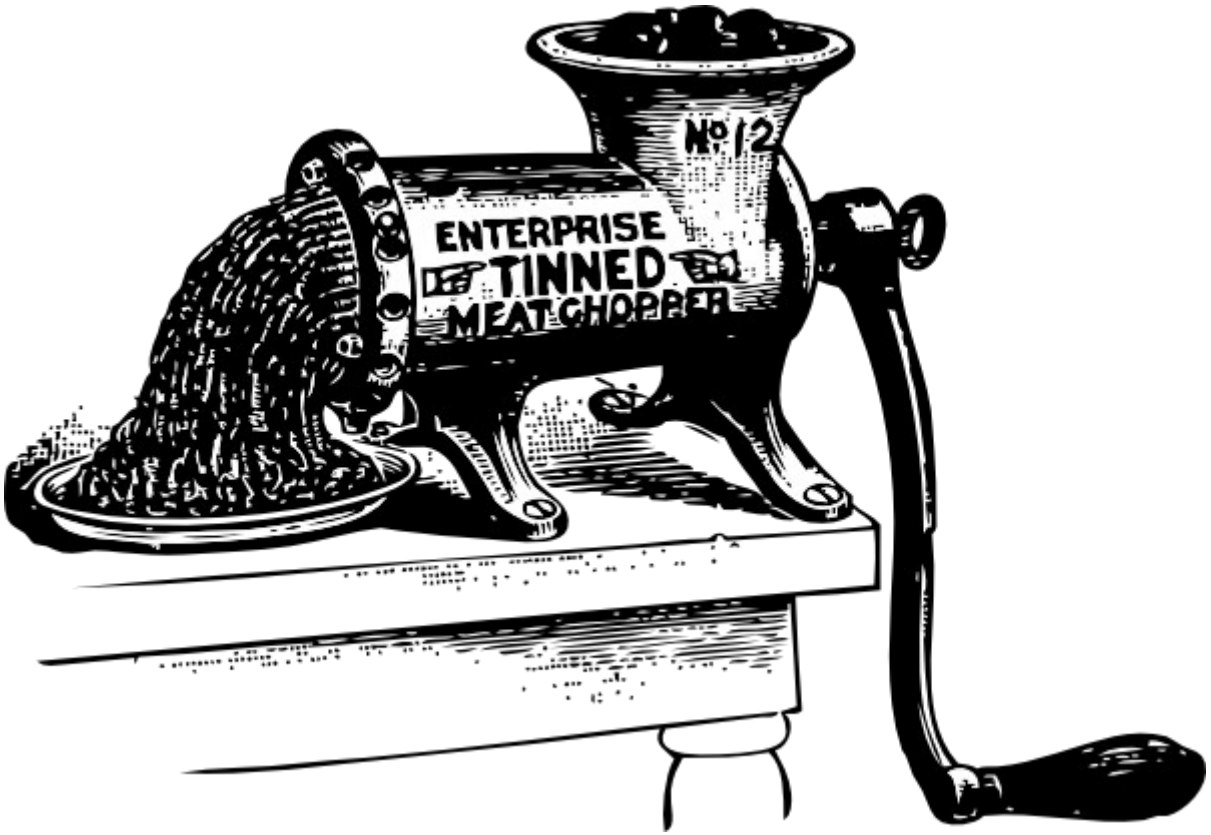
Number generators for Generative Art

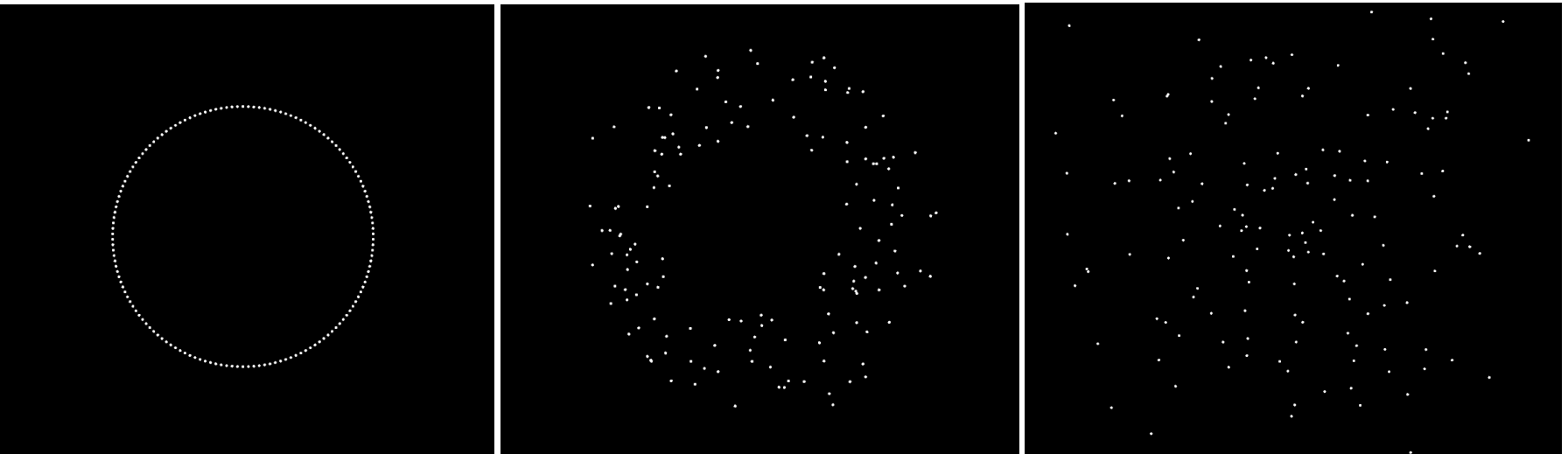


“I think there's something strangely musical about noise.”
Trent Reznor

randomness

ideal places to use

- Randomly assign position, size, colour …
- Apply random offset to more carefully chosen values
- Randomly move
- Choose random times (or offsets) for events



randomPerimeter

manipulating randomness

- Can use **`random()`** to choose actions with specific probabilities

```
float r = ofRandom(1); //generate between 0-1
if (r < 0.3) { //do something: 30% prob }
else if (r < 0.7) { //do something: 40% prob}
else { // do something: 30% prob }
```

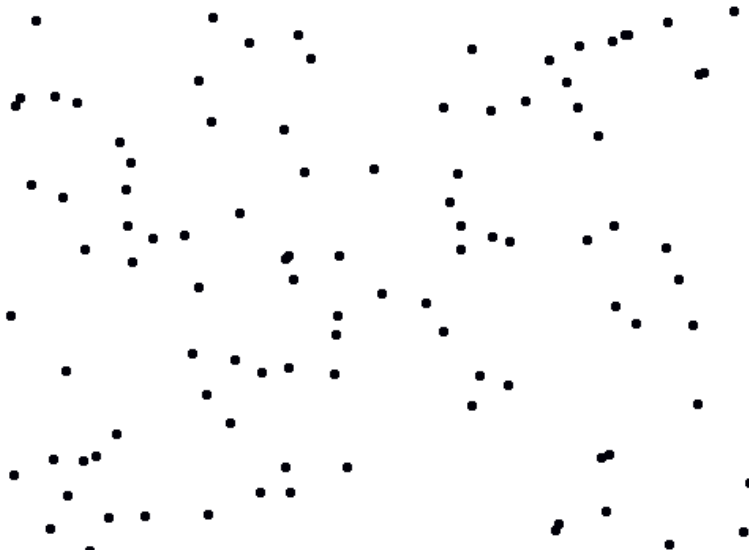
random seeds

```
ofSeedRandom();
ofSeedRandom(int val);
```

problems with random

- using random coordinates isn't helpful
- doesn't allow for natural/organic shapes or movements

