```julia
begin
    using Markdown
    using Luxor
    using Colors
    using PlutoUI
end
```

```julia
Base.:^(f::Function, n::Integer) = compose(f,n);
```

```julia
const SV = SVector{2};
```

```julia
f((x,y)) = (cos(y)^5*3^x, x^3+y) .|> p -> mod(p,1);
```

```julia
g((x,y)) = let a=2.1, b=5.9
    (sin(x*y/b)*y + cos(a*x - y), x + sin(y)/b) .|> p -> mod(p,1)
end;
```

```julia
arnold((x,y)) = (2x + y, x + y) .% 1;
```

```julia
function compose(f, n)
    function (x)
        val = x
        for _ in 1:n
            val = f(val)
        end
        return val
    end
end;
```

```julia
dist((x1,x2),(y1,y2)) = sqrt((x1-y1)^2 + (x2-y2)^2);
```

```julia
function dyndist(x,y,f,n)
    d = dist(x,y)
    for _ in 1:n
        x = f(x)
        y = f(y)
        d = max(d, dist(x,y))
    end
    d
end;
```

```
Np = 800
```

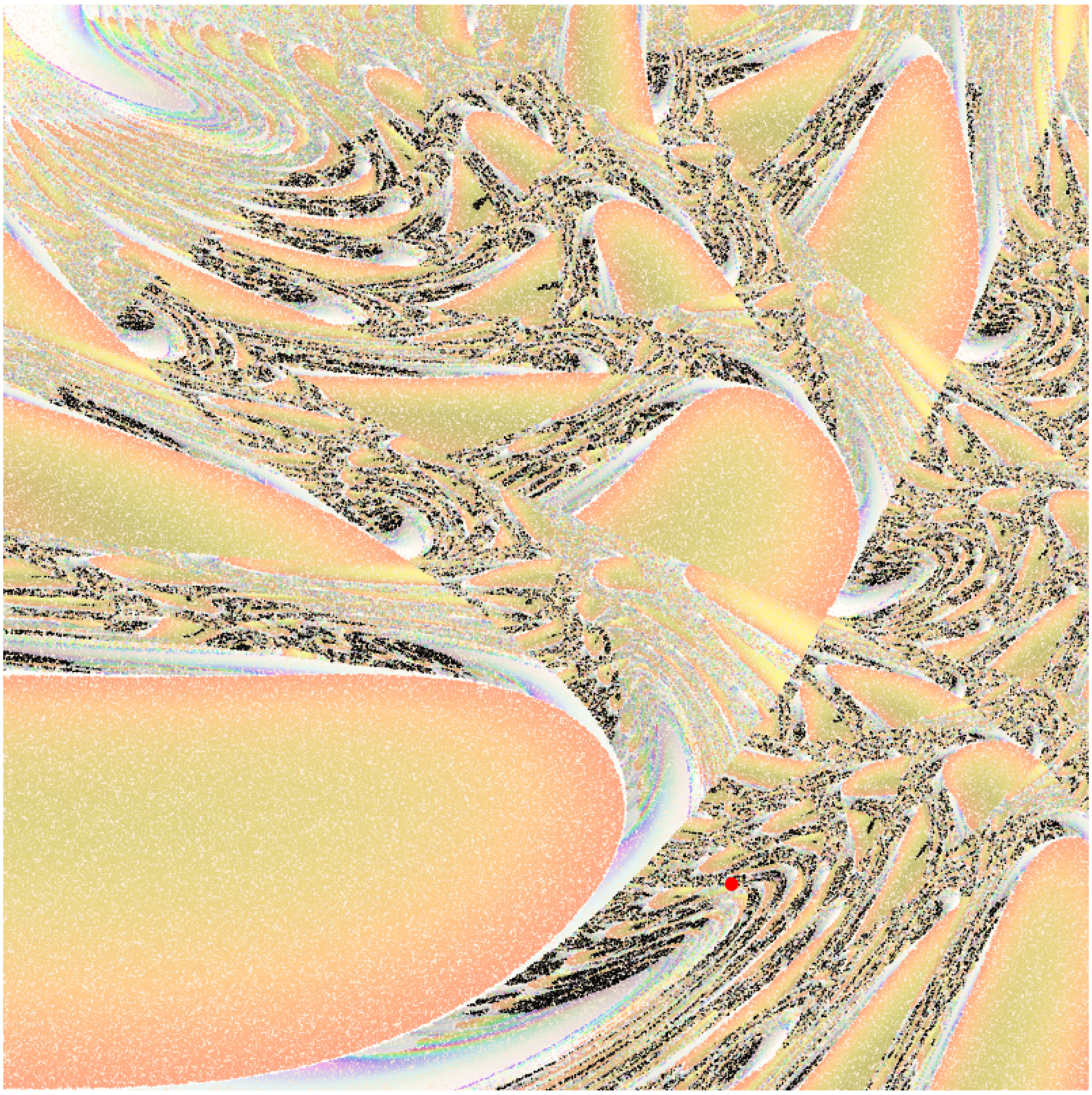
```julia
Np = 800
```

```julia
@bind cut Slider(0:0.01:1)
```

```julia
@bind n Slider(0:1:20)
```

```julia
p = (.671,.81);
```

```julia
fun = f;
```



```julia
let s = 800
    Drawing(s, s, "my-drawing.png")
    background("antiquewhite")
    for x in 0:(1/Np):1, y in 0:(1/Np):1
        x = rand() # * .01 + p[1] - .005
        y = rand() # * .01 + p[2] - .005
        let (tx,ty) = (fun^n)((x,y))
            setcolor(HSV(360tx,.5ty,dyndist((x,y),p,fun,n)))
        end
        if dyndist((x,y),p,fun,n) < cut
            setcolor("black")
        end
        circle(Point(s .* (x, y)), sqrt(s)*0.026, :fill) # .+ (.005,.005)
.- p
    end
    setcolor("red")
    circle(s .* Point(p), 5, :fill) # Point(.5,.5)
    finish()
    preview()
end
```