Perlin Noise

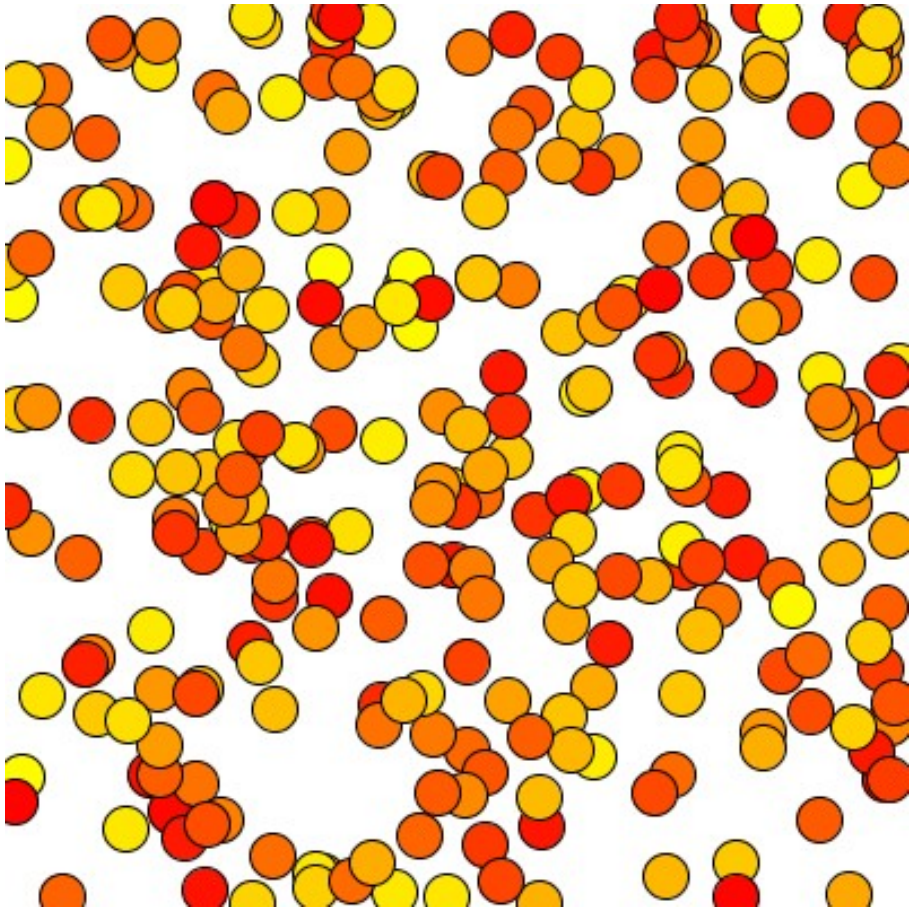
Question

How do you convert the output of a pseudo-random number generator into a smooth, naturalistic function?

Randomness – useful tool

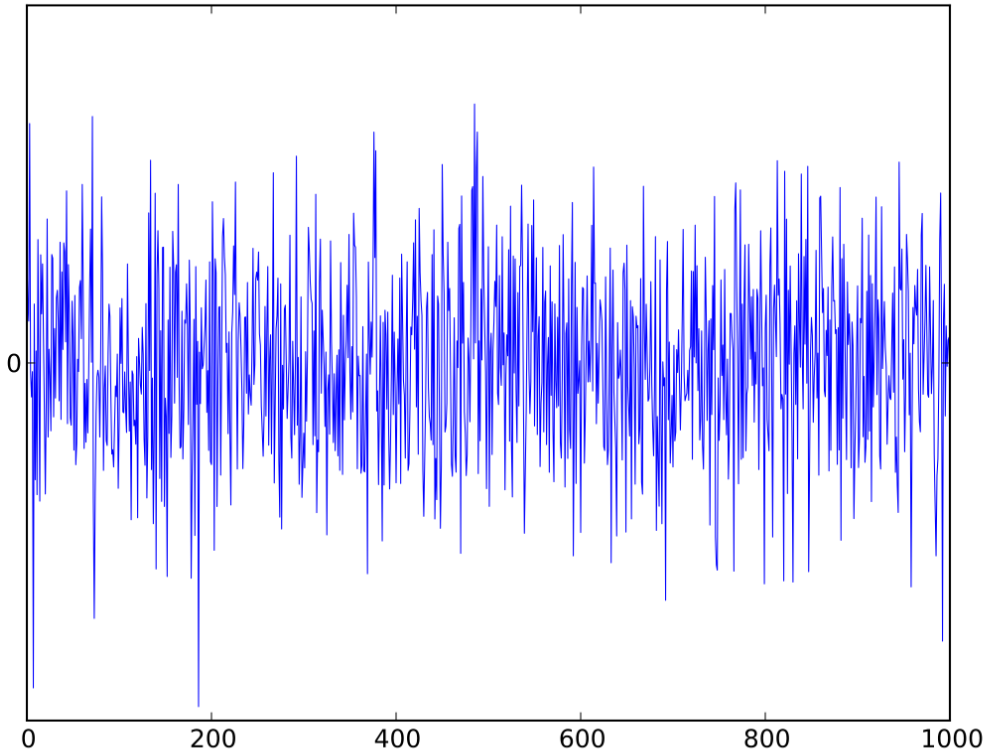
```
// RandomRain
void setup() {
  size(400,400);
  background(255);
  colorMode(HSB,360,100,100);
}

void draw() {
  float x = random(0,400);
  float y = random(0,400);
  float hue = random(0,60);
  fill(hue,100,100);
  ellipse(x,y,20,20);
}
```



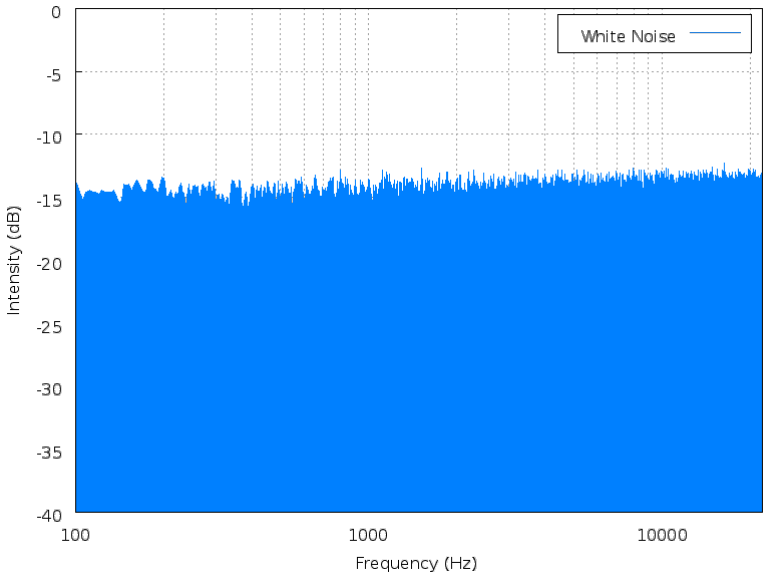
How make it natural and pleasing?

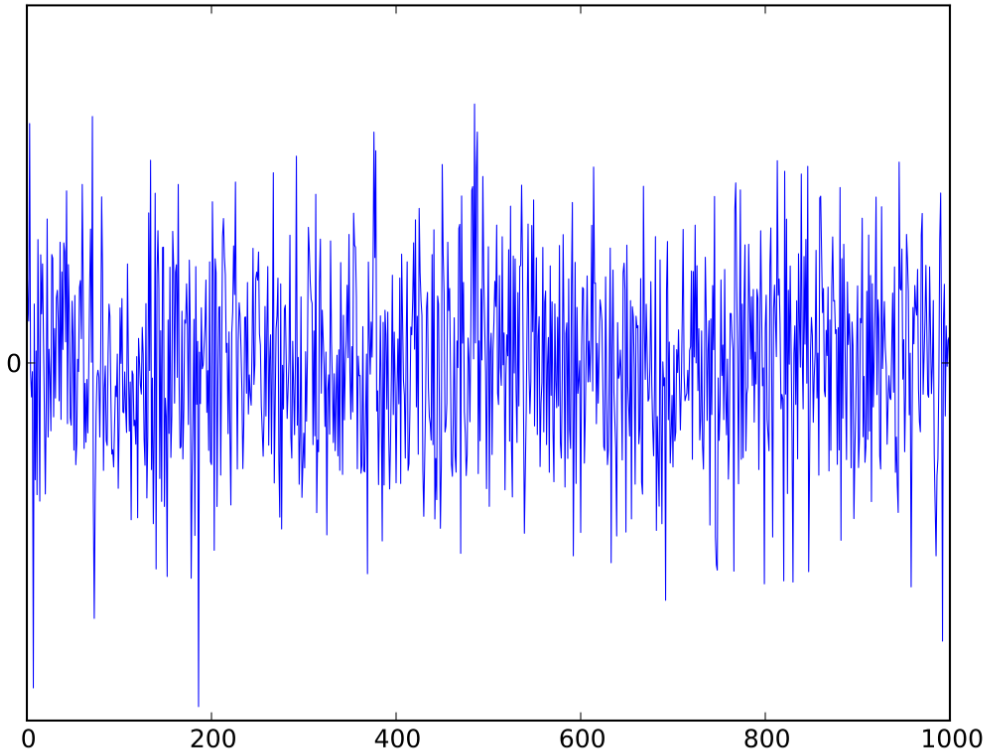
- Pure randomness – white noise
- Each data point independent of rest



White noise

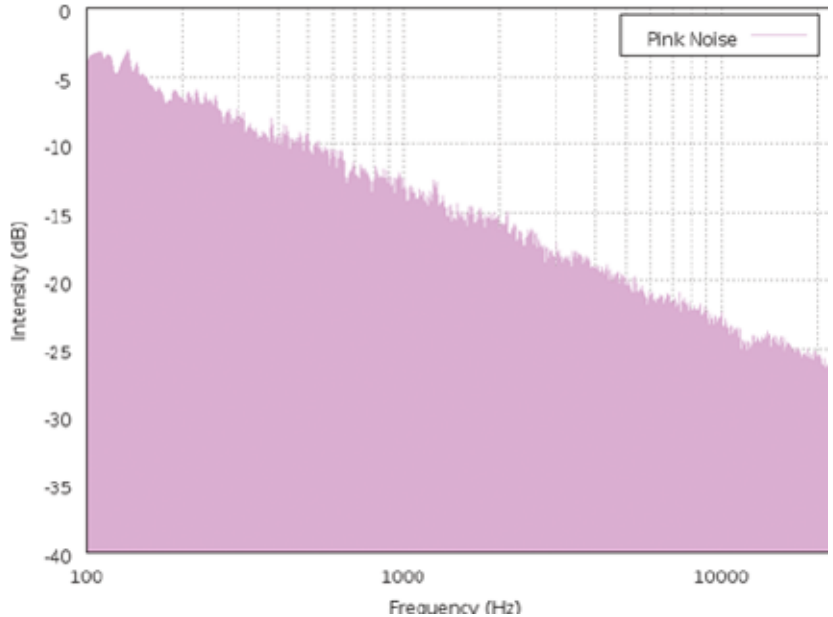
- Pure randomness – white noise
- Each data point independent of rest
- Frequency plot uniform