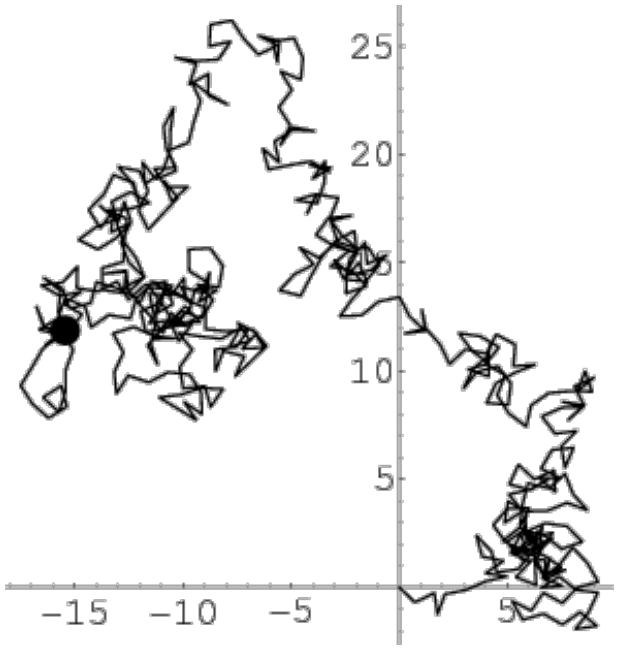
```
//completely random position
ofPoint pos;
pos.x = ofRandom(ofGetWidth());
pos.y = ofRandom(ofGetHeight());
```



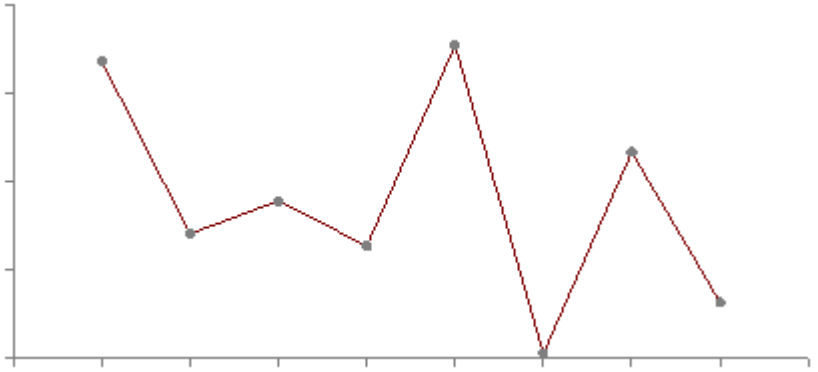
possible solutions?

random walk

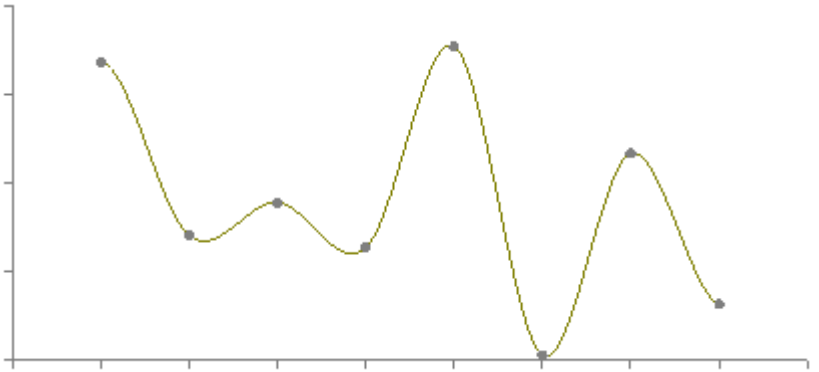
```
//start in the middle and make small steps
ofPoint pos(ofGetWidth()/2, ofGetHeight()/2));
pos.x += ofRandom(-1,1);
pos.y += ofRandom(-1,1);
```



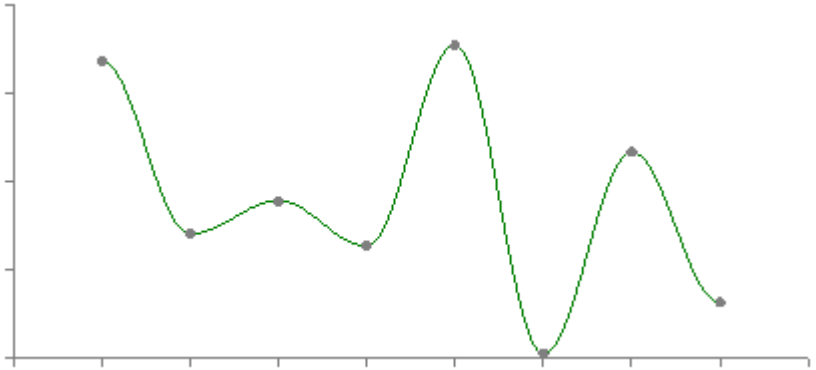
use interpolation



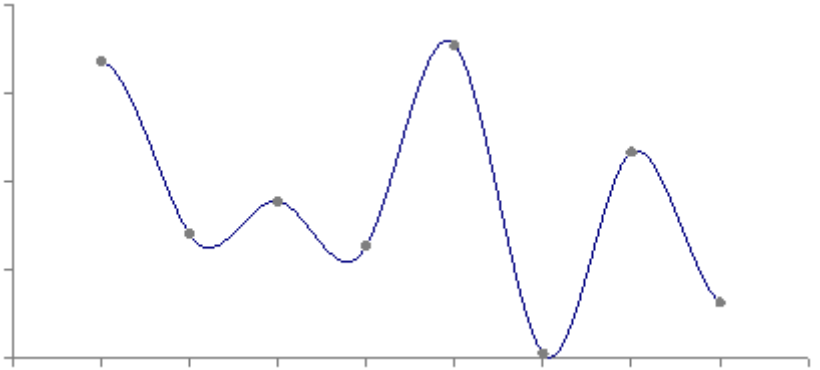
linear interpolation



hermite interpolation



cosine interpolation



cubic interpolation

* expensive but smoothest results

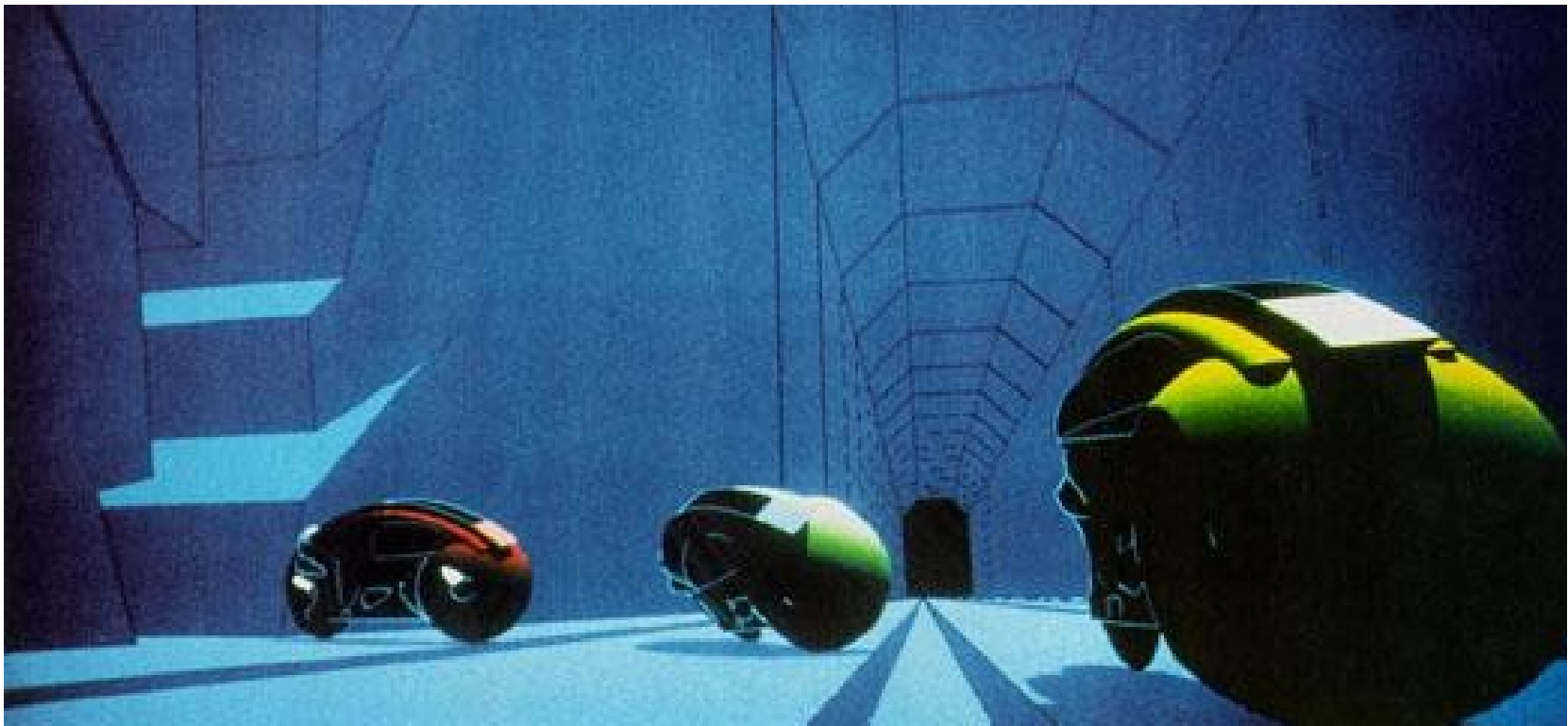
perlin noise

- an algorithm to design procedural textures
- this way... we don't need images (ideal for low bandwidth)

