Perlin Noise I

CMSC425.01 Fall 2019

Administrivia

- Practice exam and list of potential problems on
- Will get additional practice problems up

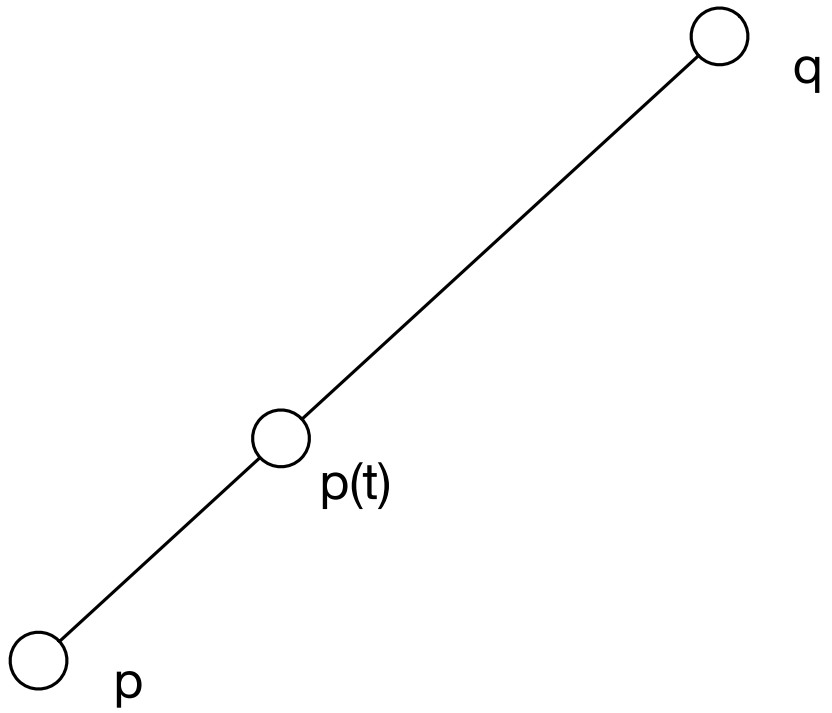
- Small winged edge homework on Elms now

Parametric line segments

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

with $v = q - p$

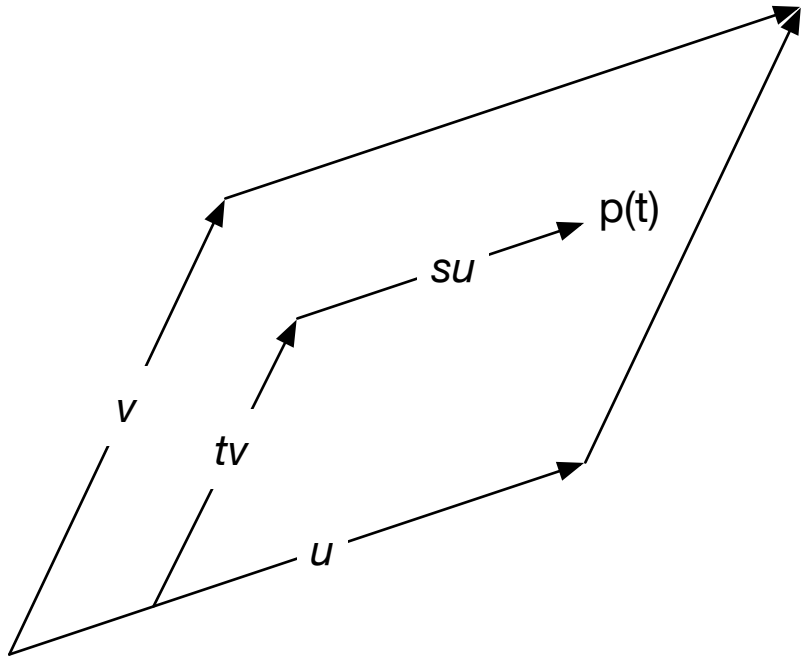
```
for t = 0 to 1 by deltat
     x = px + t * vx
     y = py + t * vy
     plot(x,y) // or line(lx,ly,x,y)
```



Parametric planar patches

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

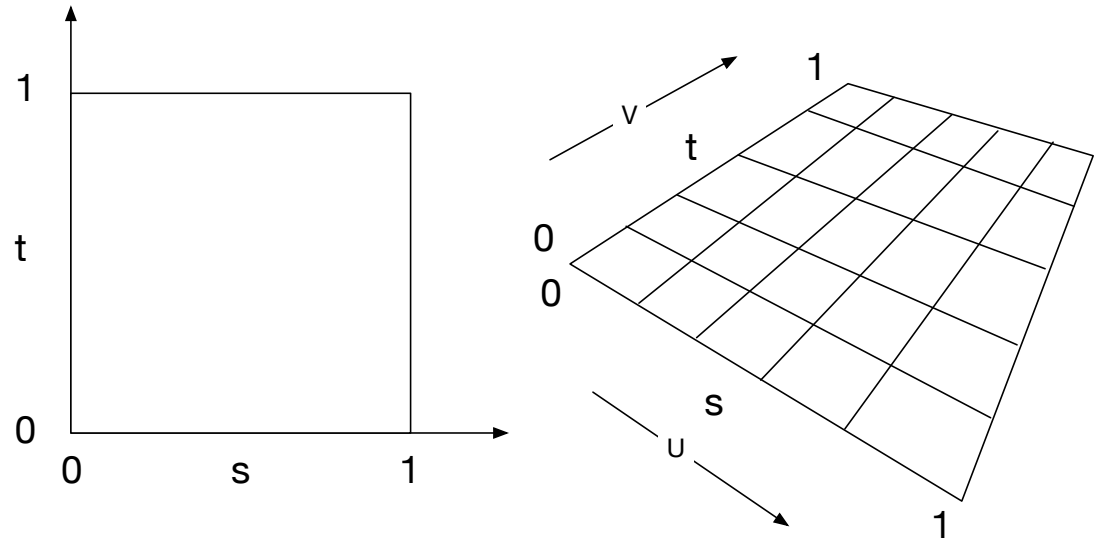
```
for t = 0 to 1 by deltat
  for s = 0 to 1 by deltas
     x = px + t * vx + s * ux
     y = py + t * vy + s * uy
     z = pz + t * vz + s * uz
     plot(x,y,z)
```



Creating planar mesh

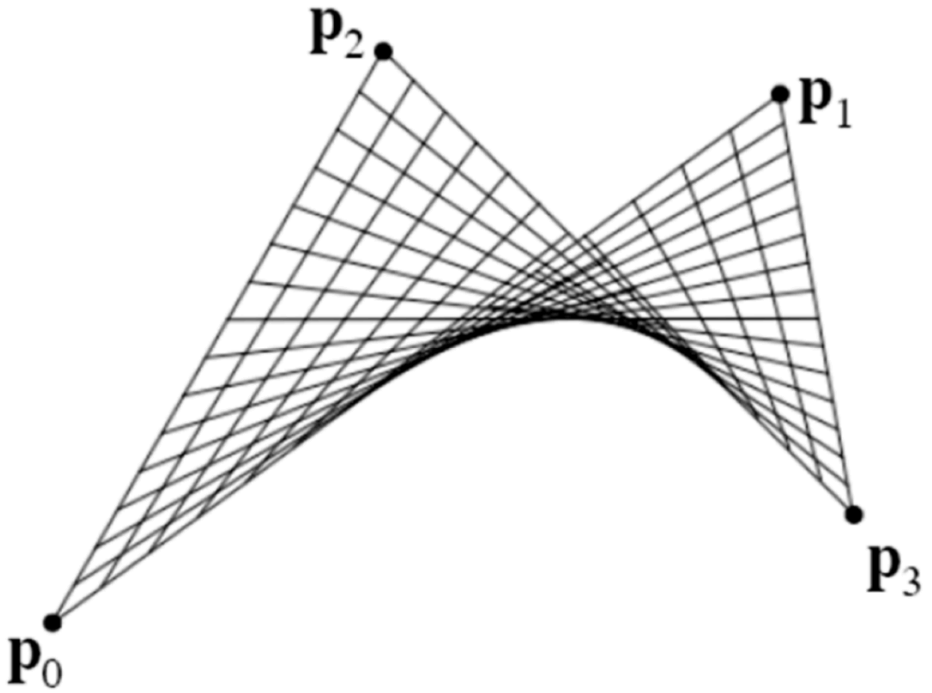
- How create mesh data structure from parametric patch?