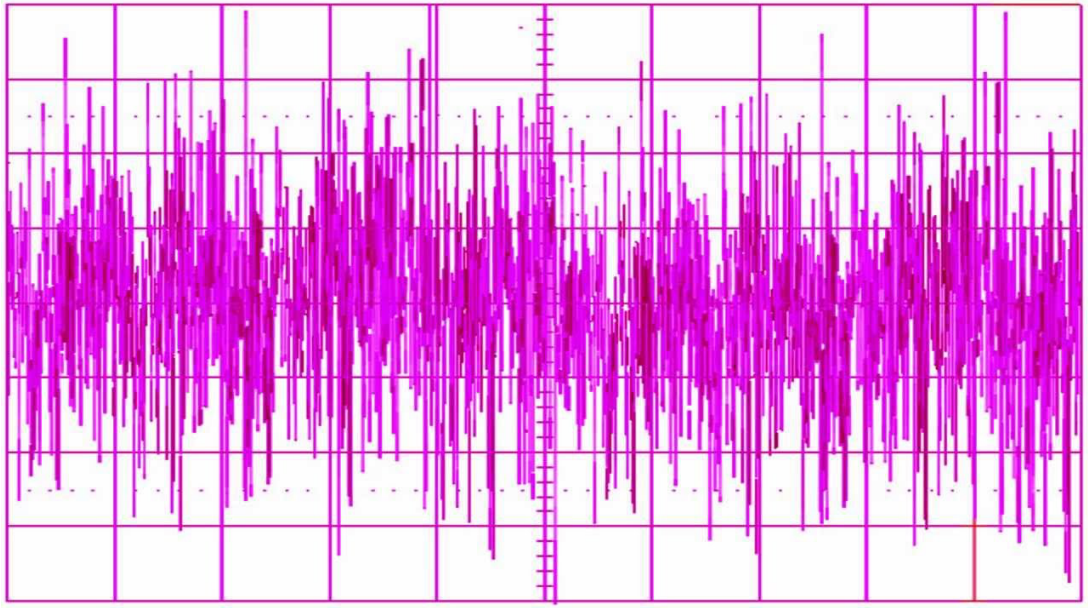




Pink noise

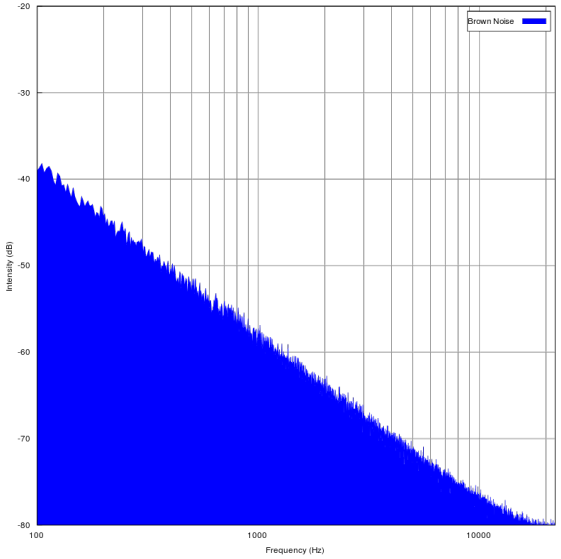
- Shaped randomness – pink noise
- Still independent
- Frequency plot 1/f

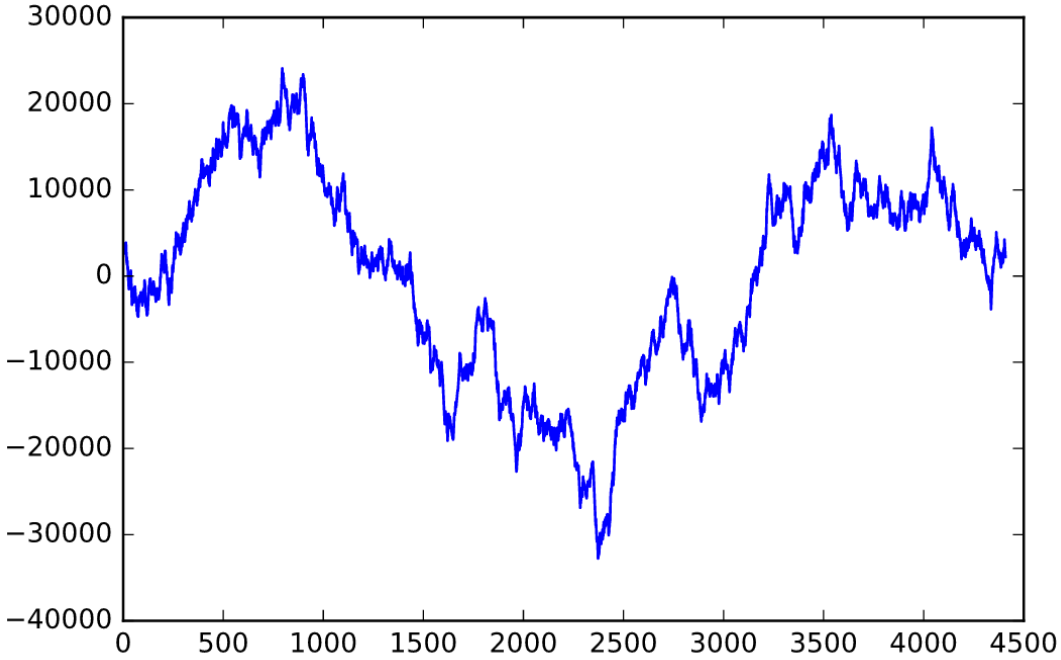




Brown noise

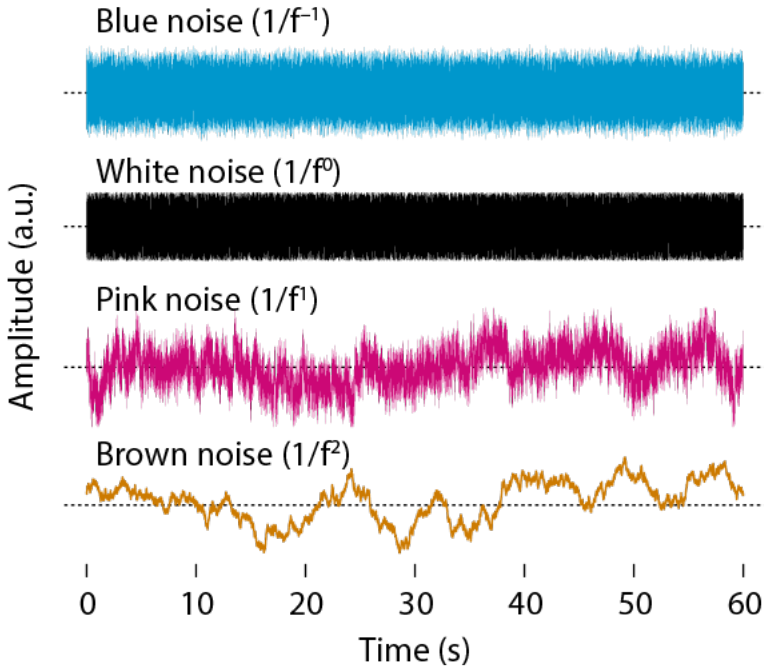
- Random walk – Brownian noise
- Each point random position from last (deltaY = random(-d,d))
- Frequency plot $1/f^2$

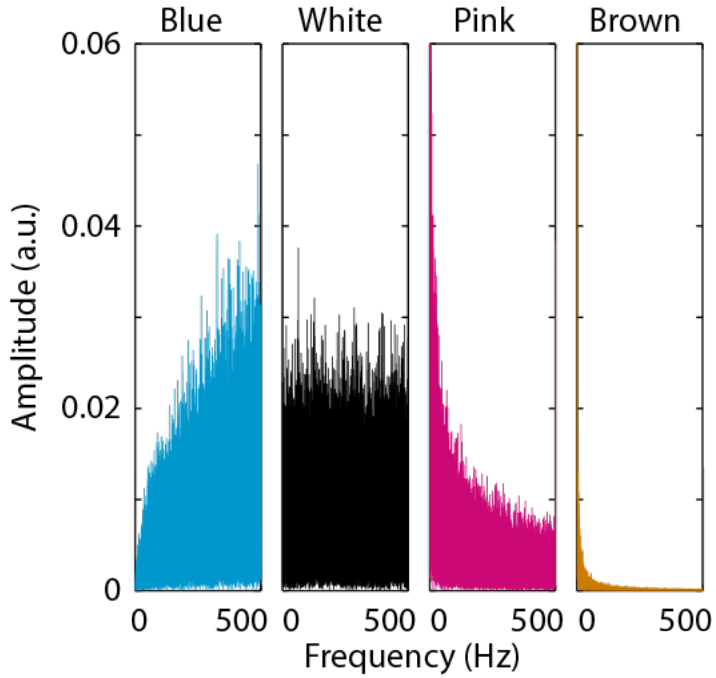




Colors of noise

- Music – close to pink noise 1/f
- Natural objects – close to brown $1/f^2$
- Some physical objects – close to white $1/f^0$
- Model object by estimating frequency spectrum