perlin noise

a bit of history

- invented by Ken Perlin
- first used in the movie Tron to achieve realistic textures



perlin noise

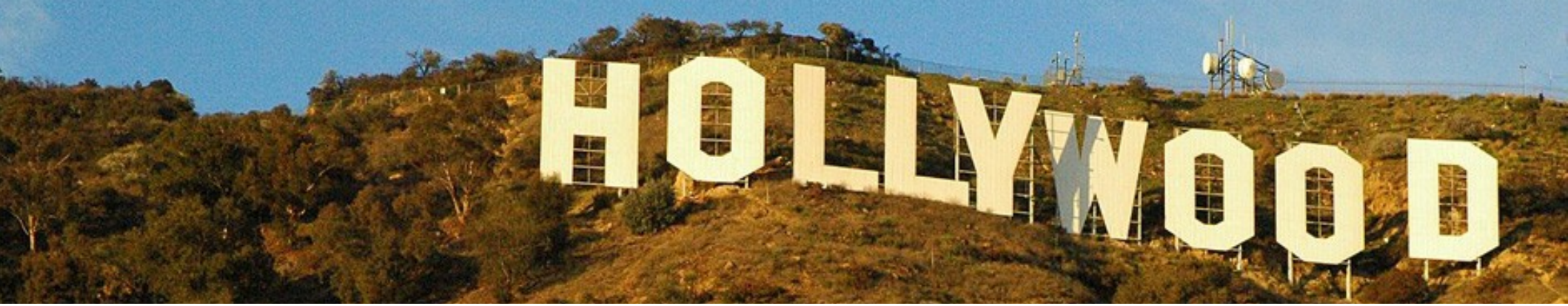
- creates more organic forms
- it uses a pseudorandom series of numbers with more natural flow
- it's the “salt and pepper” that makes shapes, movements, colours naturally variable and interesting

perlin noise: like a paintbrush

- A traditional paintbrush:
  - the hair have particular statistical properties (size, distance between them, hardness, etc)
  - we don't care about the order of the hair every time we use a brush to draw a line
- Painters have been using a “controlled random number generator” for ages
- Perlin noise allows us to do it with mathematics

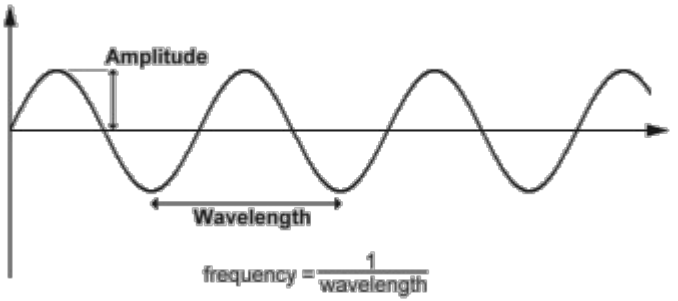
We live in a Perlin world

- the distribution of patchy grass on a field
- waves in the sea
- the movements of an ant
- the movement of branches of a tree
- patterns in marble
- weather systems
- winds

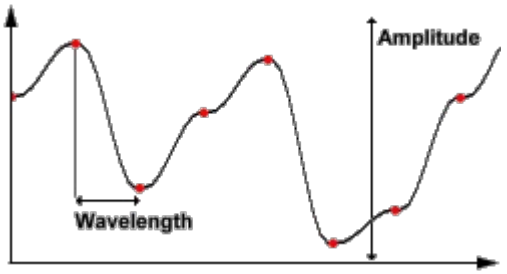


- from 1990 onwards every movie with special effects uses Perlin noise
- examples:
  - James Cameron (Abyss,Titanic, κλπ)
  - Animation (Lion King, Moses, κλπ)
  - Schwarzenegger (T2, True Lies, ...)
  - Star Wars
  - Star Trek
  - Batman
  - ...and many more

terminology



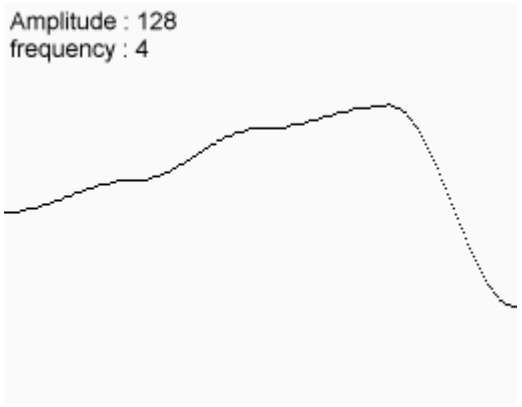
for a sin wave

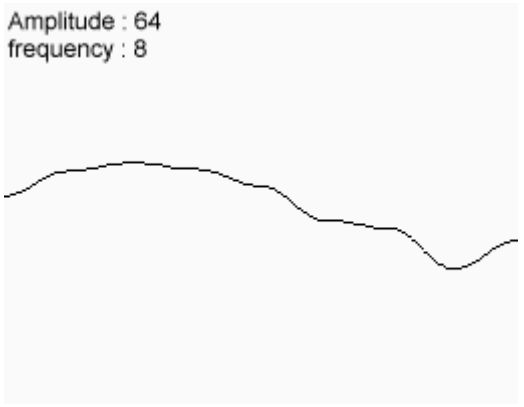


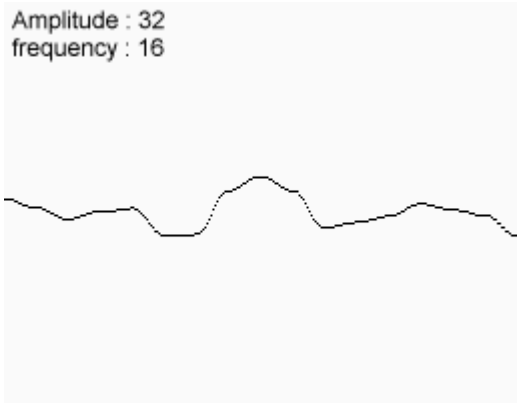
for a noise wave

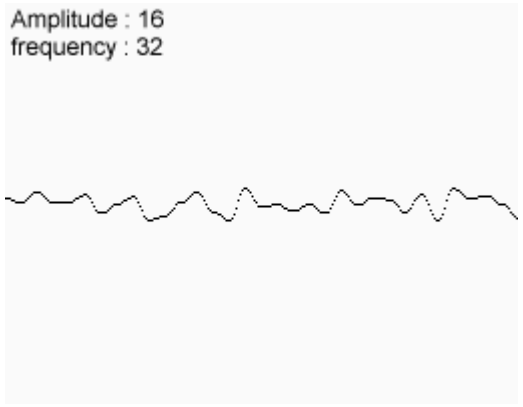
how is it made?