- 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
- Developed equations in class



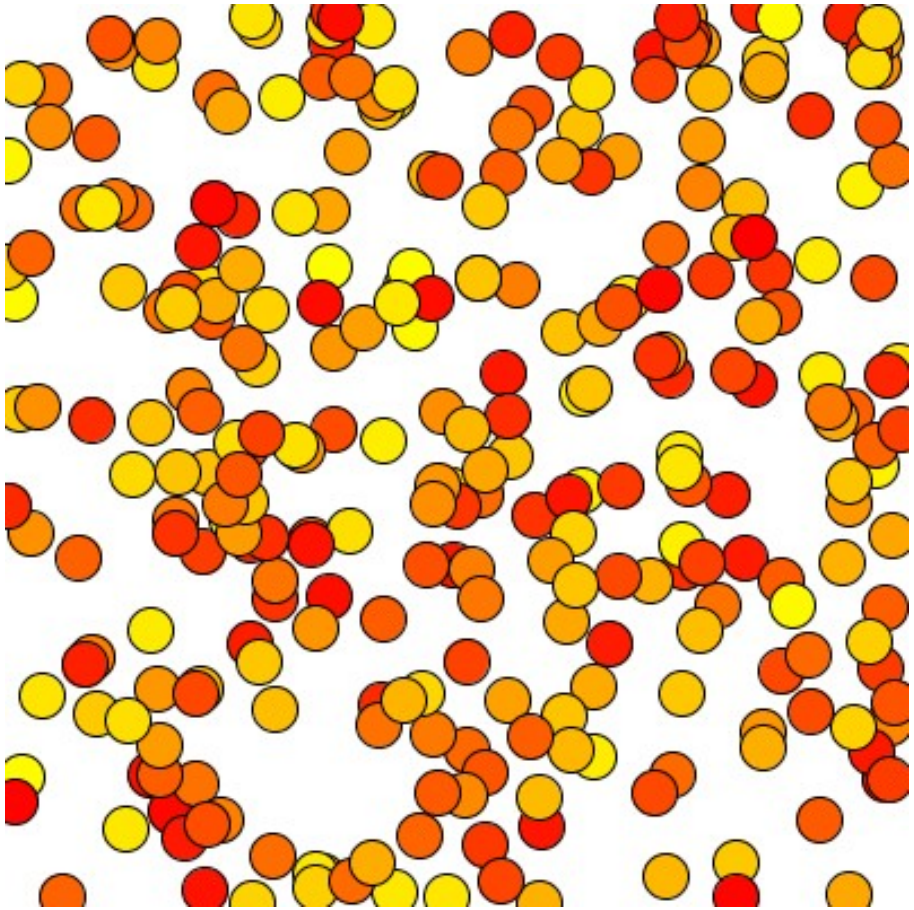
Today's question

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

Randomness – useful tool

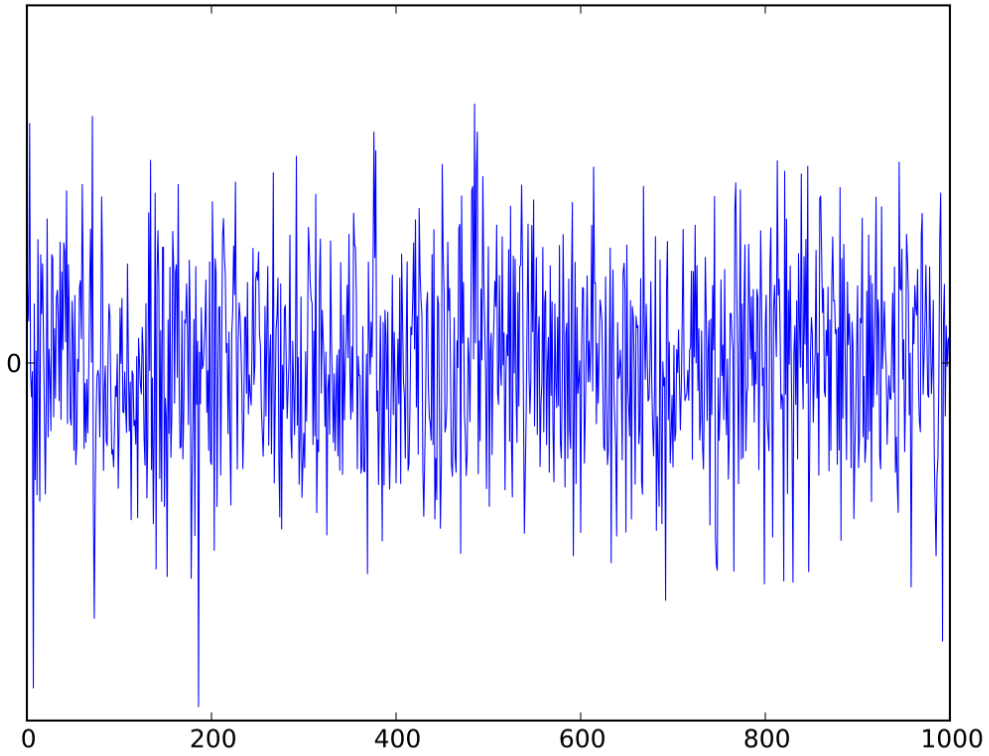
```
// RandomRain
void setup() {
  size(400,400);
  background(255);
  colorMode(HSB,360,100,100);
}

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);
}
```



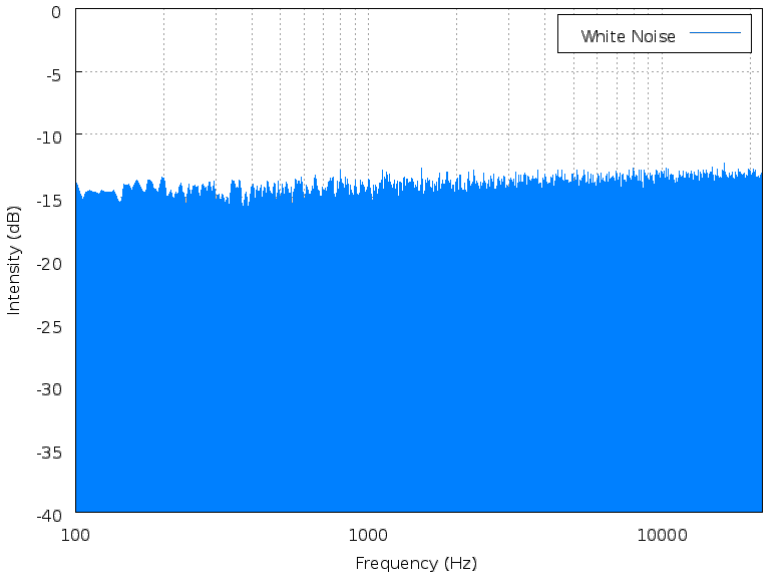
How make it natural and pleasing?