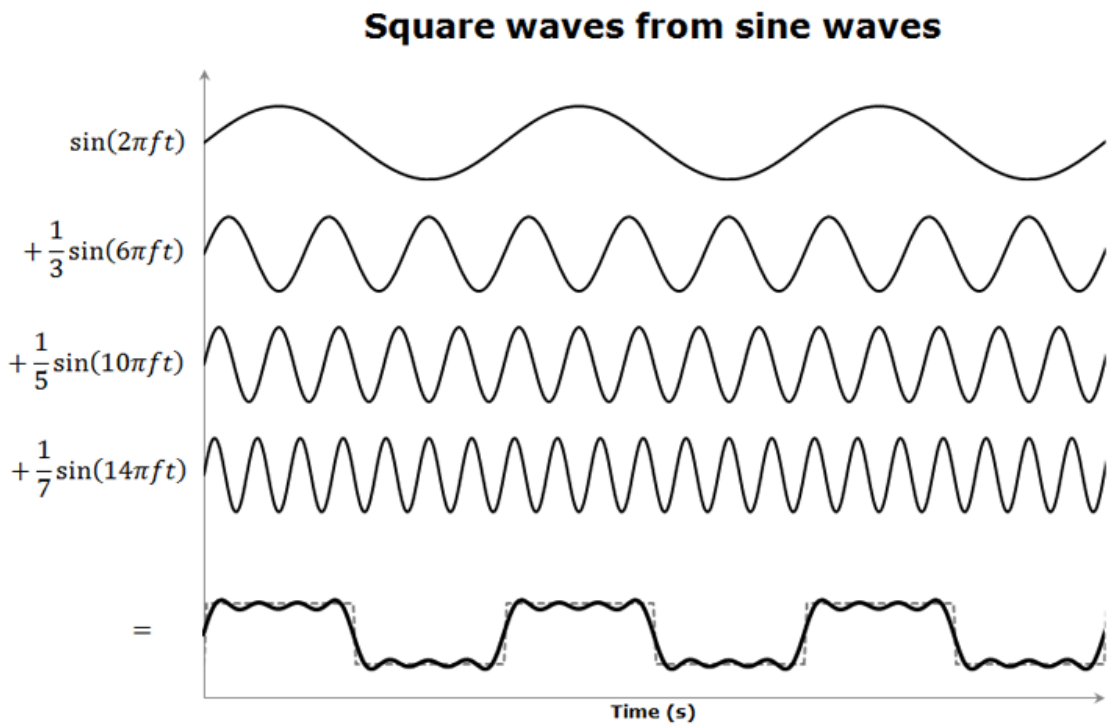


https://archive.org/details/TenMinutesOfWhiteNoisePinkNoiseAndBrownianNoise/BrownianNoise.flac

Generating $1/f^x$ noise

- Fourier Cosine (sine) Series
- Frequency set by *n*

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi x}{L}\right)$$

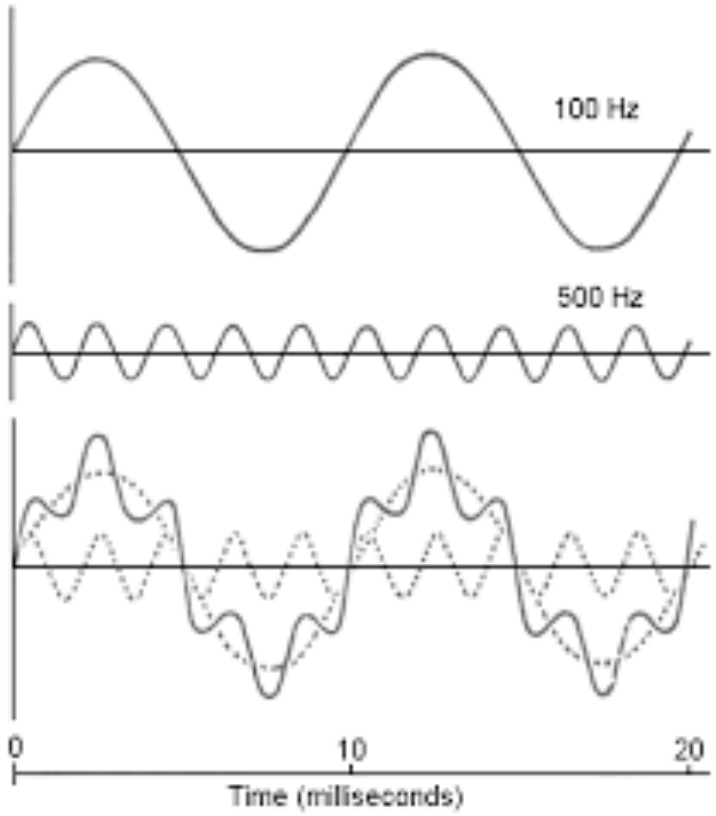


Generating $1/f^x$ noise

- Fourier Cosine (sine) Series
- Frequency set by *n*

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi x}{L}\right)$$

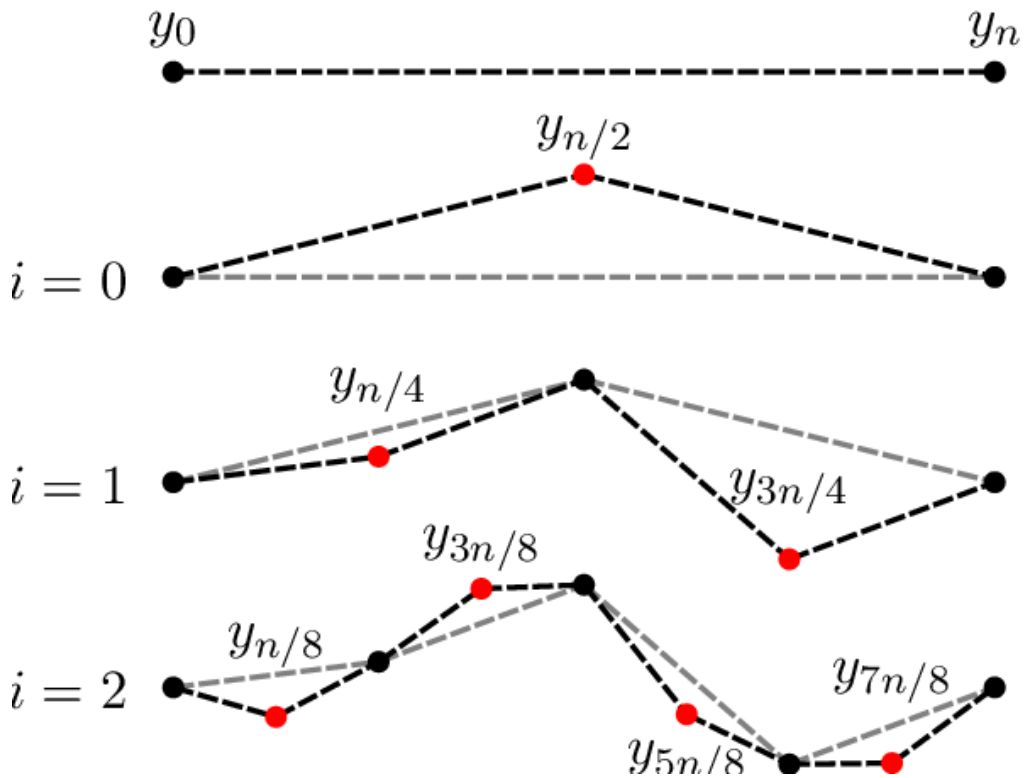
- Generate random terms of frequency, phase
- Decrease amplitude (height) as you increase frequency (n)



More energy higher frequencies => rugged

Application: midpoint displacement

- Recursive curve generation
- Given two points:
  - Create perp bisector
  - Randomly pick t in (-h,h), generate point
  - Repeat for two new line segments
- Works in 3D



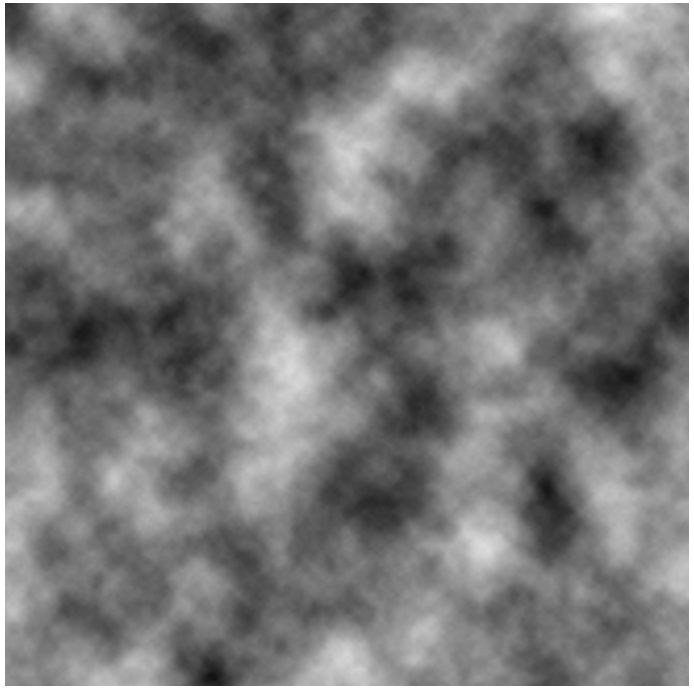
Application: midpoint displacement

- Recursive curve generation
- Given two points:
  - Create perp bisector
  - Randomly pick t in (-h,h), generate point
  - Repeat for two new line segments
- Works in 3D

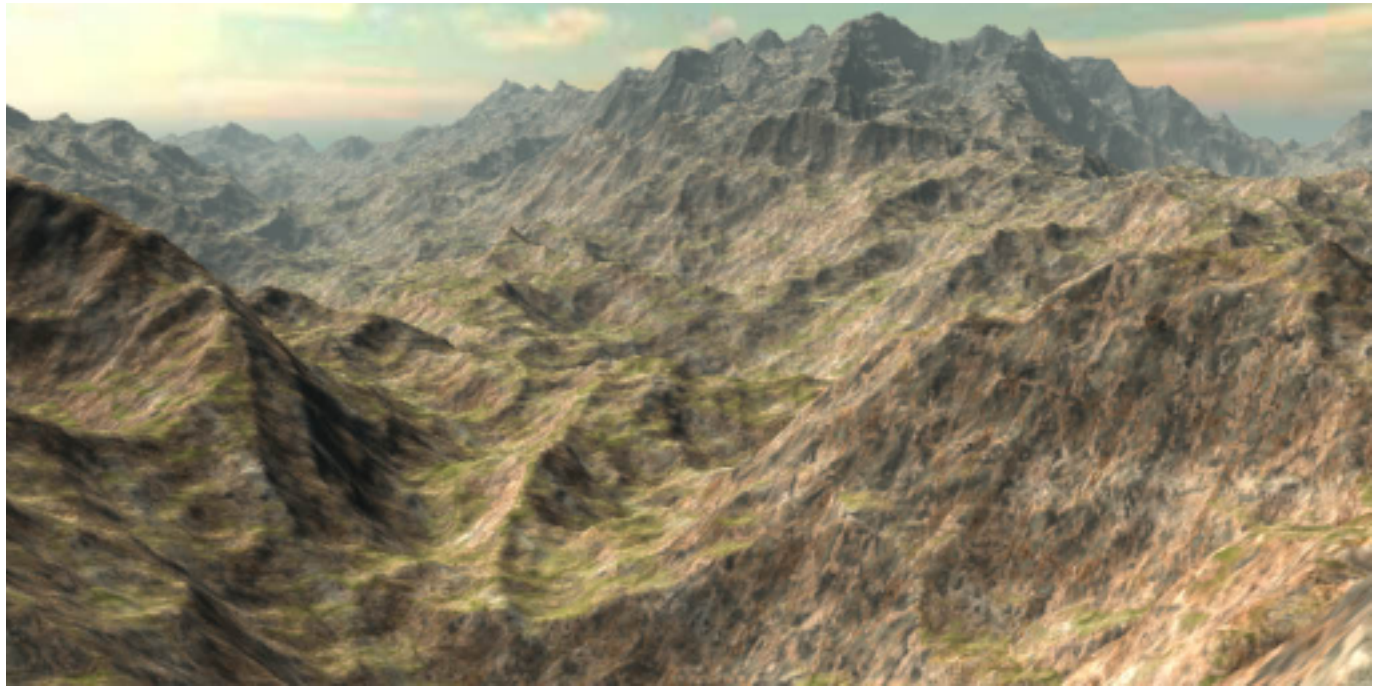
- Question
- How would you tune midpoint displacement to get more or less rugged landscapes?