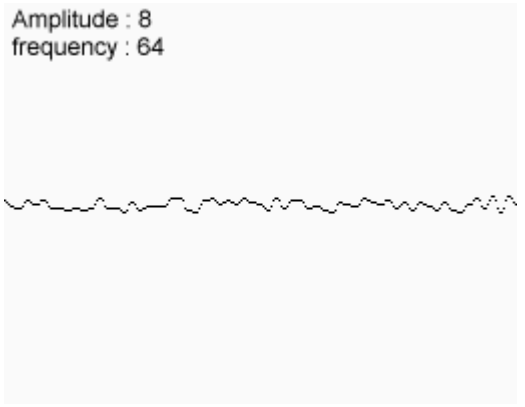
adding different noise waves together

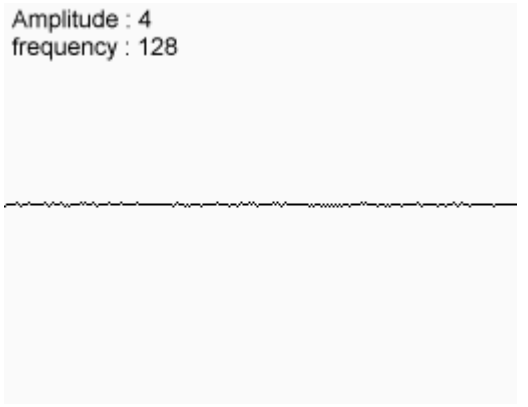




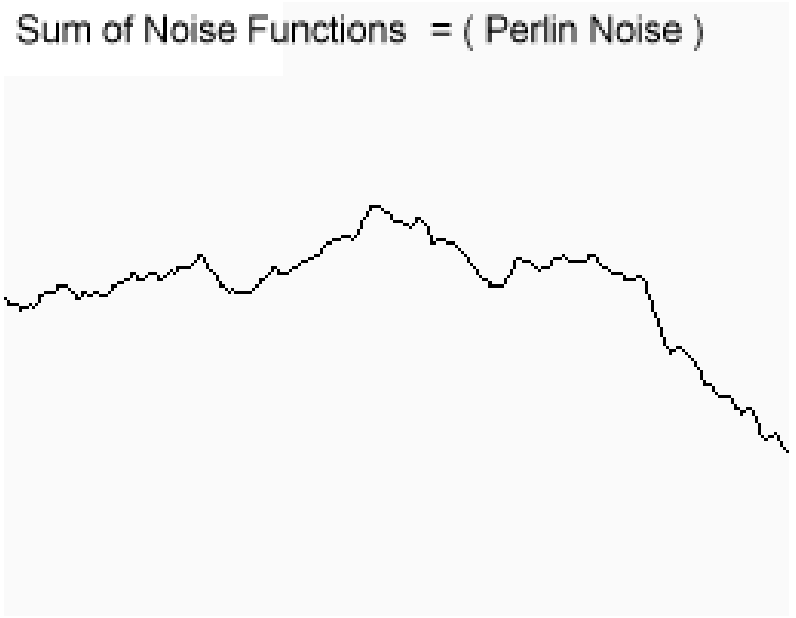








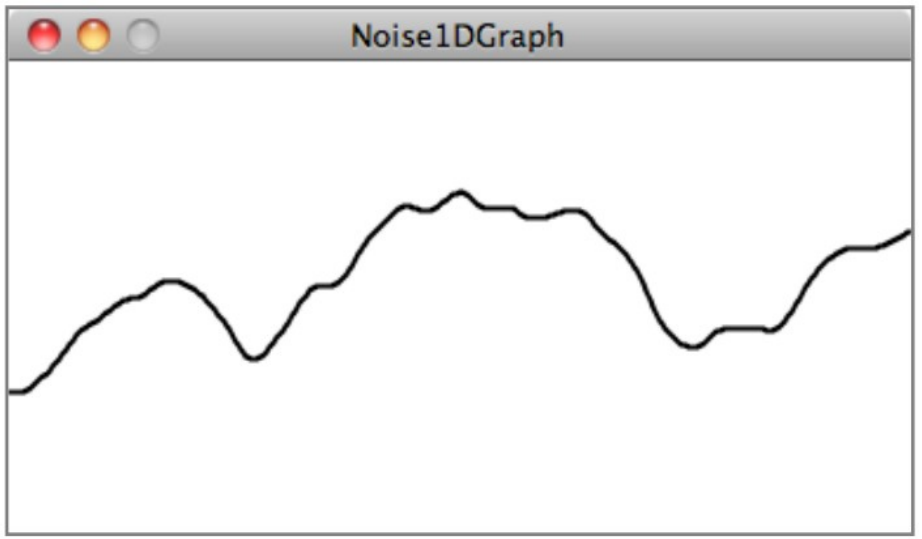
...you get Perlin noise

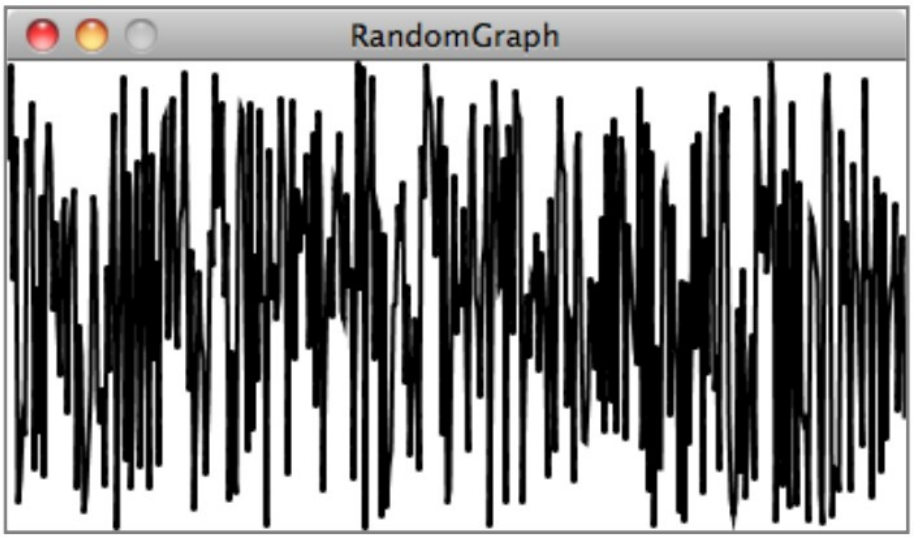


`ofNoise()`

the details

- `ofNoise()` returns a value between 0-1
- `ofSignedNoise()` returns a value between -1 and 1
- we pass as parameters the point in time we want
- ie `ofNoise(5)` always returns the same value





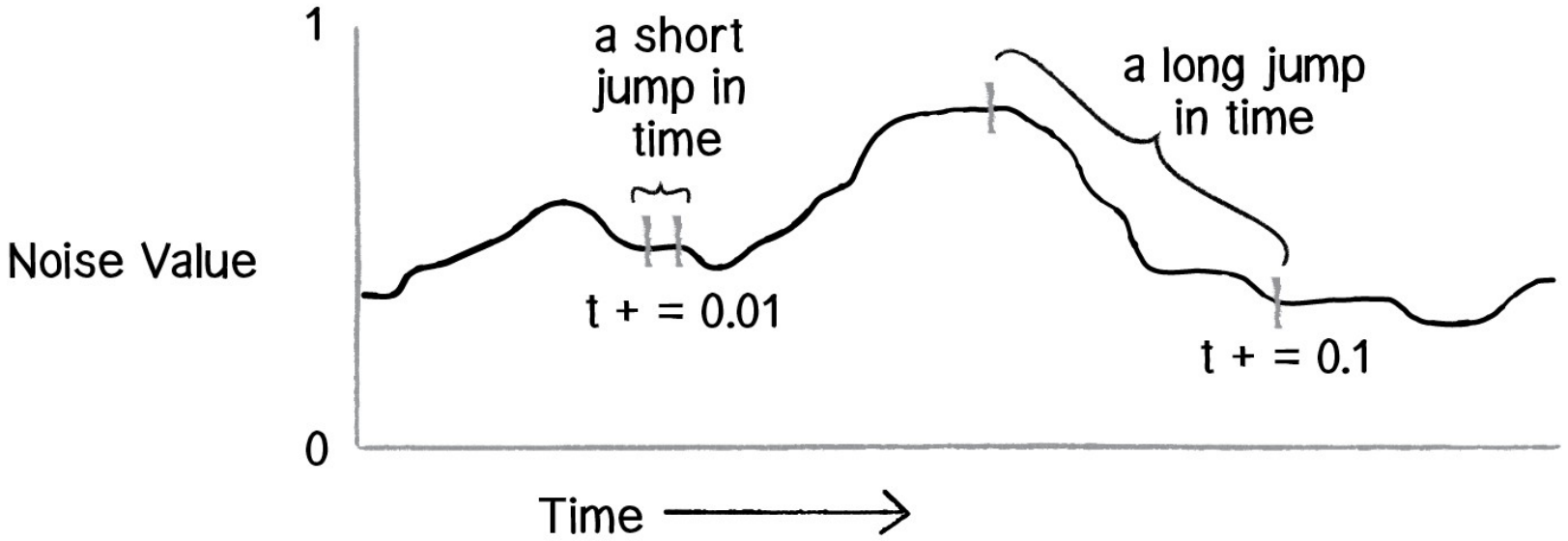
ofNoise()