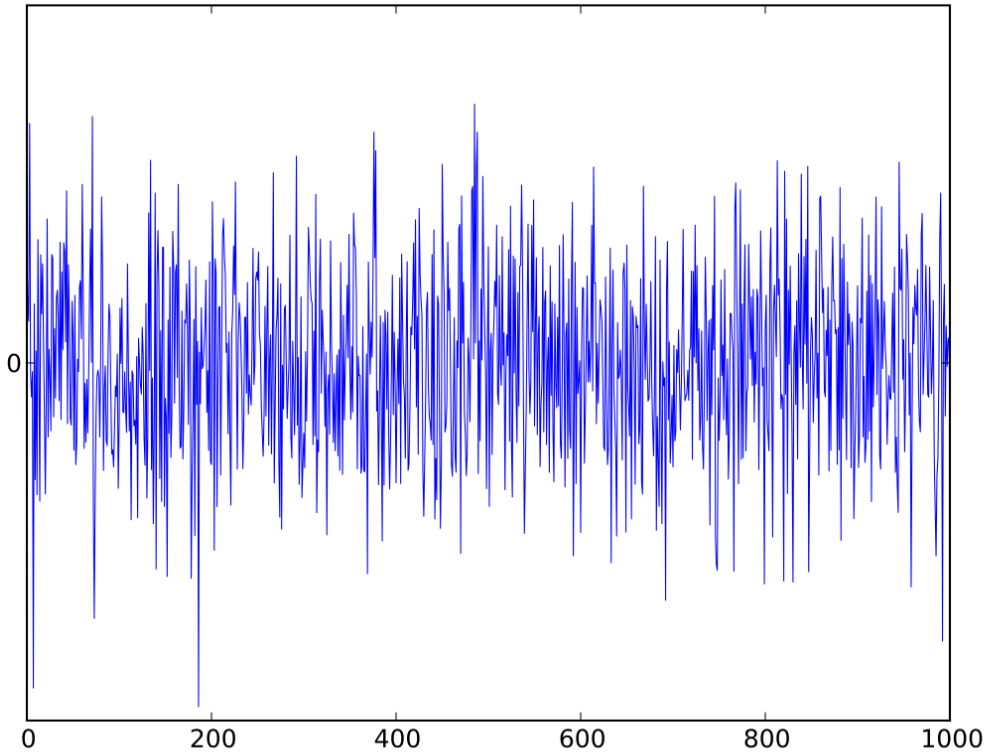
- Pure randomness – white noise
- Each data point independent of rest



White noise

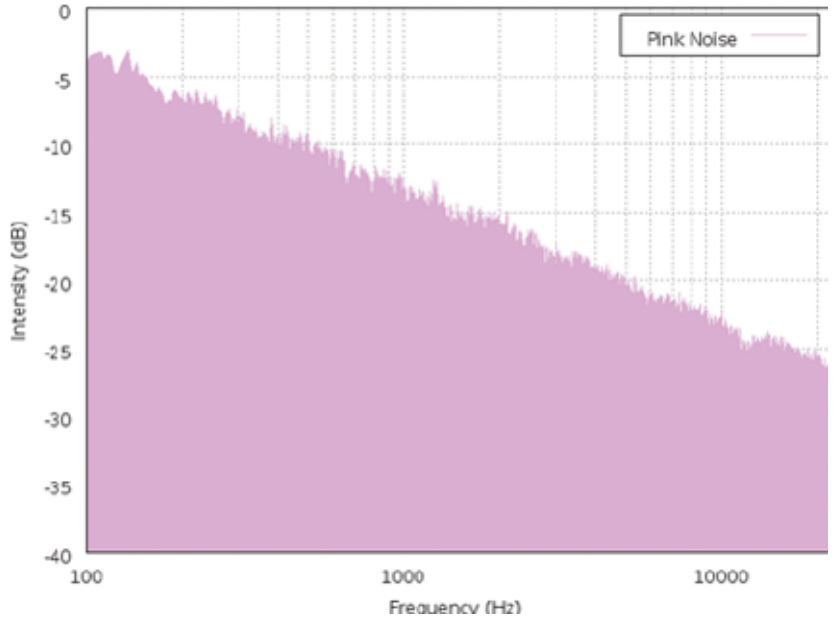
- Pure randomness – white noise
- Each data point independent of rest
- Frequency plot uniform

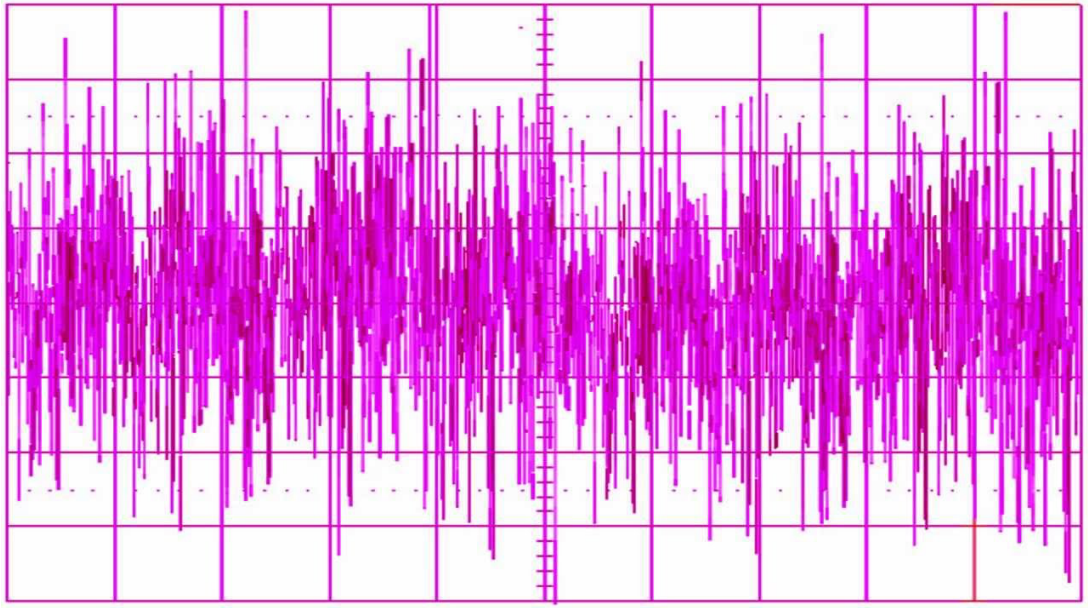




Pink noise

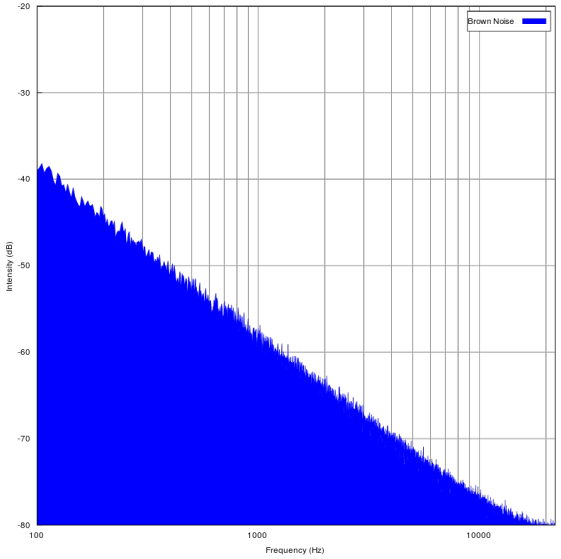
- Shaped randomness – pink noise
- Still independent
- Frequency plot 1/f