Perlin noise

- Ken Perlin 1983
- (a) height map (b) resulting landscape



(a)



(b)

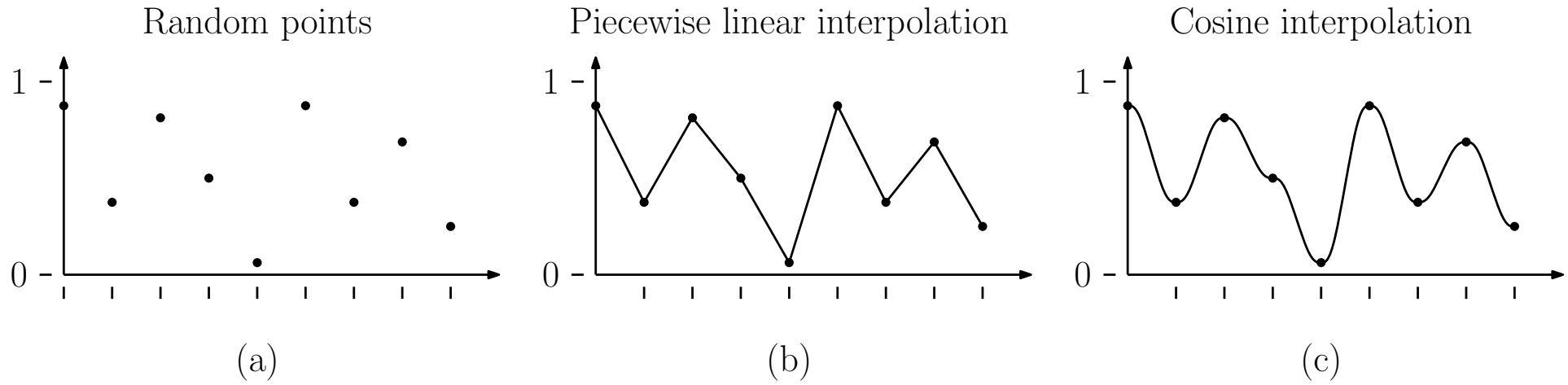
Perlin noise

- Ken Perlin 1983
- Vary frequency component => control ruggedness

Noise fcn f(x) - interpolating random points

- Generate series $Y = \langle y_0, y_1, y_2, \ldots, y_n \rangle$
  at uniformly placed $X = \langle x_0, x_1, x_2, \ldots, x_n \rangle$

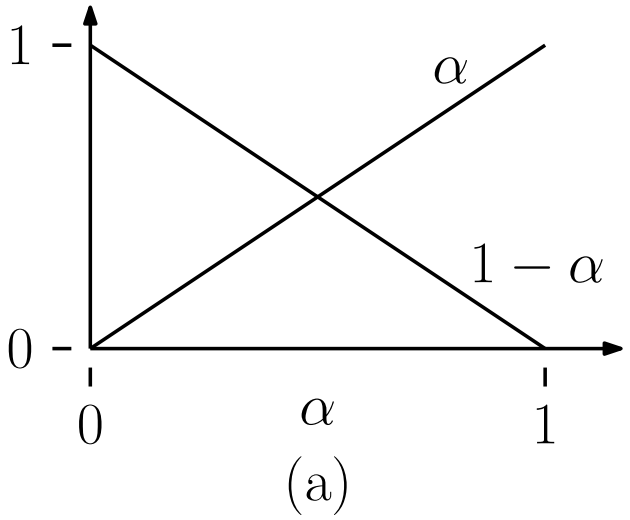
$$f_\ell(x) = \mathrm{lerp}(y_i, y_{i+1}, \alpha), \qquad \text{where } i = \lfloor x \rfloor \text{ and } \alpha = x \bmod 1$$



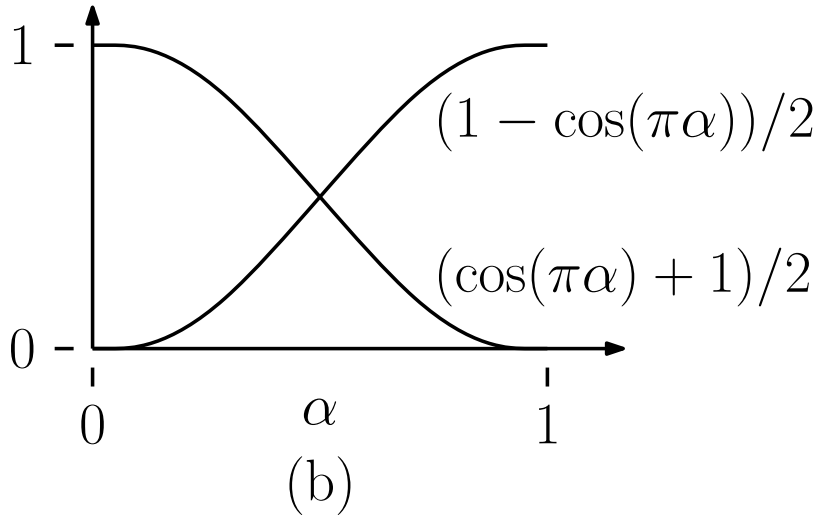
Interpolating weight functions

- Generate series $Y = \langle y_0, y_1, y_2, \ldots, y_n \rangle$ at uniformly placed $X = \langle x_0, x_1, x_2, \ldots, x_n \rangle$

$$f_\ell(x) = \text{lerp}(y_i, y_{i+1}, \alpha), \qquad \text{where } i = \lfloor x \rfloor \text{ and } \alpha = x \bmod 1$$



(a)



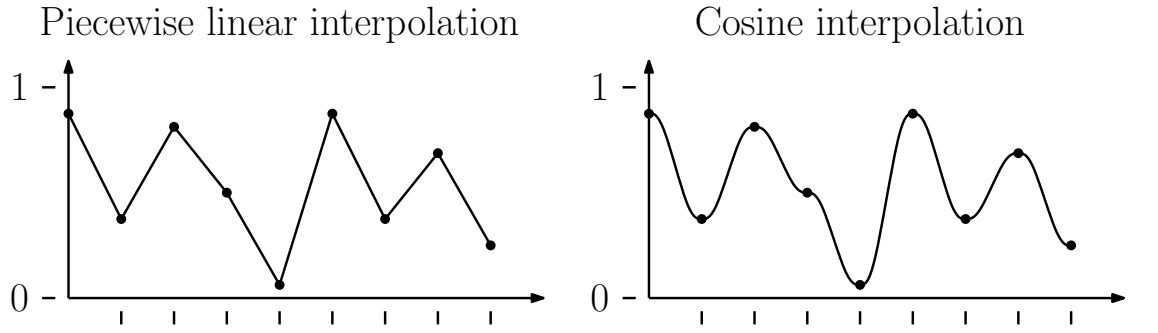
(b)

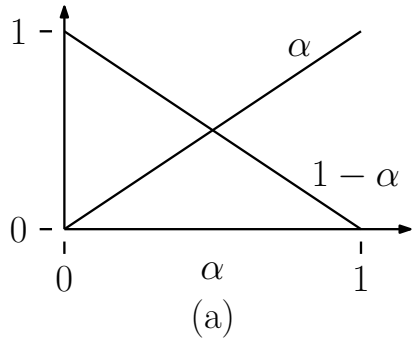
Interpolating weight functions

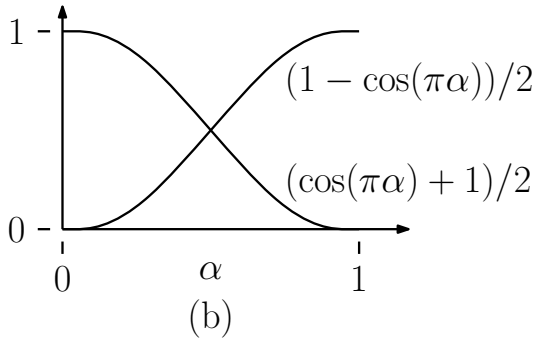
Cosine – smoother because

Slower to leave p0

Faster to arrive at p1

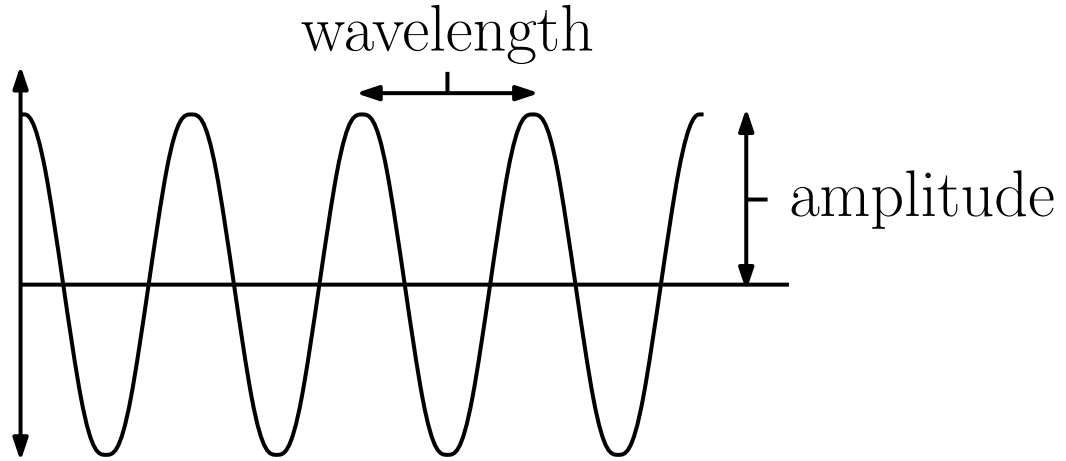






$\alpha \sin(\omega t)$

- ***Wavelength***: The distance between successive wave crests
- ***Frequency***: The number of crests per unit distance, that is, the reciprocal of the wavelength
- ***Amplitude***: The height of the crests