Time	Noise Value
0	0.365
1	0.363
2	0.363
3	0.364
4	0.366

```
ofNoise(0)   = 0.365
ofNoise(1)   = 0.363
```

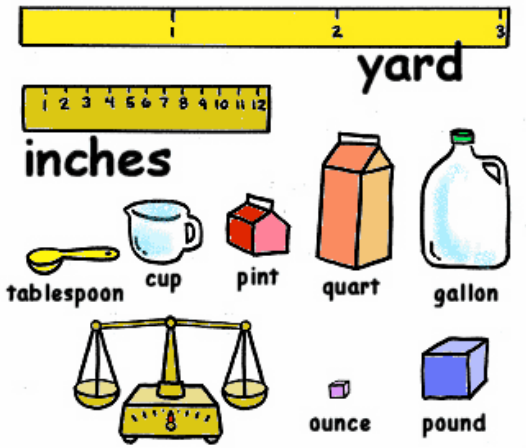
etc...

ofNoise()

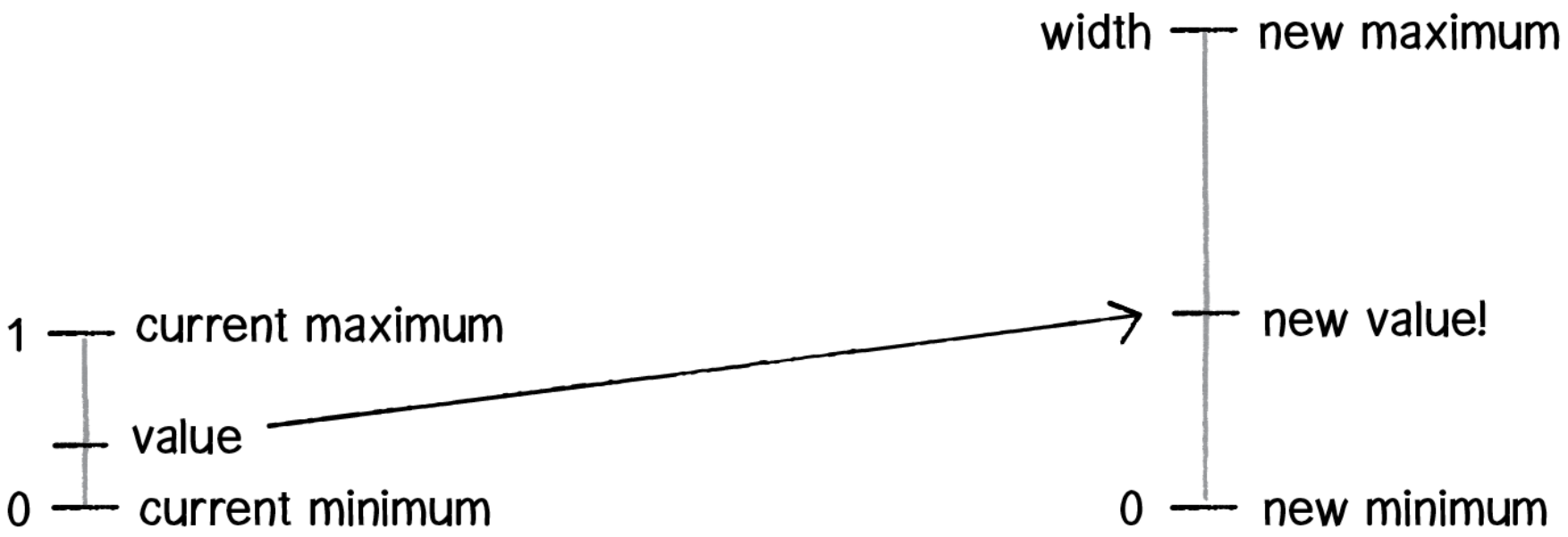


simpleNoise

ofMap()