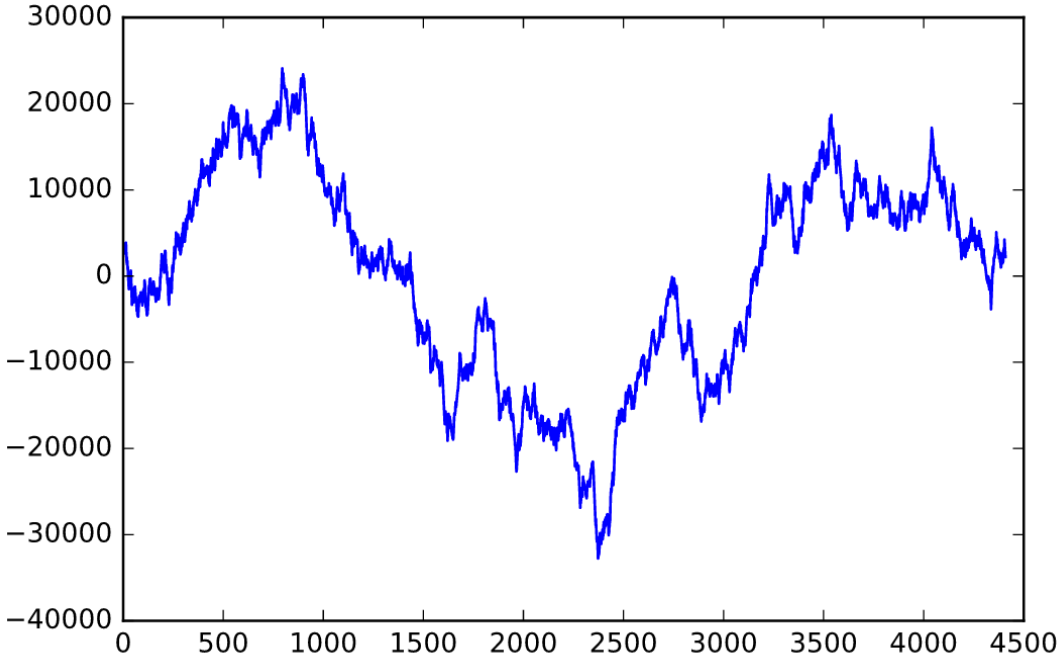




Brown noise

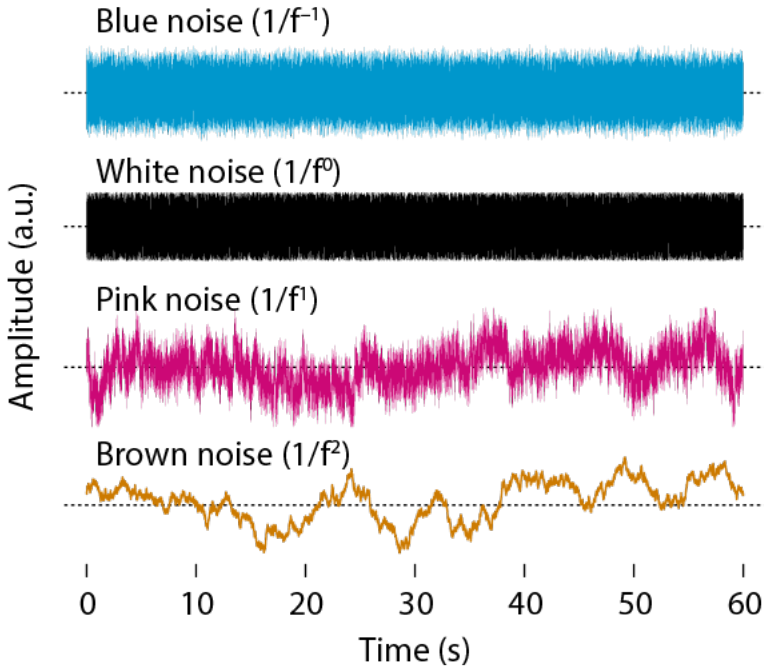
- Random walk – Brownian noise
- Each point random position from last (deltaY = random(-d,d))
- Frequency plot $1/f^2$