- α – amplitude
- ω – frequency
- $2\pi/\omega$ – wavelength

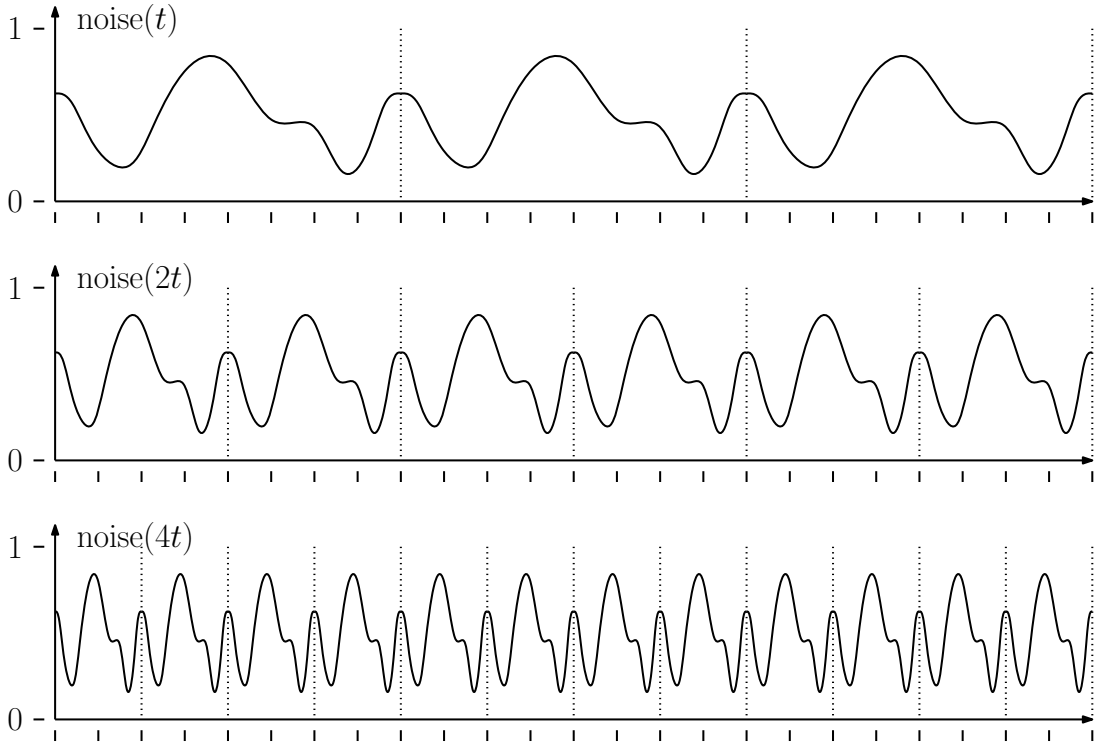
Periodic noise function

- $f(x)$ defined on range [0,n]
- With $f(0) = f(n)$

- Now define

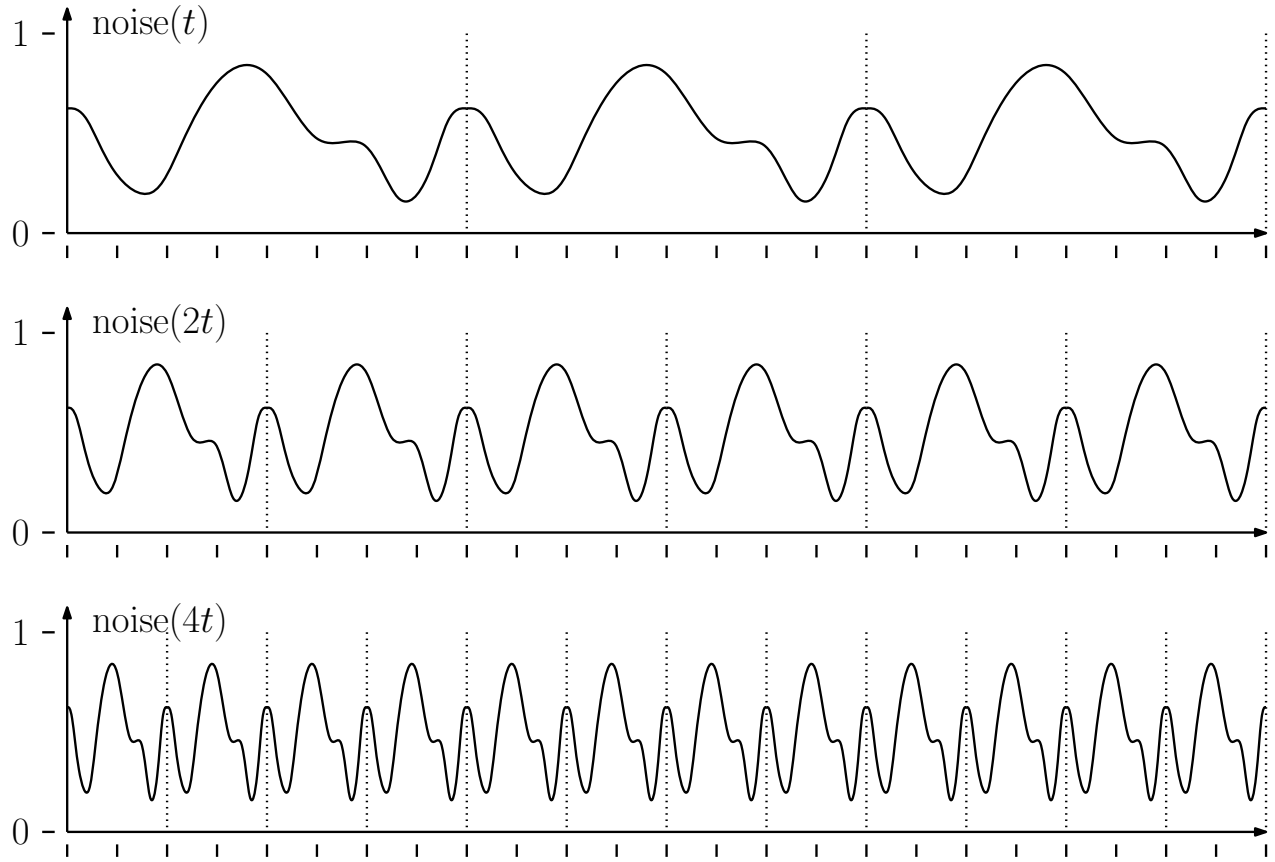
- $\text{noise}(t) = f(t \bmod n)$

- *Not sine* – randomly created
- Same curve – self-similar



Frequency octaves

- $\text{noise}(t)$
- $\text{noise}(2t)$
- $\text{noise}(4t)$
- …
- $\text{noise}(2^i t)$

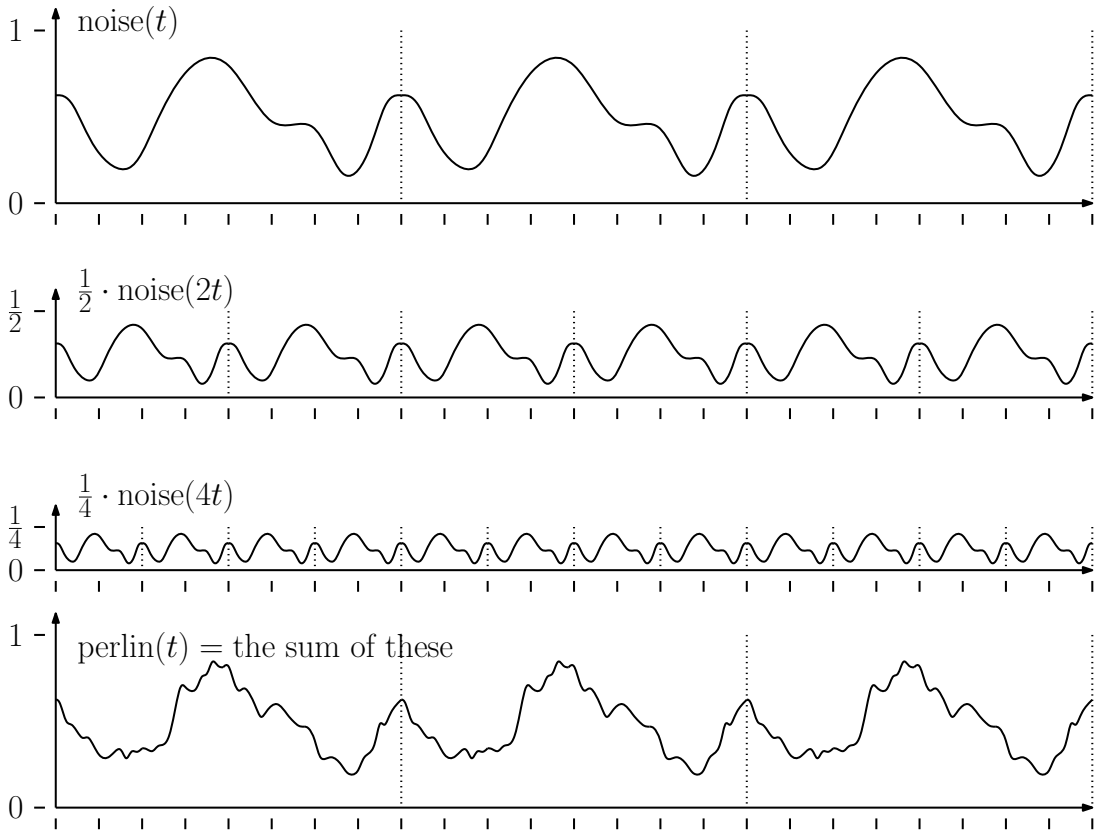


Persistence

- $p^0\text{noise}(t)$
- $p^1\text{noise}(2t)$
- $p^2\text{noise}(4t)$
- …
- $p^i\text{noise}(2^i t)$