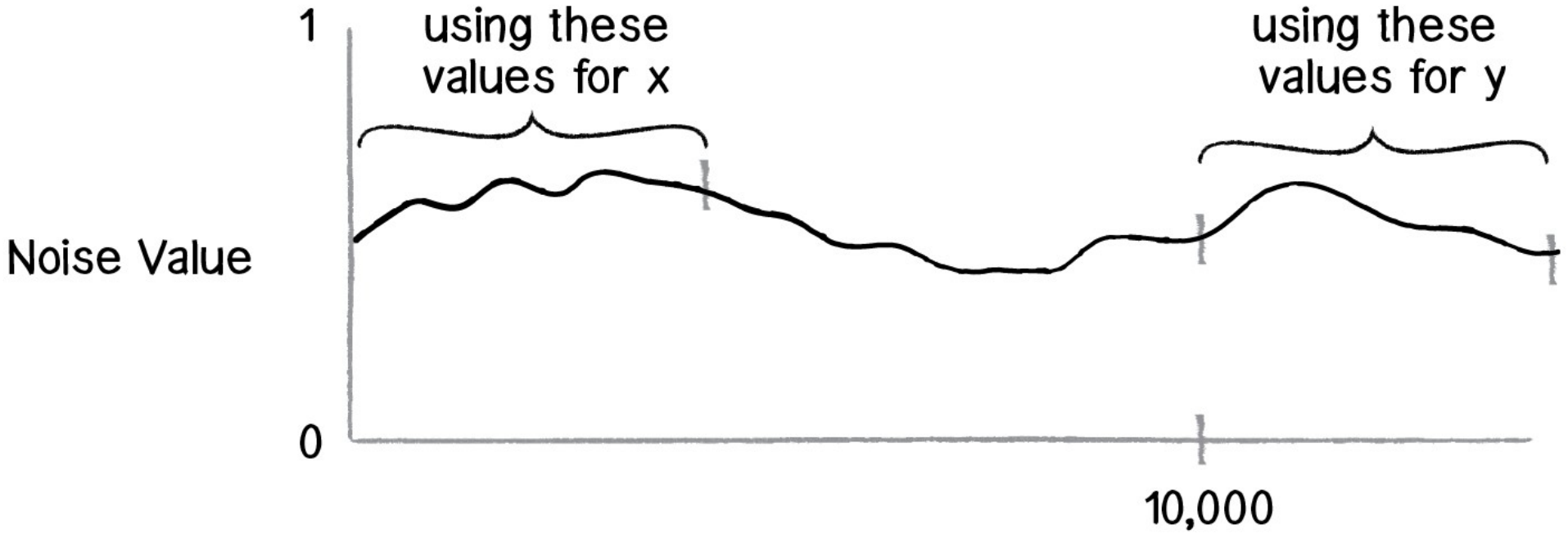


- converting from one unit to another
- `ofNoise()` returns values 0-1

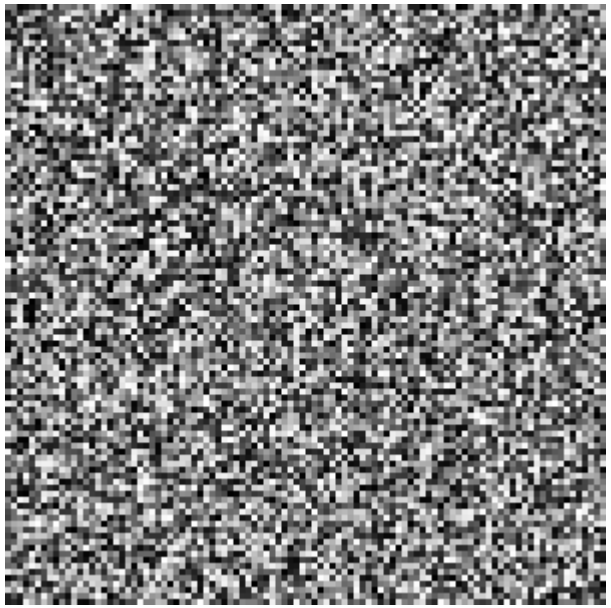


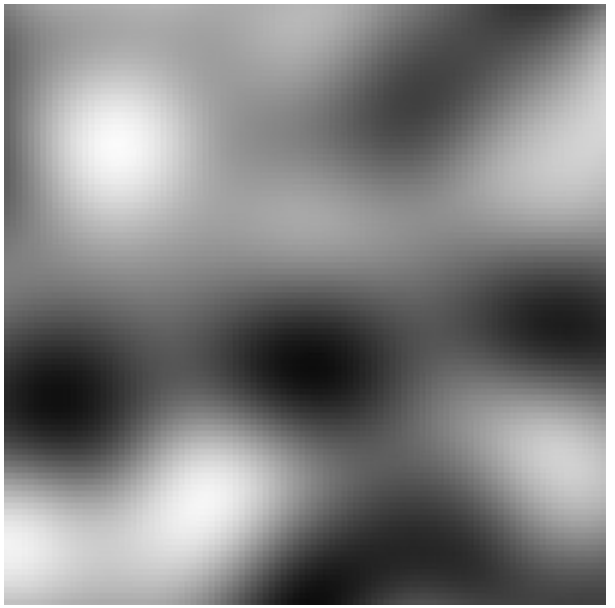

```
newValue = ofMap(value, min1, max1, min2, max2, clampFlag);
```