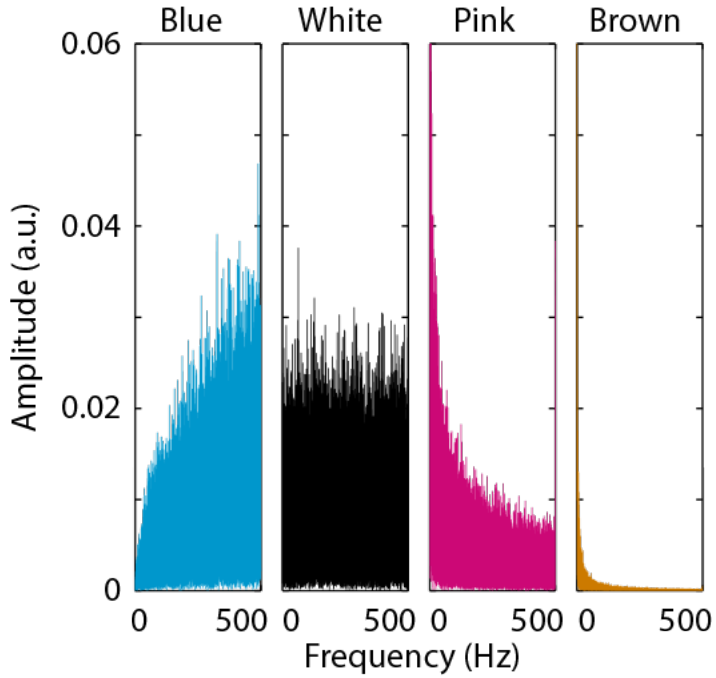




Colors of noise

- 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,



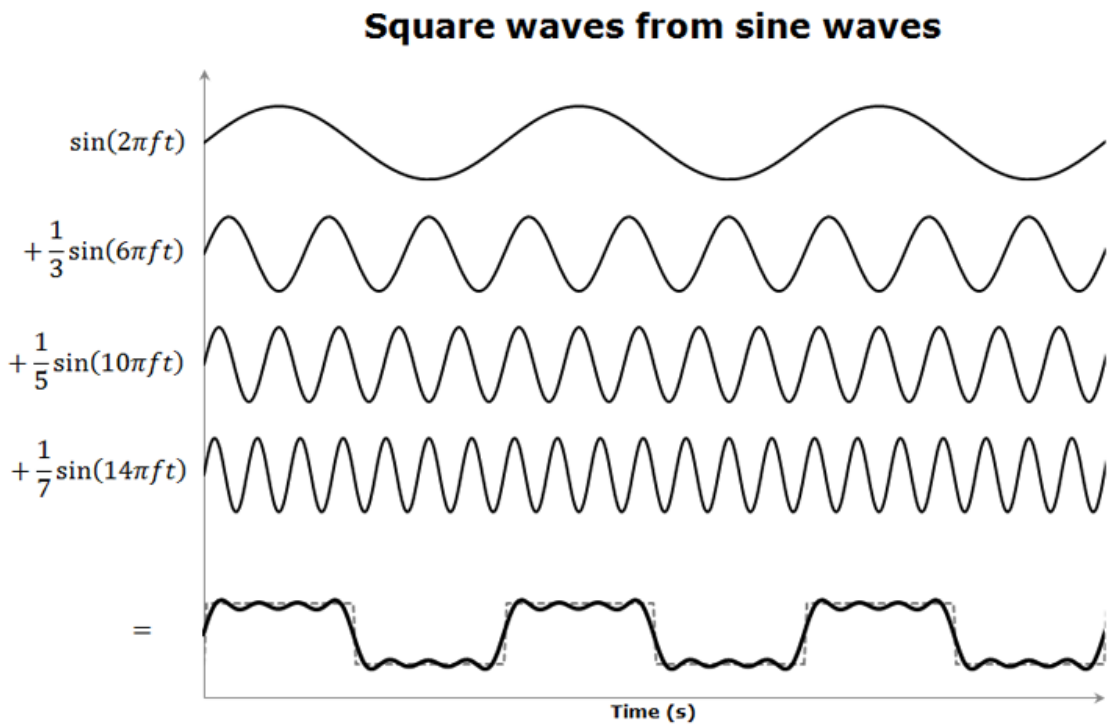


https://archive.org/details/TenMinutesOfWhiteNoisePinkNoiseAndBrownianNoise/BrownianNoise.flac

Generating $1/f^x$ noise

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

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

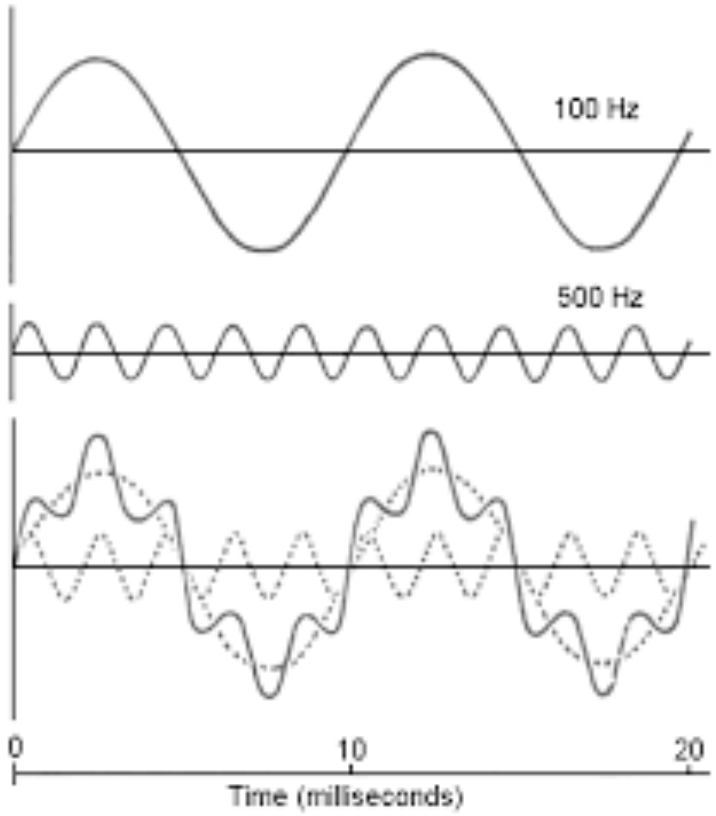


Generating $1/f^x$ noise

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

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

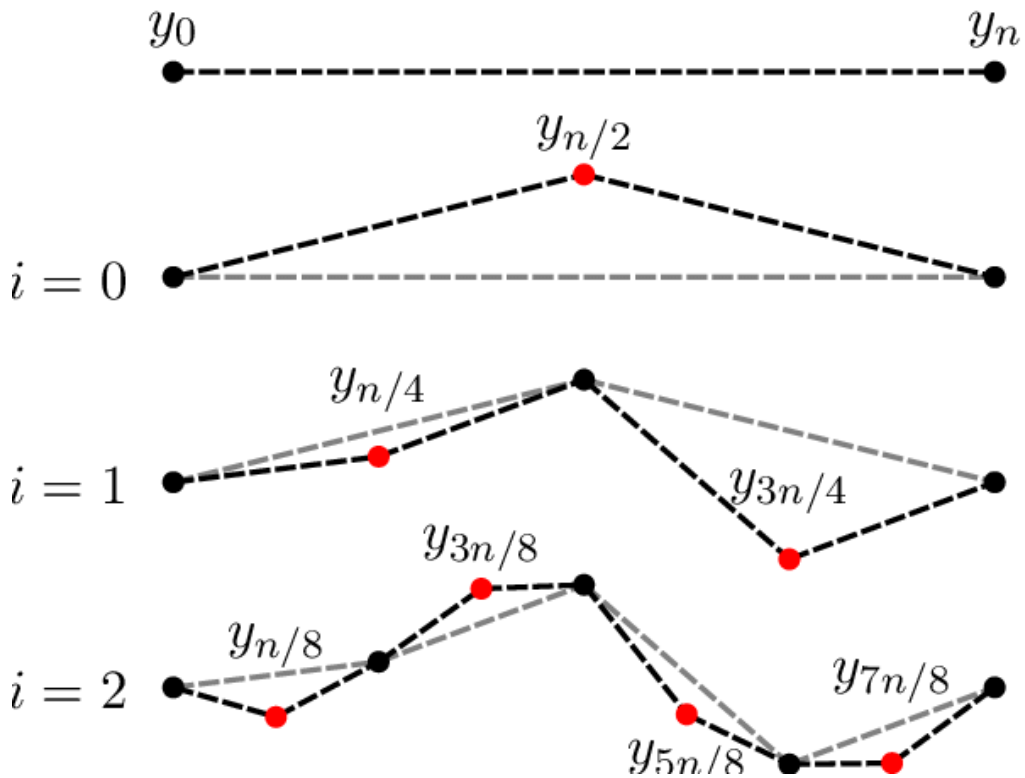
- Generate random terms of frequency, phase
- Decrease amplitude (height) as you increase frequency (n)



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

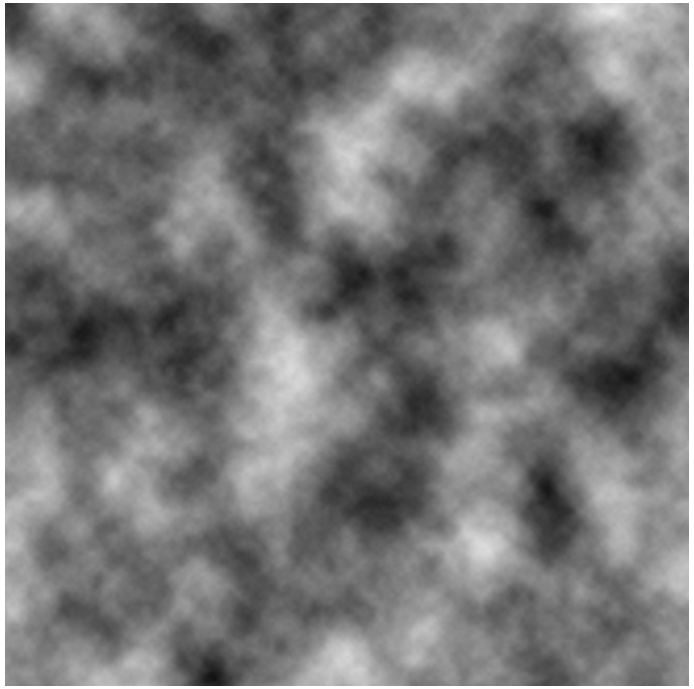
- Recursive curve generation
- Given two points:
  - Create perp bisector
  - Randomly pick t in (-h,h), generate point
  - Repeat for two new line segments
- Works in 3D