$$p = \frac{1}{2}$$

$$\text{perlin}(t) = \sum_{i=0}^{k} p^i\text{noise}(2^i t)$$



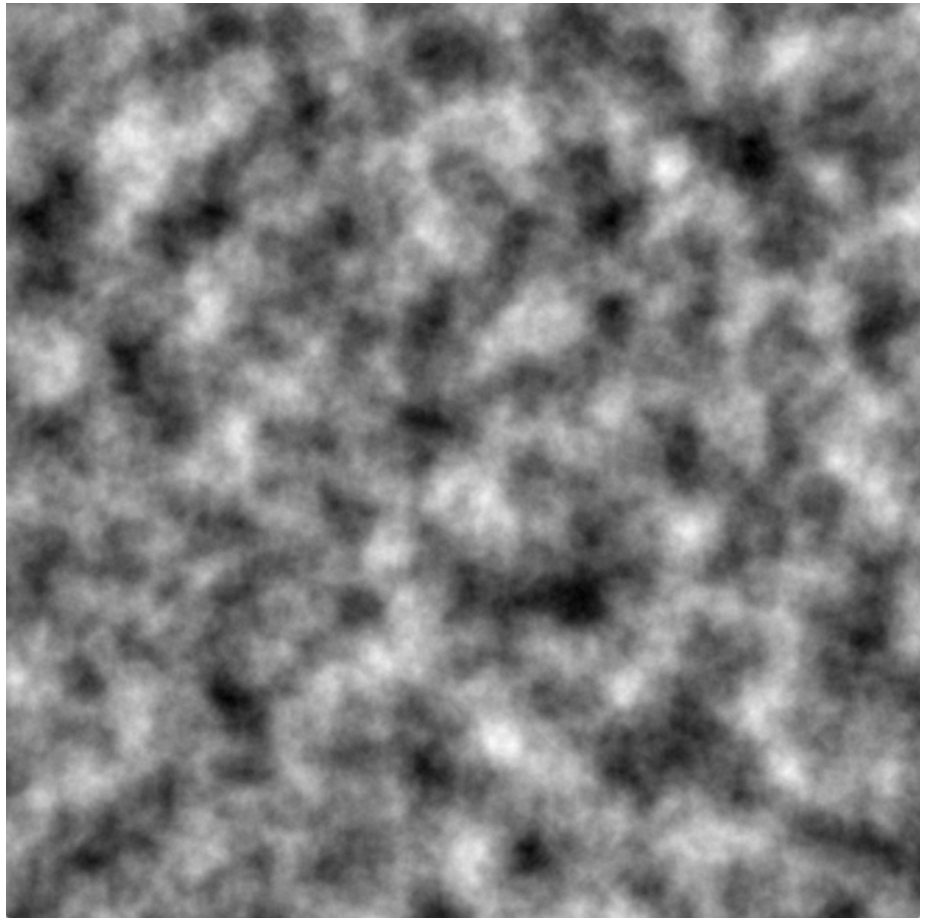
Perlin noise summary

- Perlin noise is
  - Constant after generation
  - Periodic
  - Fractally self-similar

- Unity
  public static float **PerlinNoise**(float **x**, float **y**);

  returns value in [0,1.0]

  (Set y = constants to get 1D function)



https://cpetry.github.io/TextureGenerator-Online/

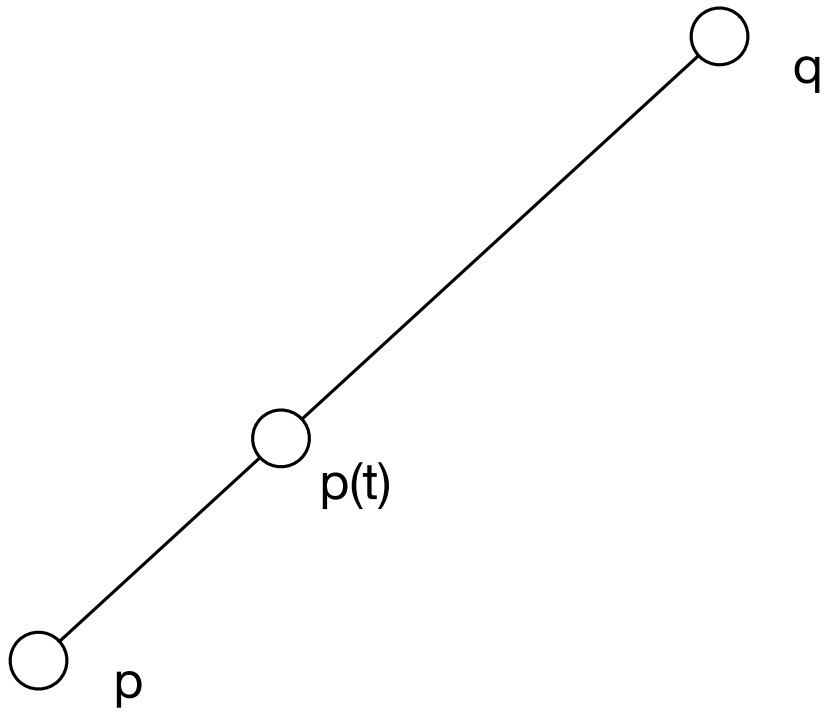
Perlin noise in 2D?

Parametric line segments

$$p(t) = p + tv$$

with $v = q - p$

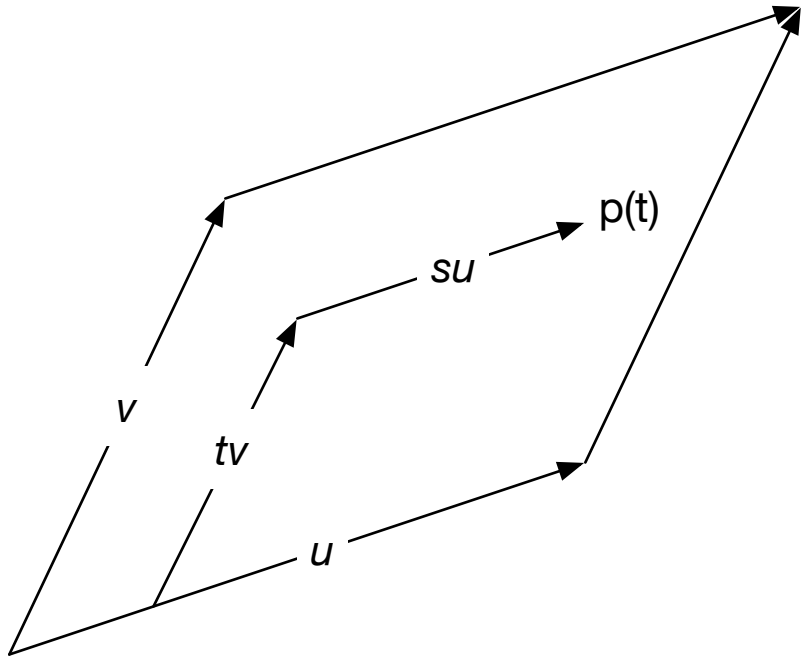
```
for t = 0 to 1 by deltat
     x = px + t * vx
     y = py + t * vy
     plot(x,y) // or line(lx,ly,x,y)
```



Parametric planar patches

$$p(s,t) = p + tv + su$$

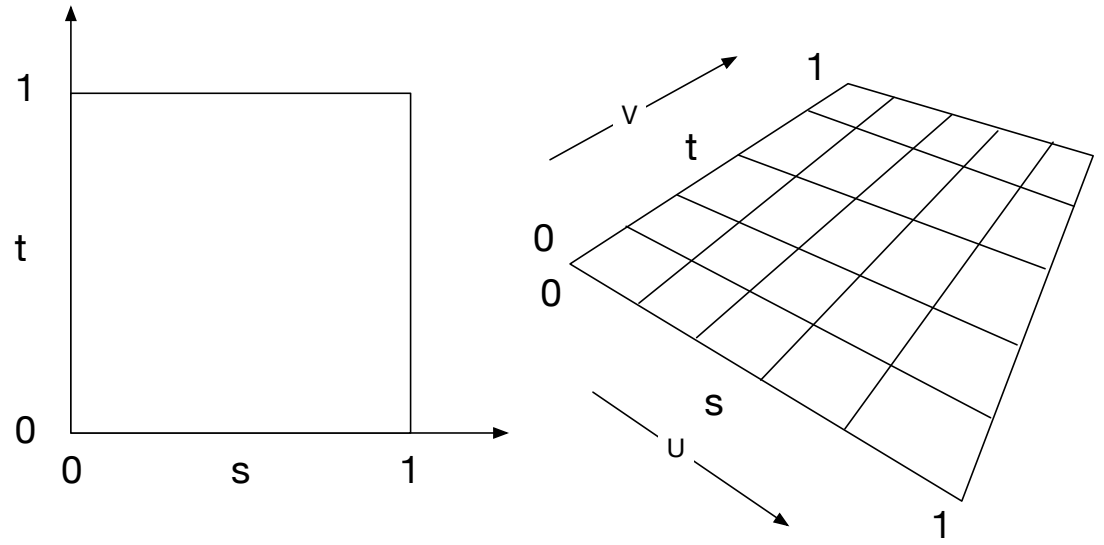
```
for t = 0 to 1 by deltat
     x = px + t * vx + s * ux
     y = py + t * vy + s * uy
     z = pz + t * vz + s * uz
     plot(x,y,z)
```



Creating planar mesh

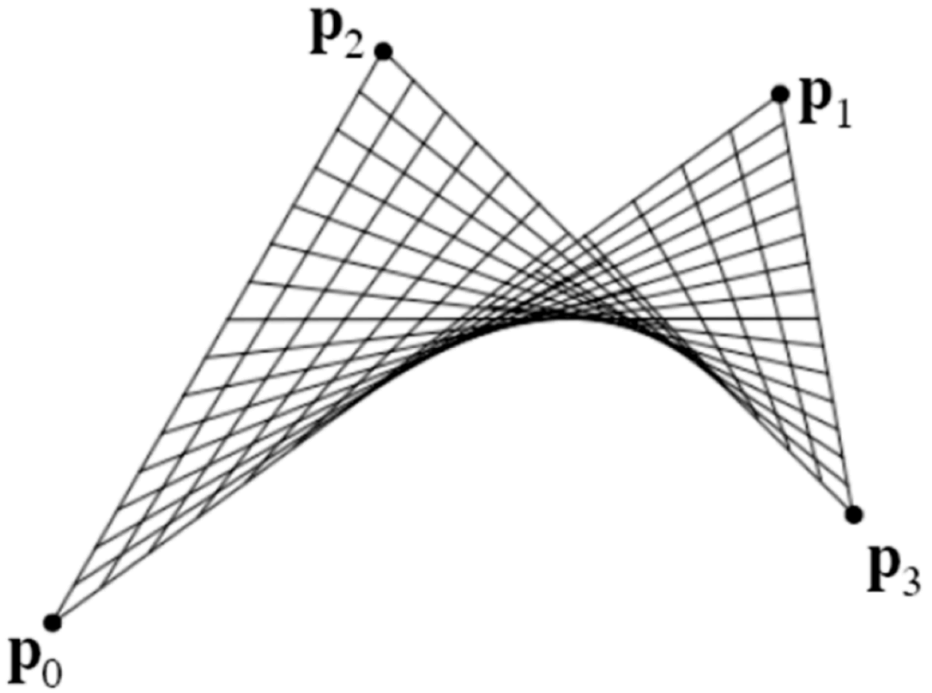
- How create mesh data structure from parametric patch?

