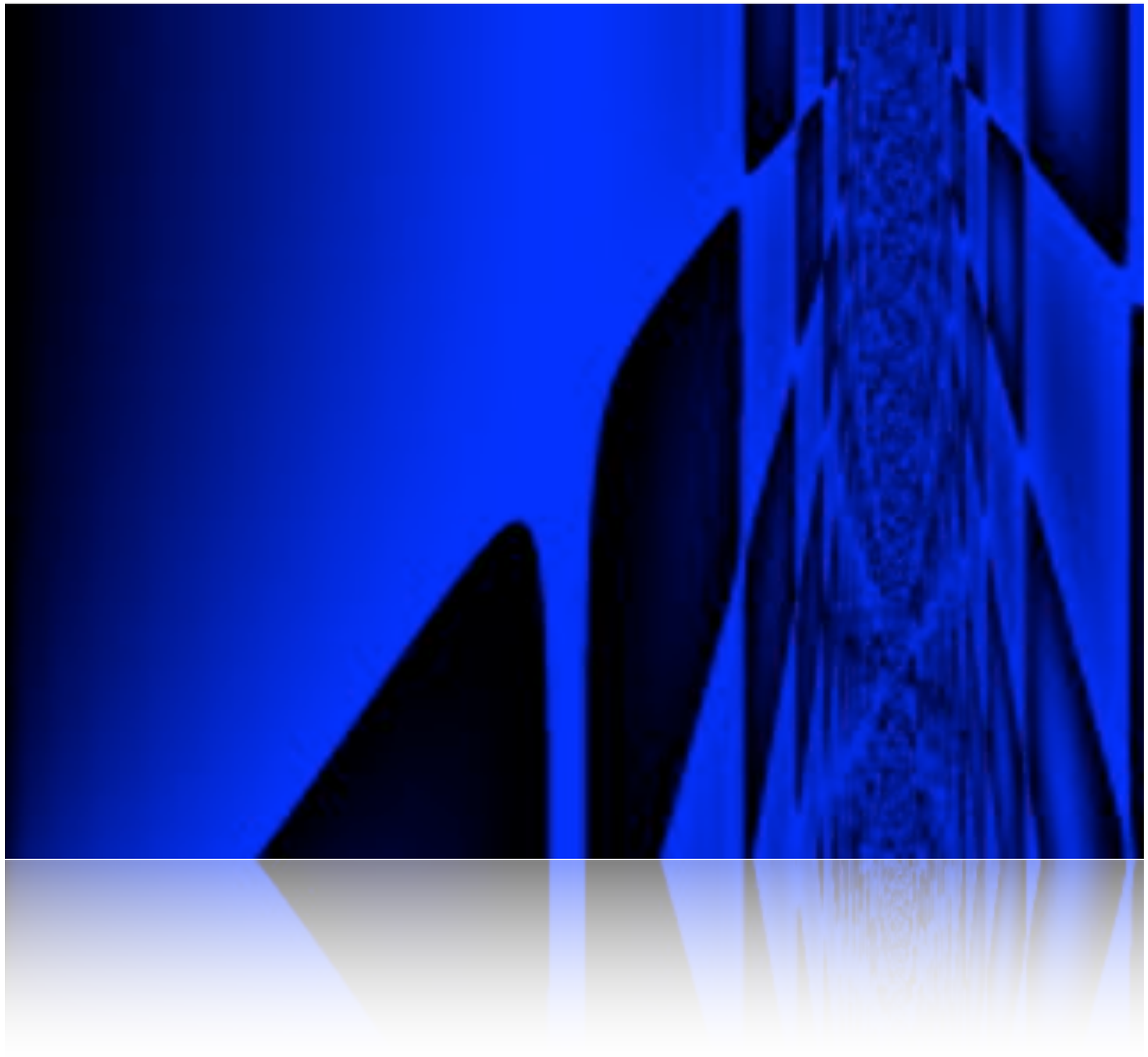
Artwork



Artwork

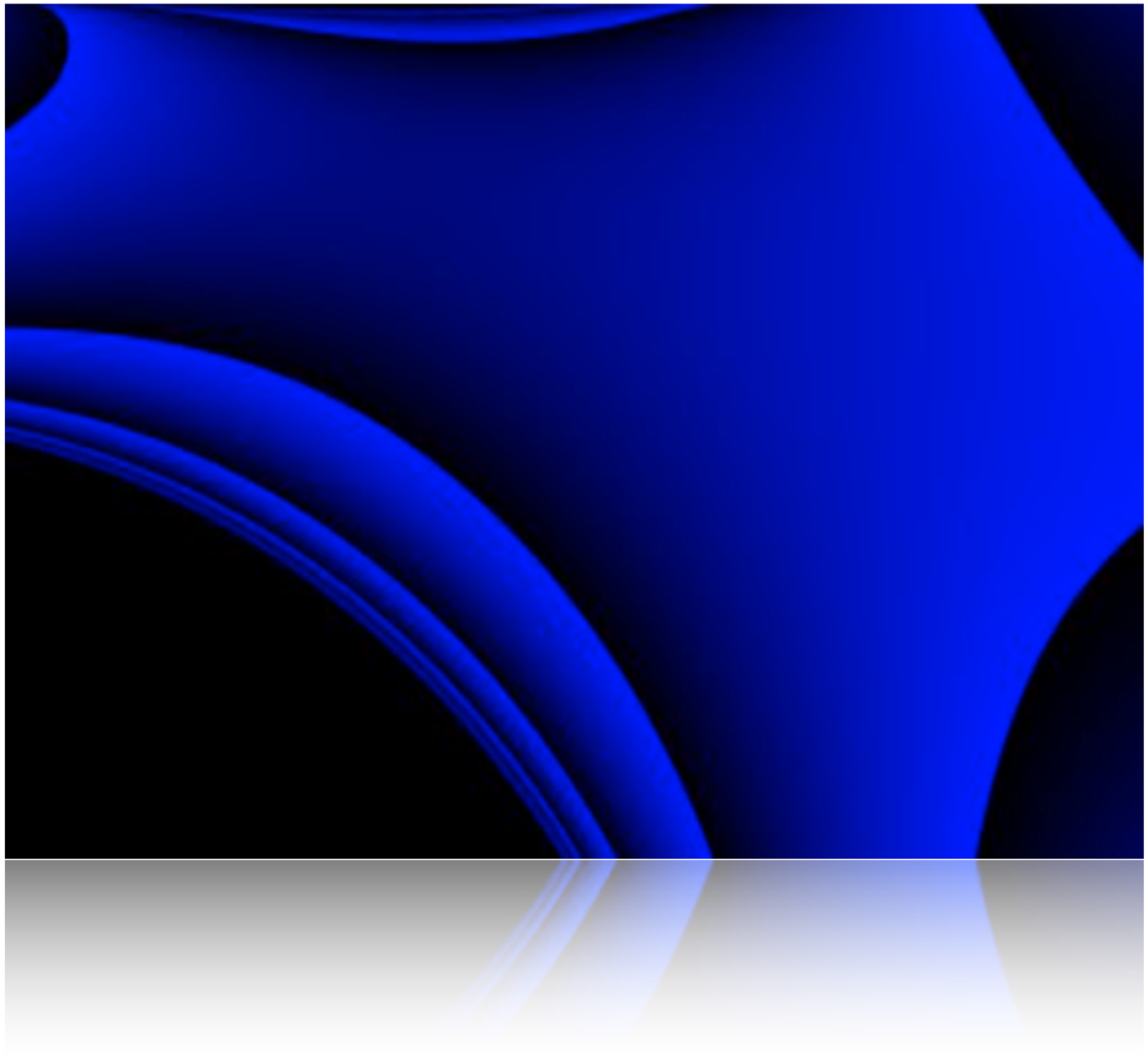


Artwork