using different values on noise function



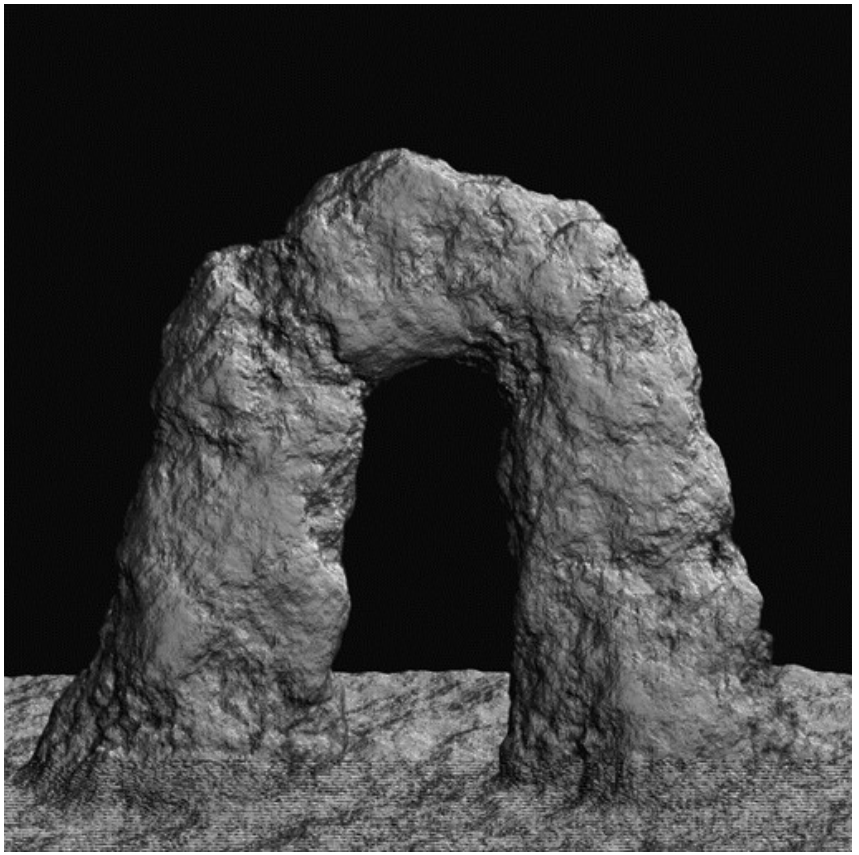
in 2D



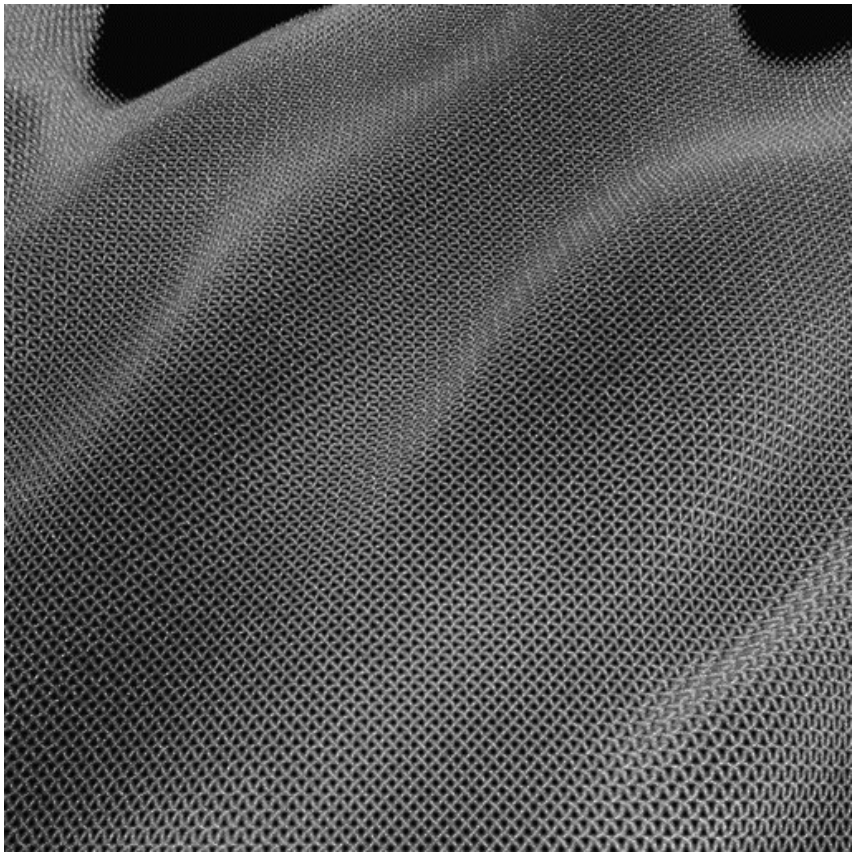


randomPixels
noisePixels

perlin noise - examples

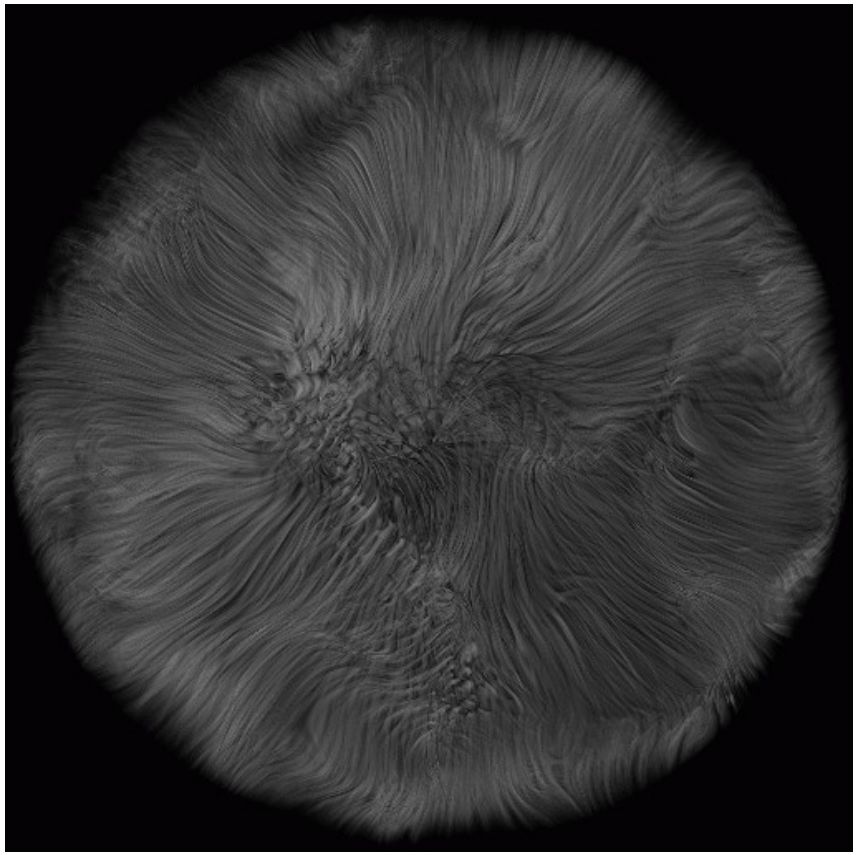


stone

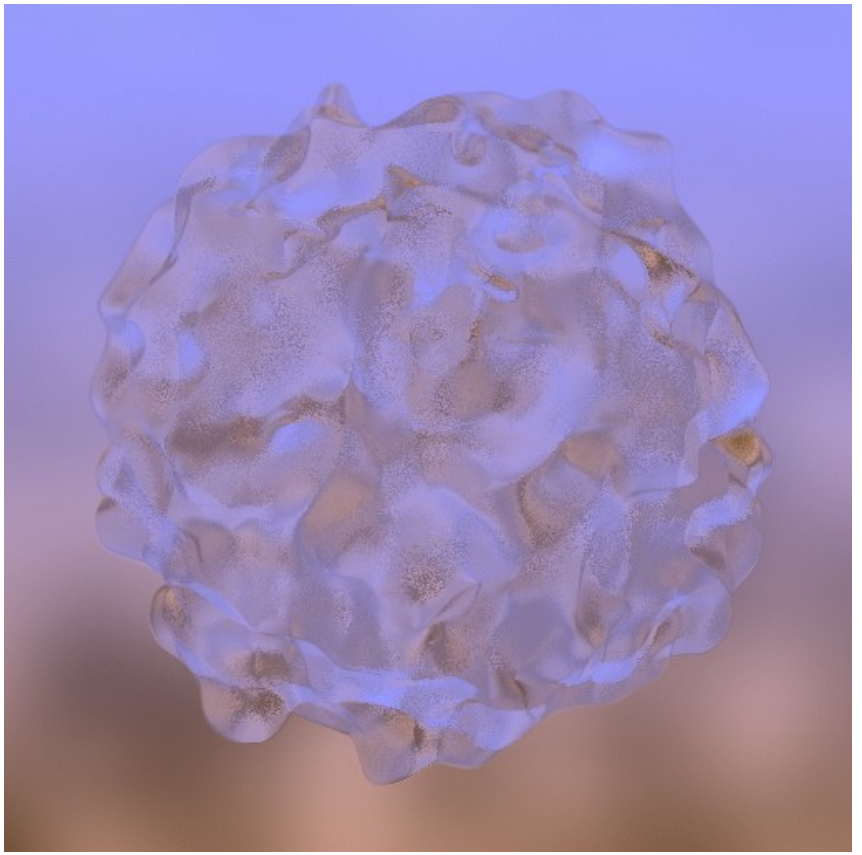


cloth

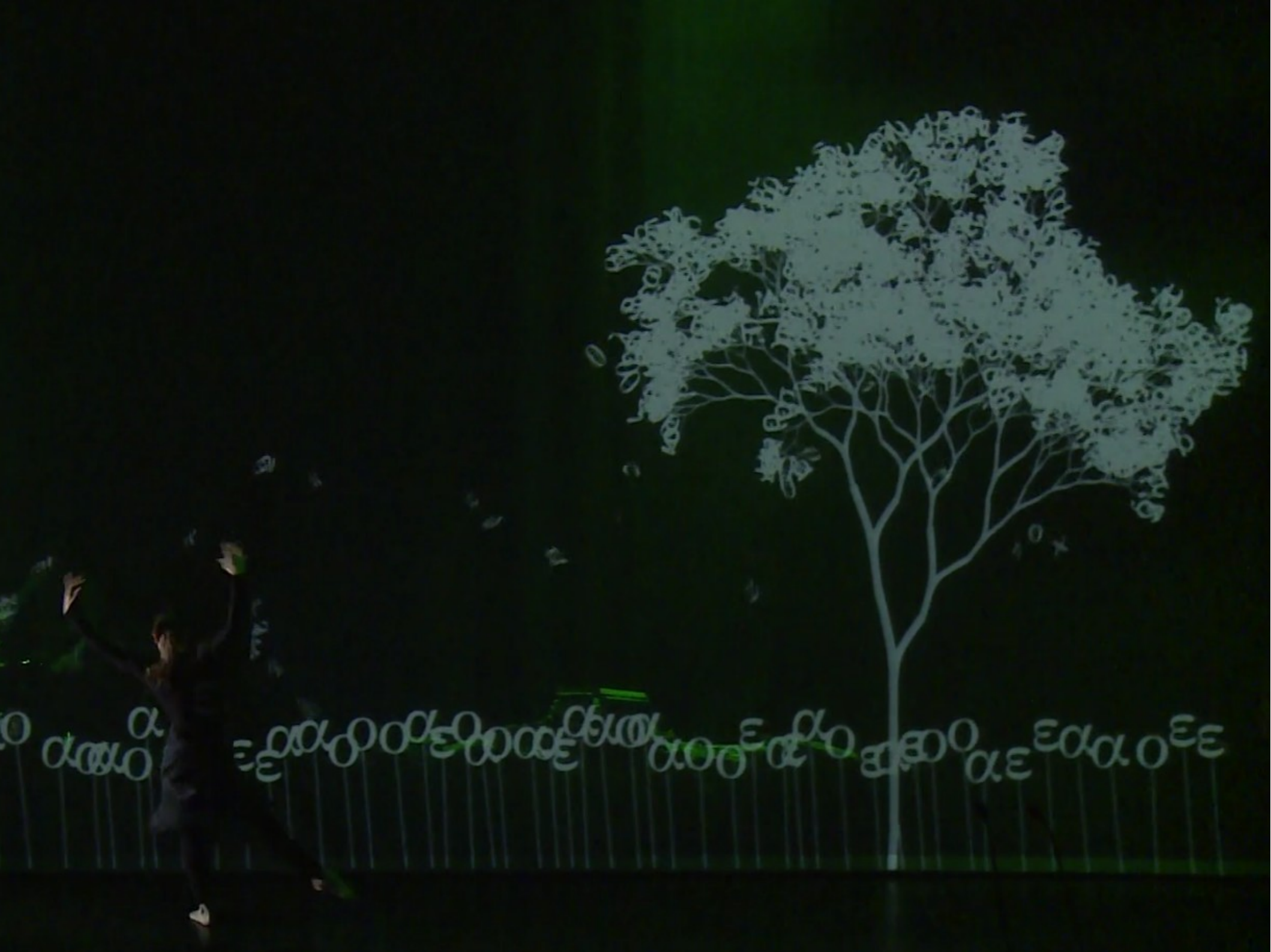
perlin noise - examples

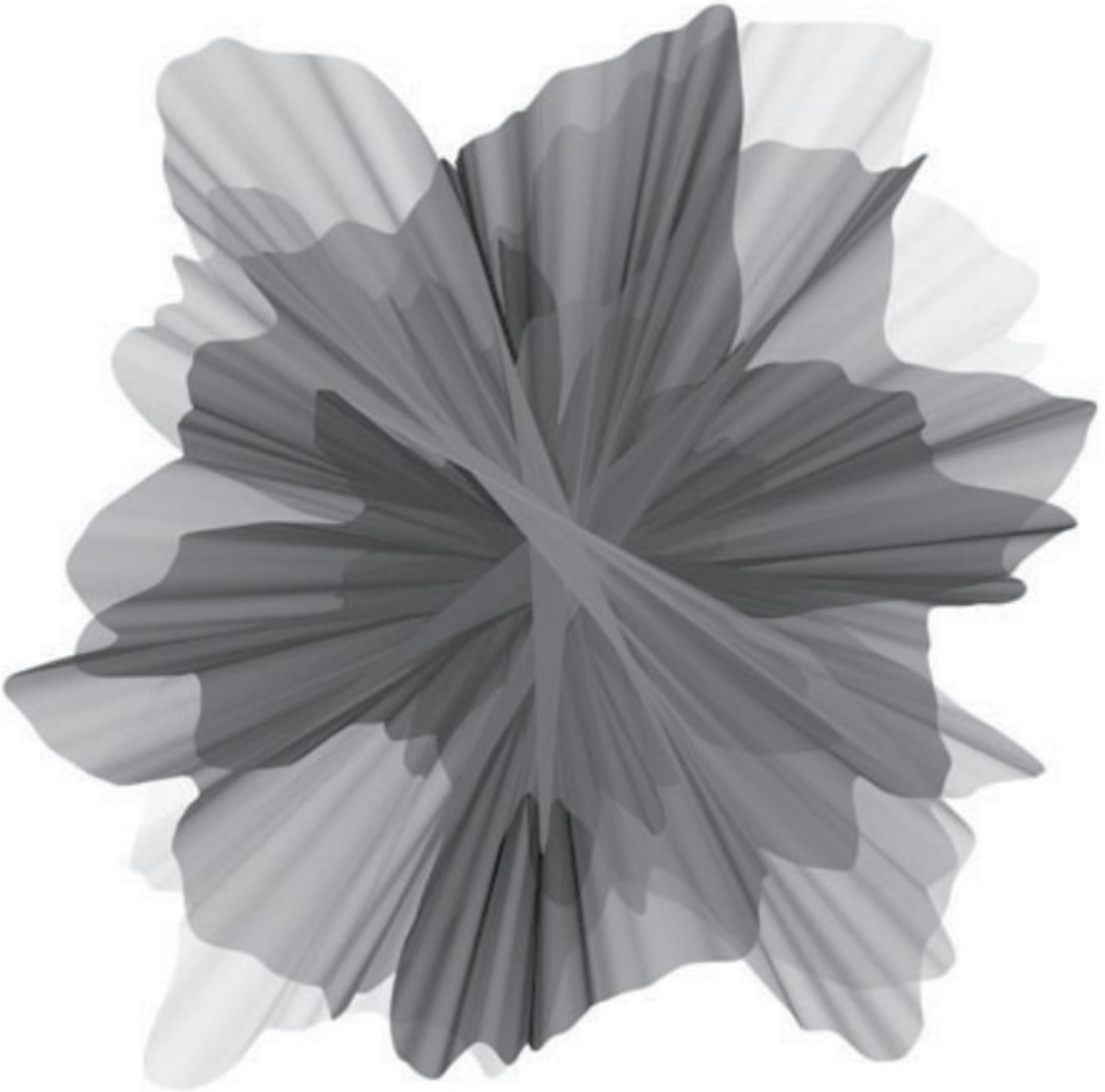


hair



glass

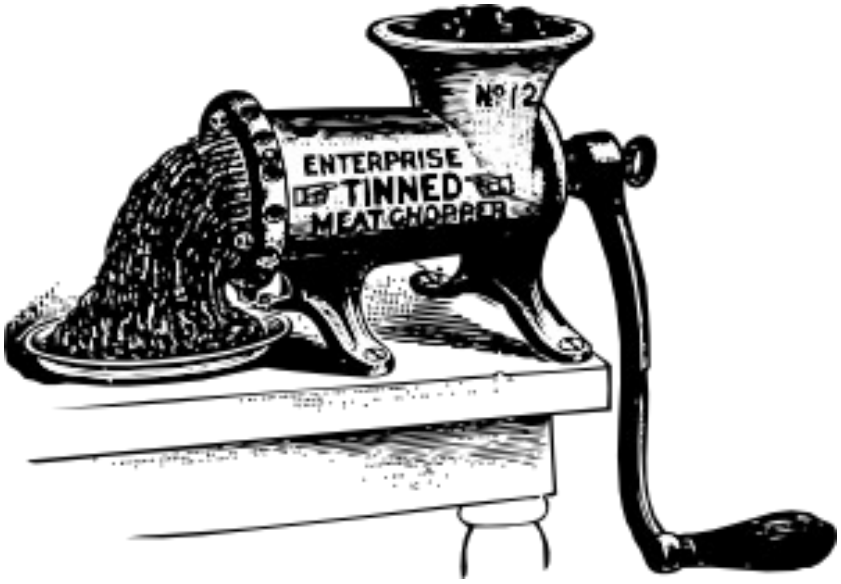