- List of vertices
- List of edges
- List of faces



Bilinear patches and interpolation

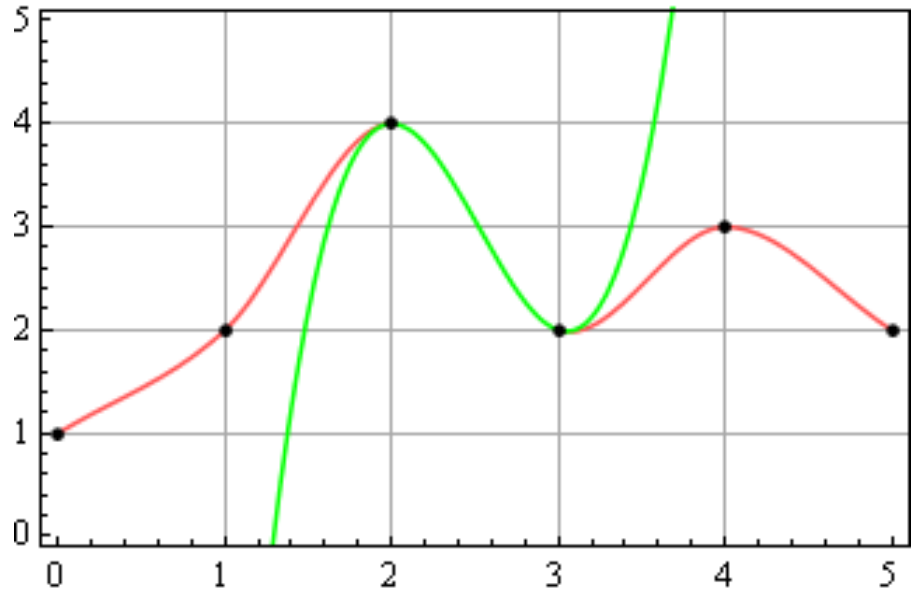
- Interpolation of four points
  - May not be co-planar
- Ruled surface – swept out by straight line
- Will develop equations in class



Cubic interpolation

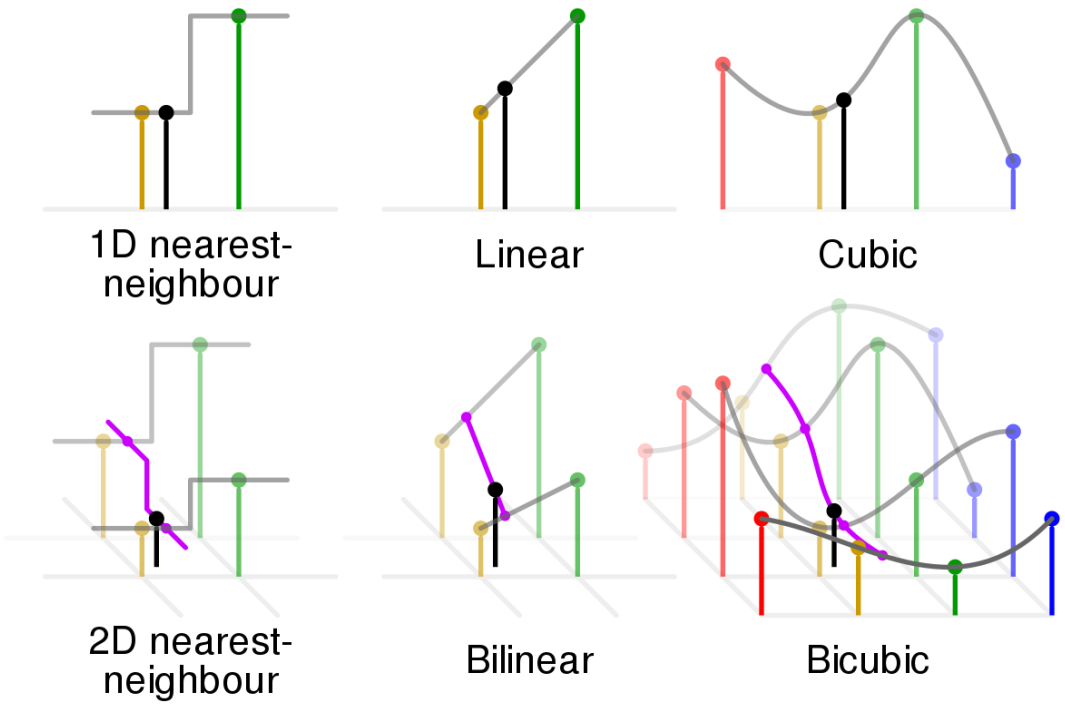
- $P(t) = ax^3 + bx^2 + cx + d$

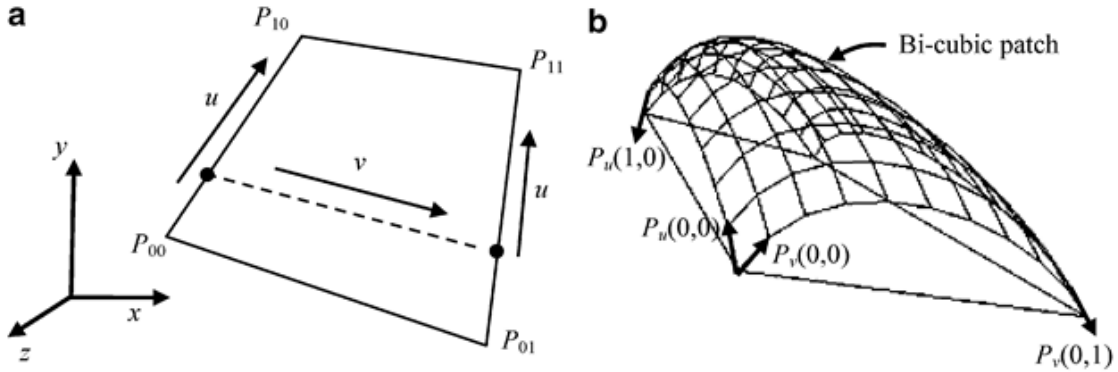
- Can match tangents at ends
- Good enough for human eye