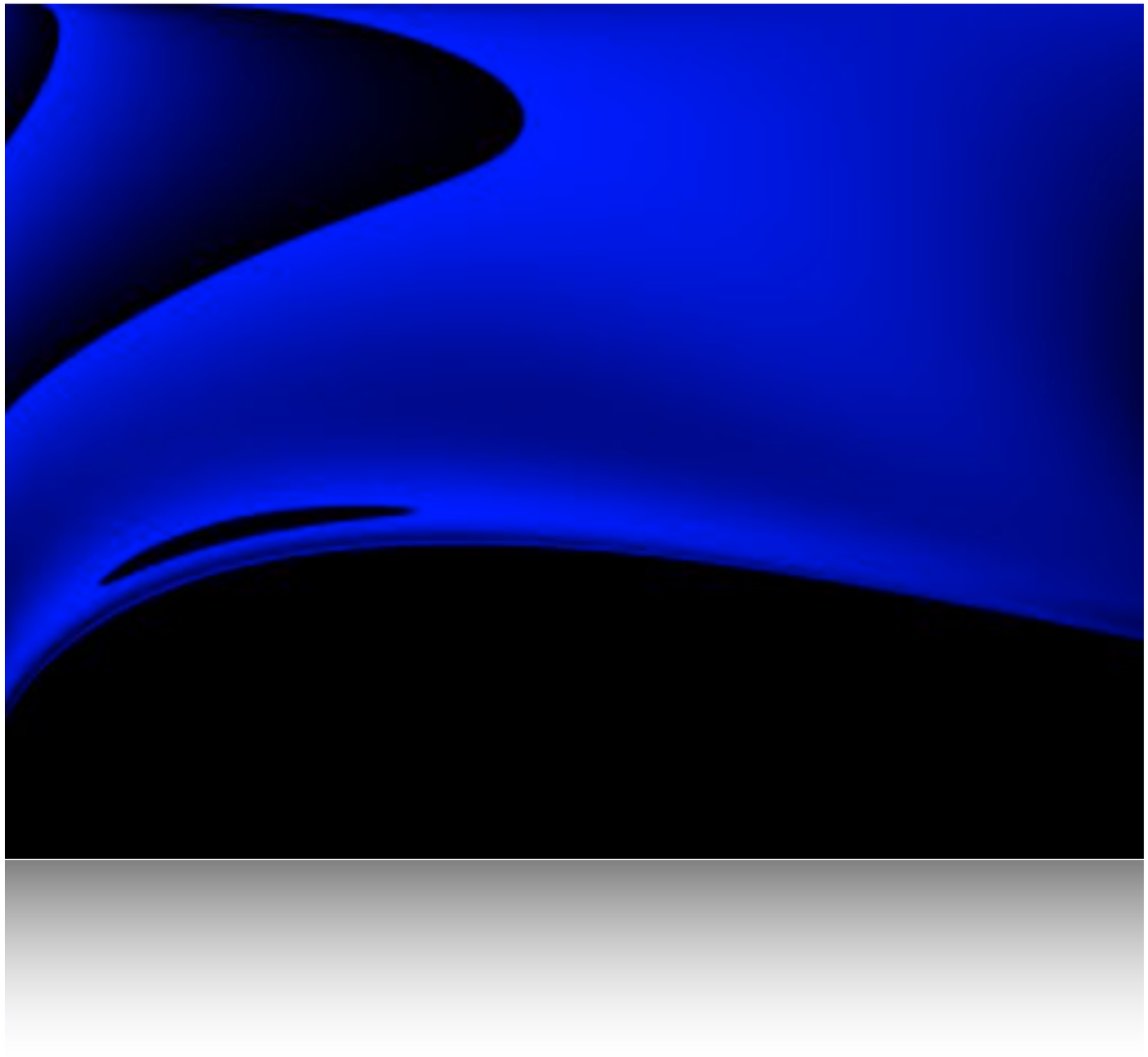


Artwork

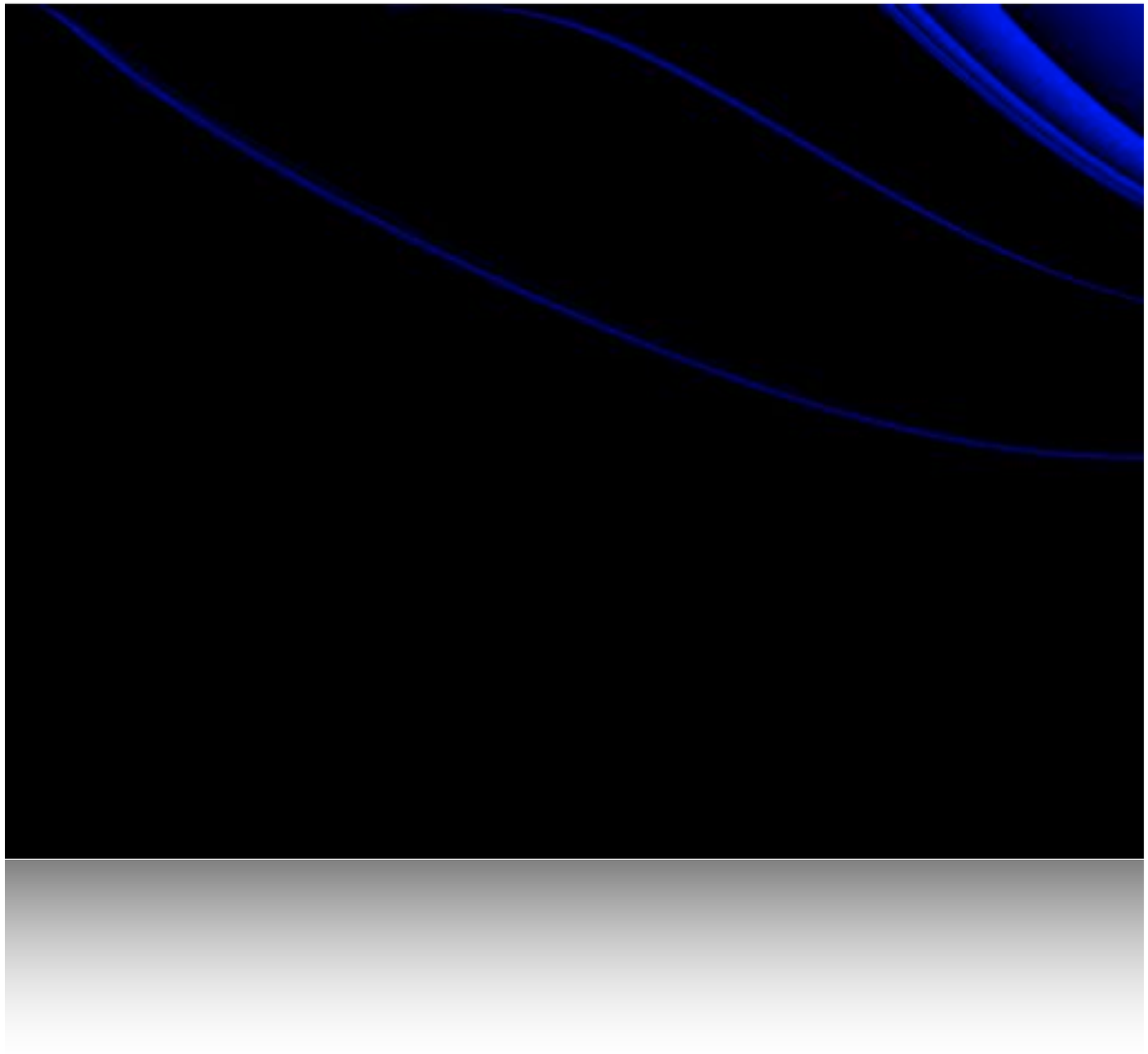
Artwork



Artwork



Artwork



Summary

- Artwork evolution can provide fast images
- No artistic or technical skill
- Artwork experimentation

Questions?

(+ (log (- (sin (+ (* X (- X 0.688415)) (cos (sin (+ (* X X) (+ (- (sin Y) (- Y (tan Y))) Y)))))) 0.488145)) (compareRound X 0.88552))