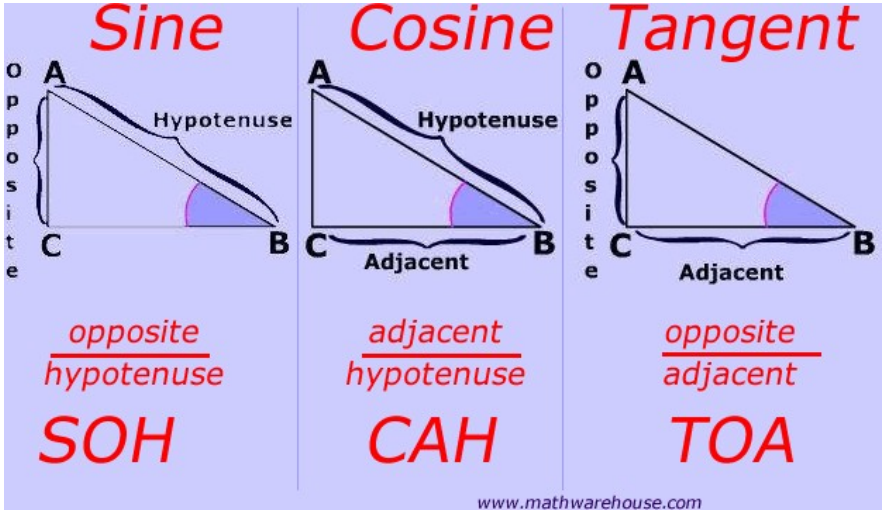

realistic snow

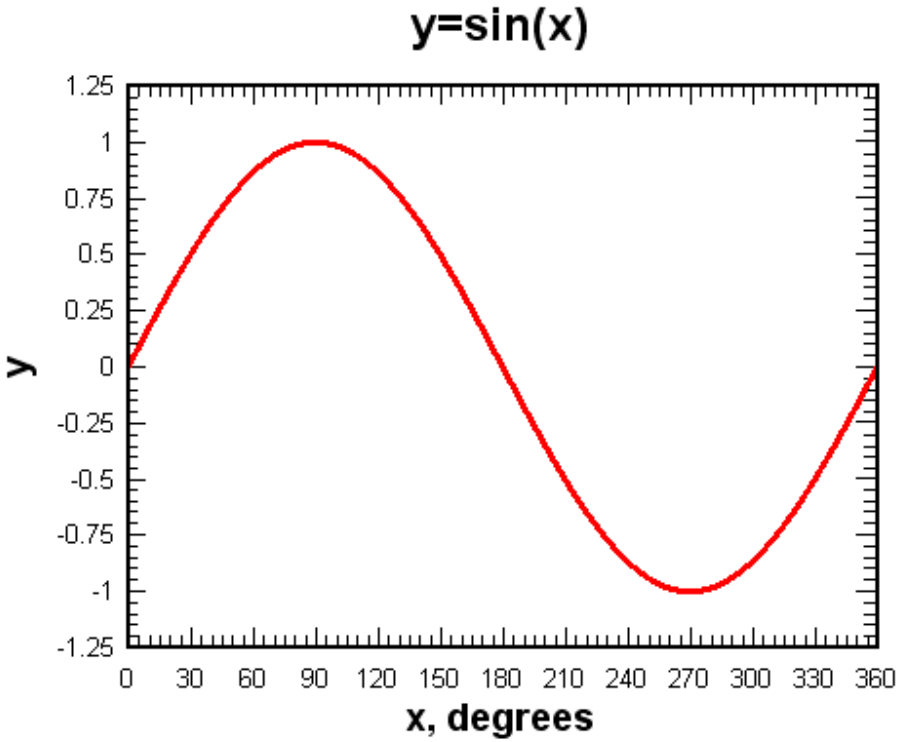
Sine as a number machine

- We feed in linear values from 0 to 360
- We get a smooth sequence of non-linear values
- They cycle between -1 and 1
- allows us to make oscillating objects



sine function





simpleSine

combine sin + cos to make circles

```
x = sin(angleInRadians) * radius;
y = cos(angleInRadians) * radius;
```

```
ofGetFrameNum();  // use as counter
```