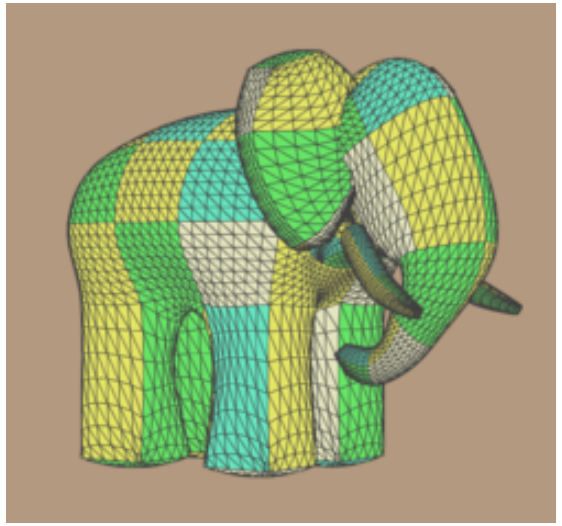


Bicubic surface patch

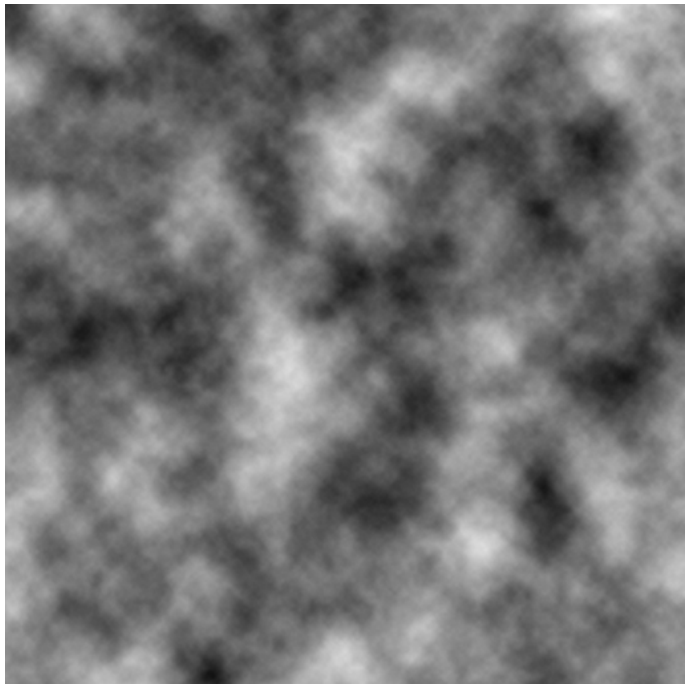
- Cubic curve in both directions



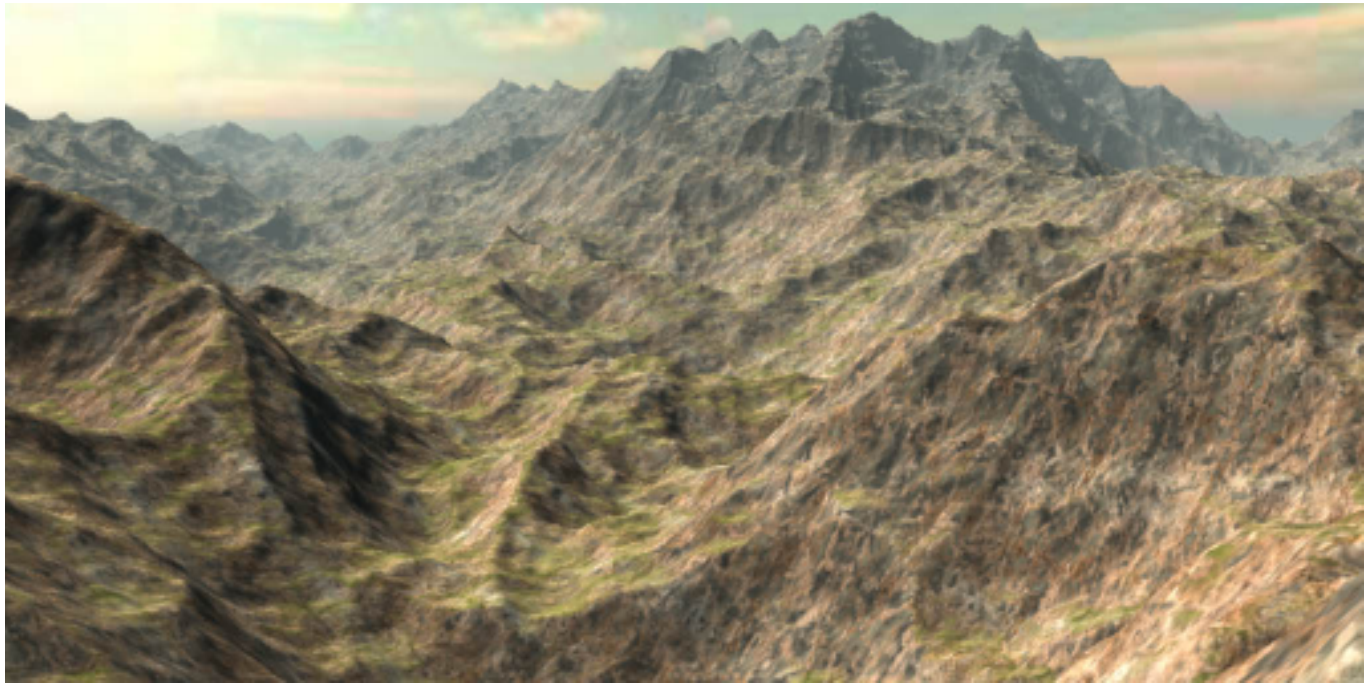




2D Perlin noise



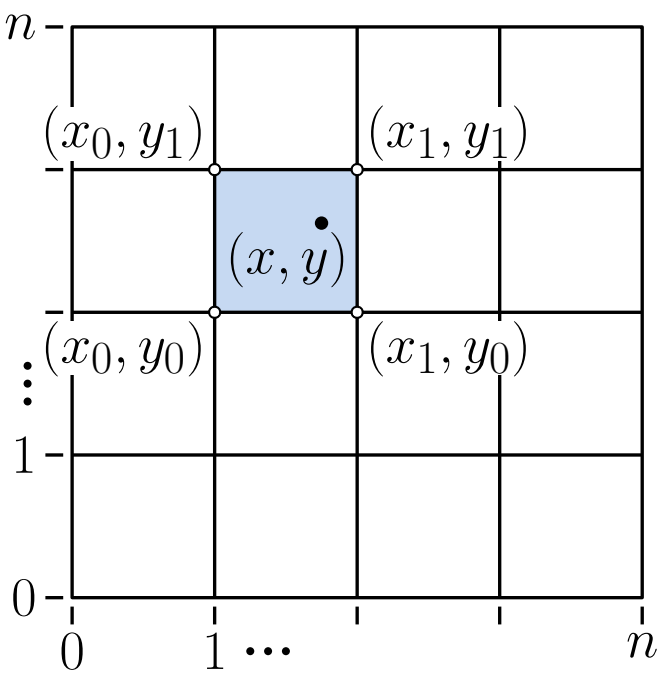
(a)



(b)

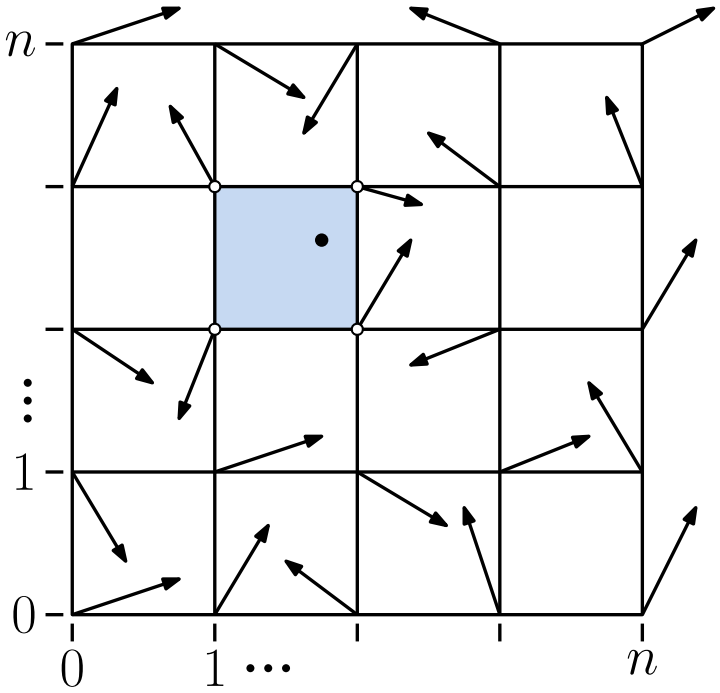
2D Perlin algorithm

Initial grid



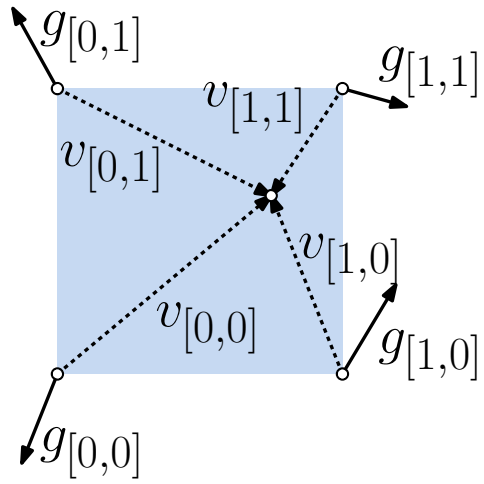
(a)

Random gradient vectors



(b)

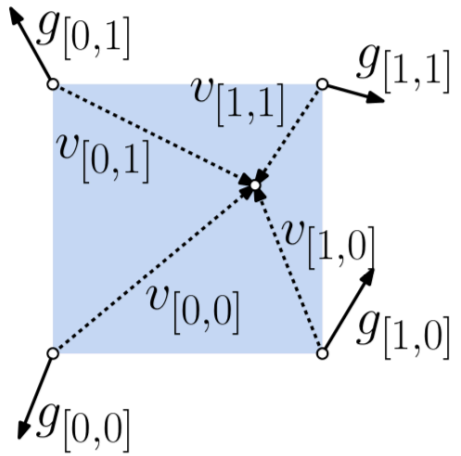
Smoothing/Interpolation



(c)

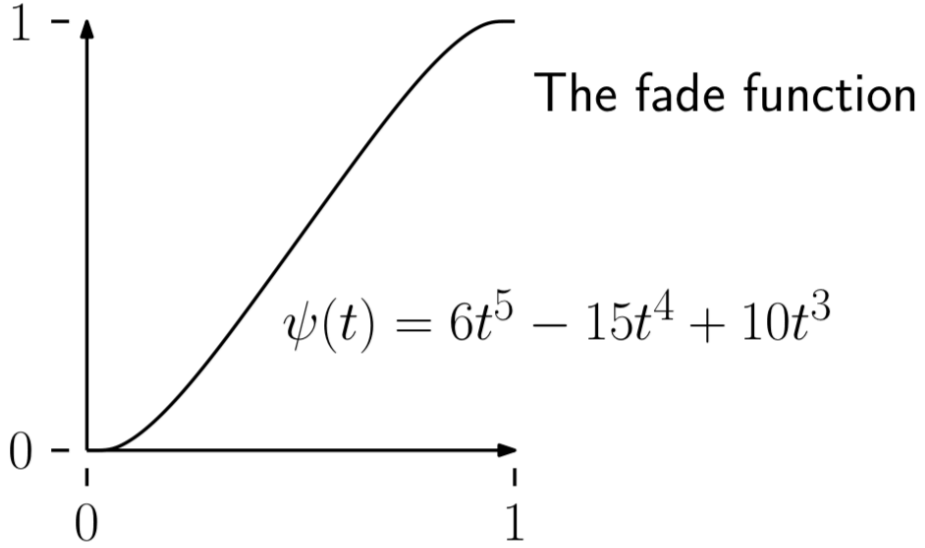
2D Perlin algorithm – gradient factor

$$\delta_{[0,0]} = (v_{[0,0]} \cdot g_{[0,0]}) \quad \text{and} \quad \delta_{[0,1]} = (v_{[0,1]} \cdot g_{[0,1]})$$
$$\delta_{[1,0]} = (v_{[1,0]} \cdot g_{[1,0]}) \quad \text{and} \quad \delta_{[1,1]} = (v_{[1,1]} \cdot g_{[1,1]}).$$



2D Perlin – fade function

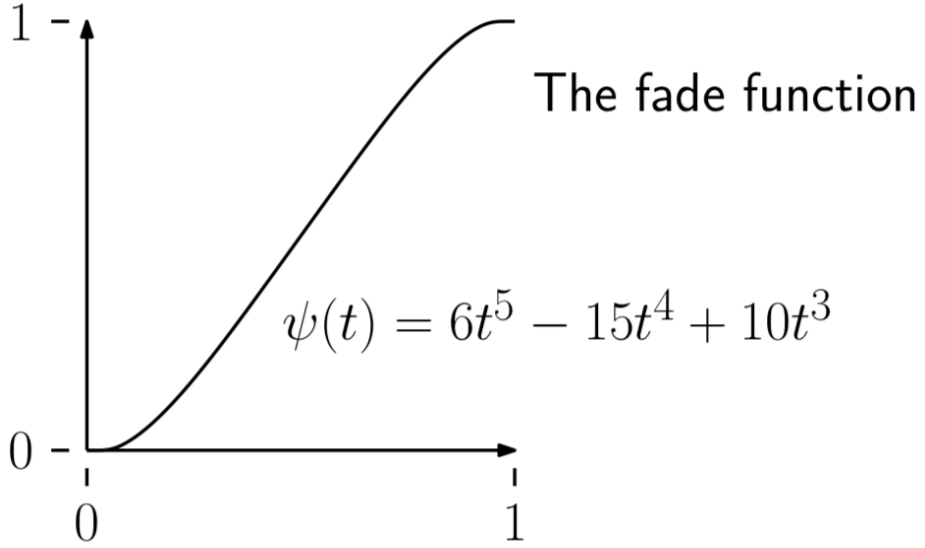
- $\psi(0) = 0$
- $\psi(1) = 1$
- $\psi'(0) = \psi'(1) = ??$



2D Perlin – fade function

- $\psi(0) = 0$
- $\psi(1) = 1$
- $\psi'(0) = \psi'(1) = ??$

- $\Psi(1) = \psi(s)\,\psi(t)$



2D Perlin – noise function

$$\text{noise}(x,y) \;=\; \Psi(1-x,1-y)\delta_{[0,0]} + \Psi(x,1-y)\delta_{[1,0]} + \Psi(1-x,y)\delta_{[0,1]} + \Psi(x,y)\delta_{[1,1]}$$

2D Perlin – noise function

$$\text{noise}(x,y) \;=\; \Psi(1-x,1-y)\delta_{[0,0]} + \Psi(x,1-y)\delta_{[1,0]} + \Psi(1-x,y)\delta_{[0,1]} + \Psi(x,y)\delta_{[1,1]}$$

$$\text{perlin}(x,y) \;=\; \sum_{i=0}^{k} p^i \cdot \text{noise}(2^i \cdot x, 2^i \cdot y)$$

- P is what parameter?