- Question

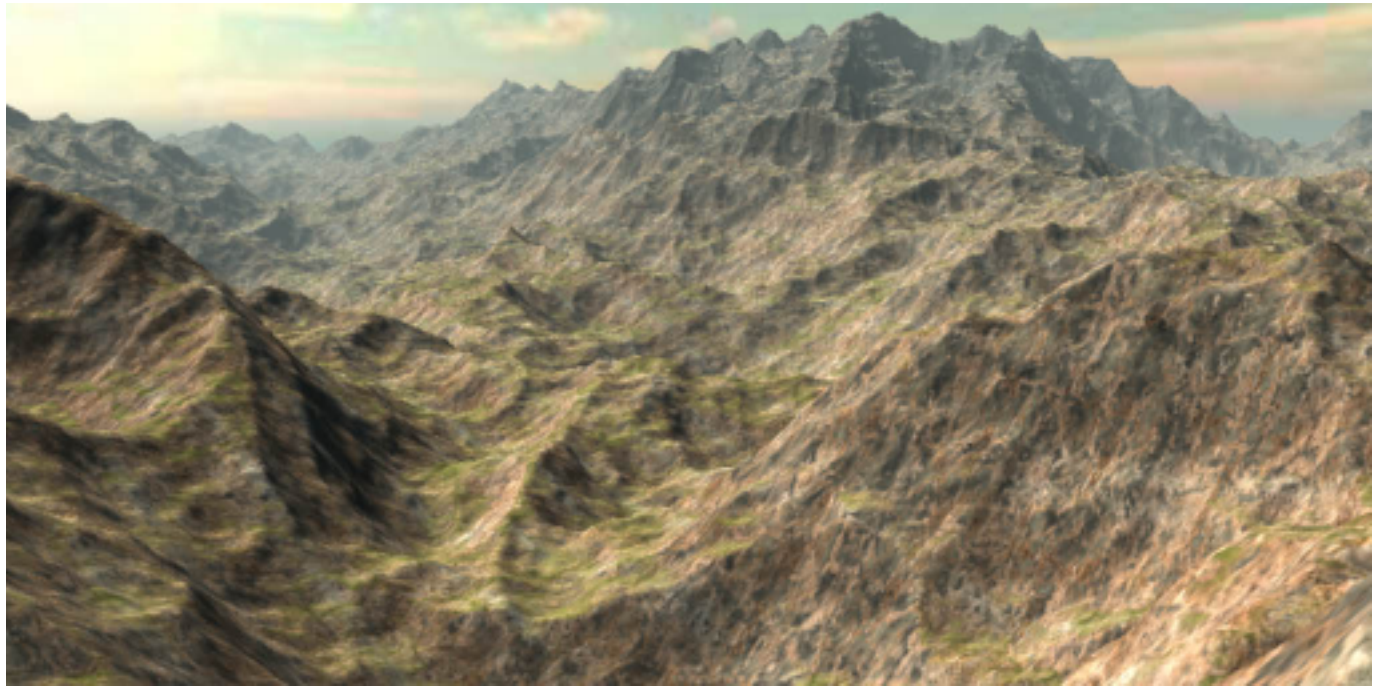
- How would you tune midpoint displacement to get more or less rugged landscapes?

Perlin noise

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



(a)



(b)

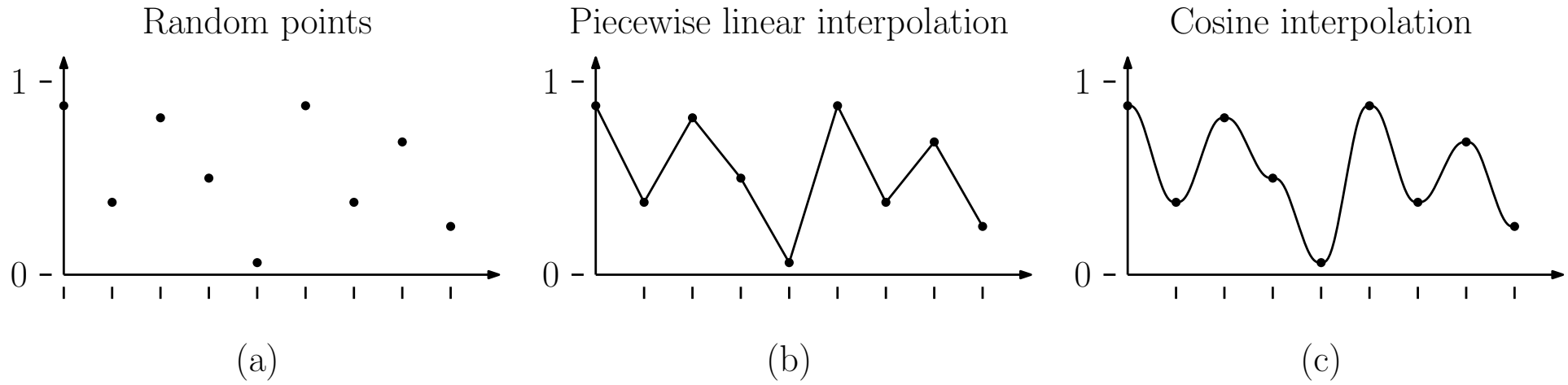
Perlin noise

- 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 $X = \langle x_0, x_1, x_2, \ldots, x_n \rangle$

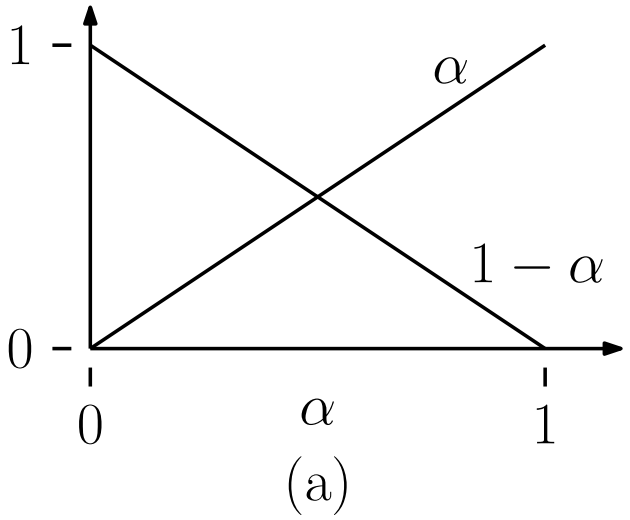
$$f_\ell(x) = \mathrm{lerp}(y_i, y_{i+1}, \alpha), \qquad \text{where } i = \lfloor x \rfloor \text{ and } \alpha = x \bmod 1$$



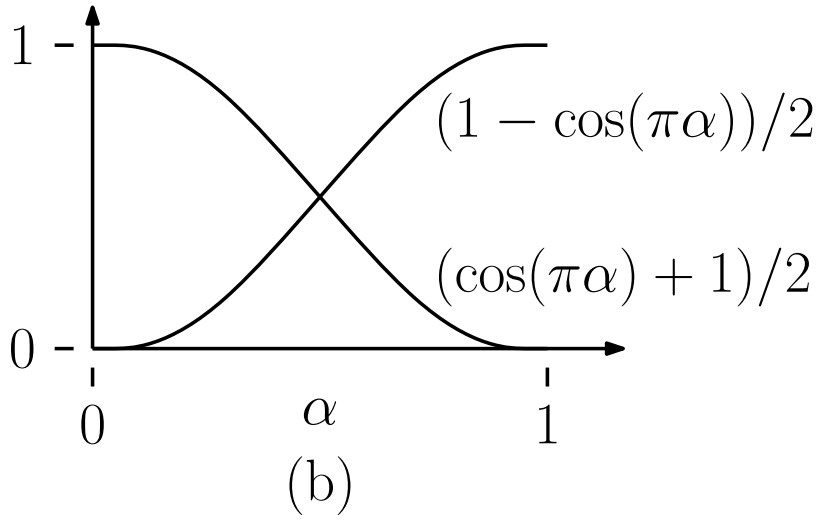
Interpolating weight functions

- 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)



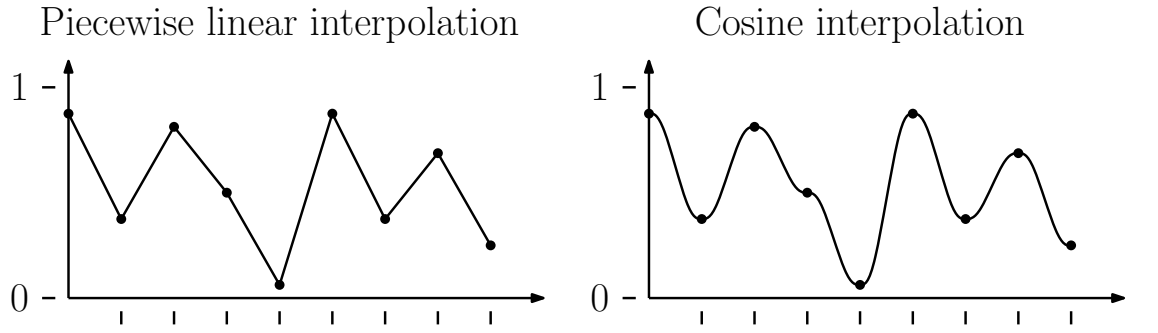
(b)

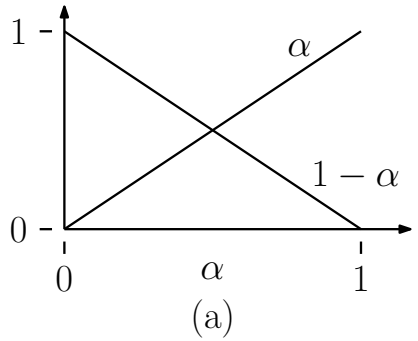
Interpolating weight functions

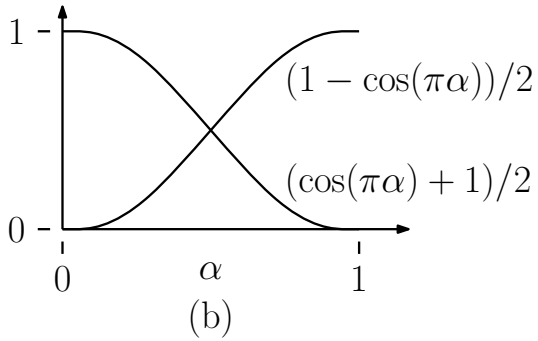
Cosine – smoother because

Slower to leave p0

Faster to arrive at p1

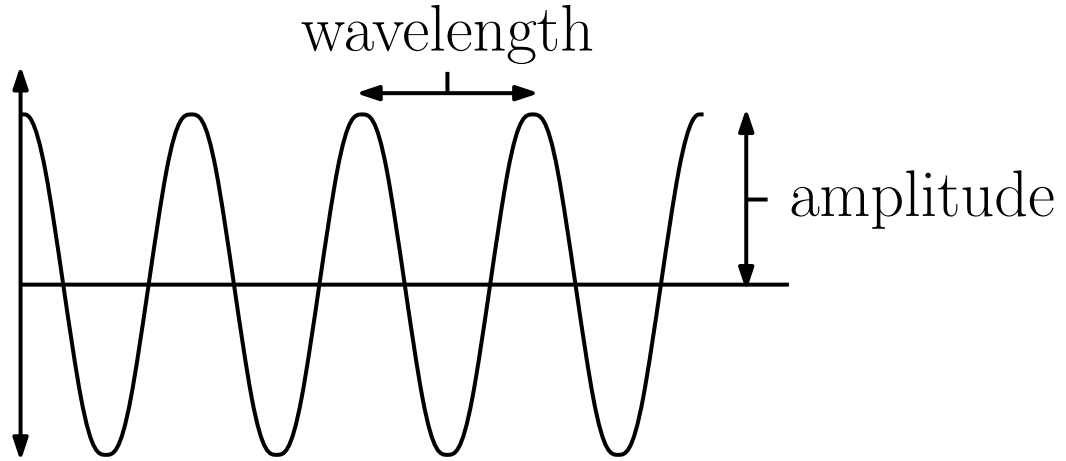






$\alpha \sin(\omega t)$

- ***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

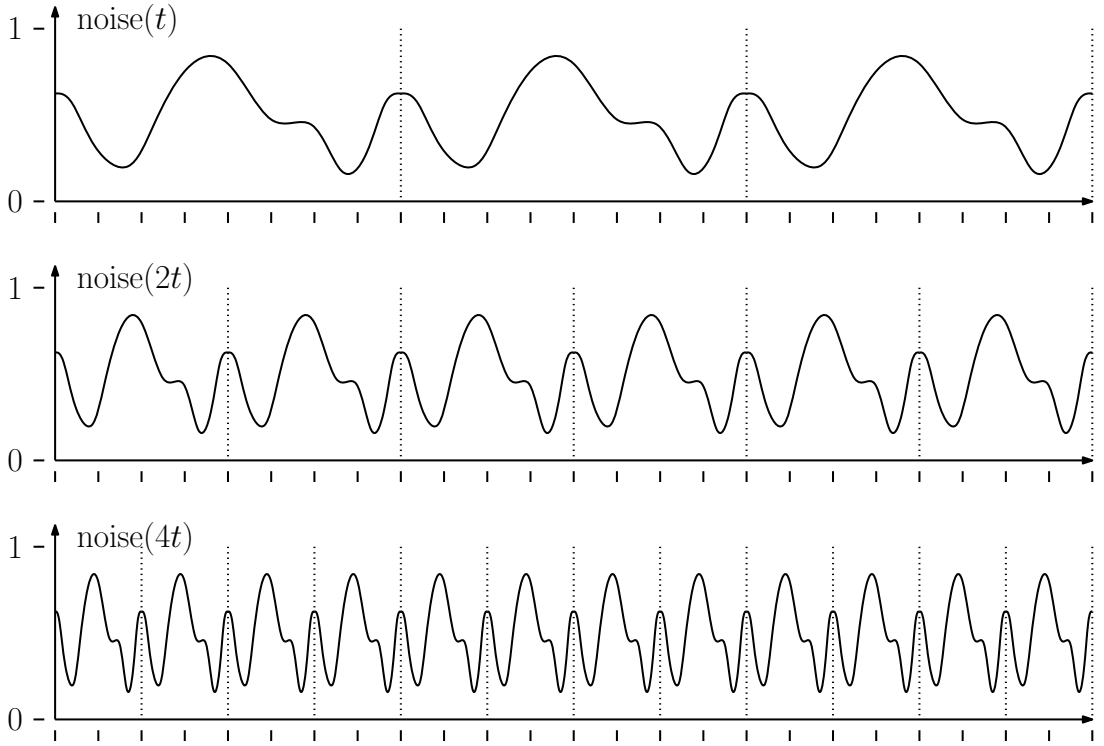
Periodic noise function

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

- Now define

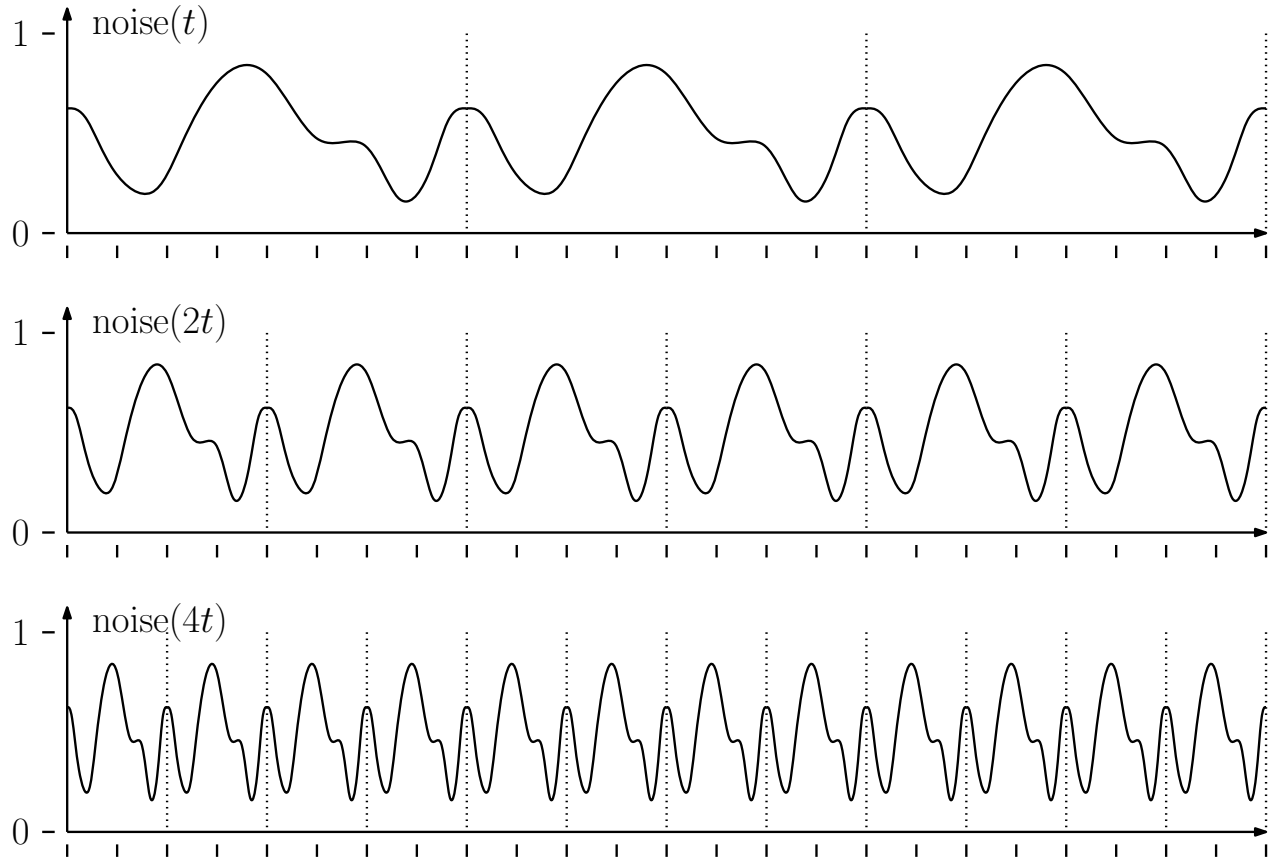
- $\text{noise}(t) = f(t \bmod n)$

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



Frequency octaves

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

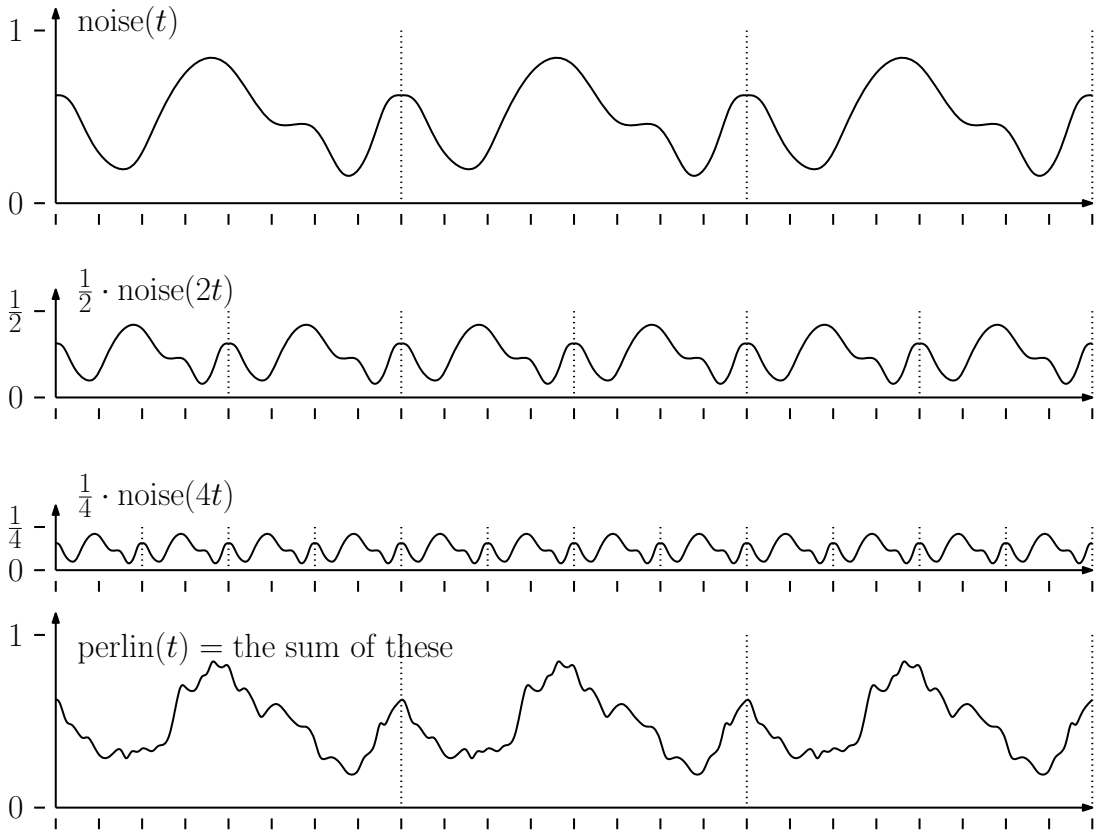


Persistence

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

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

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



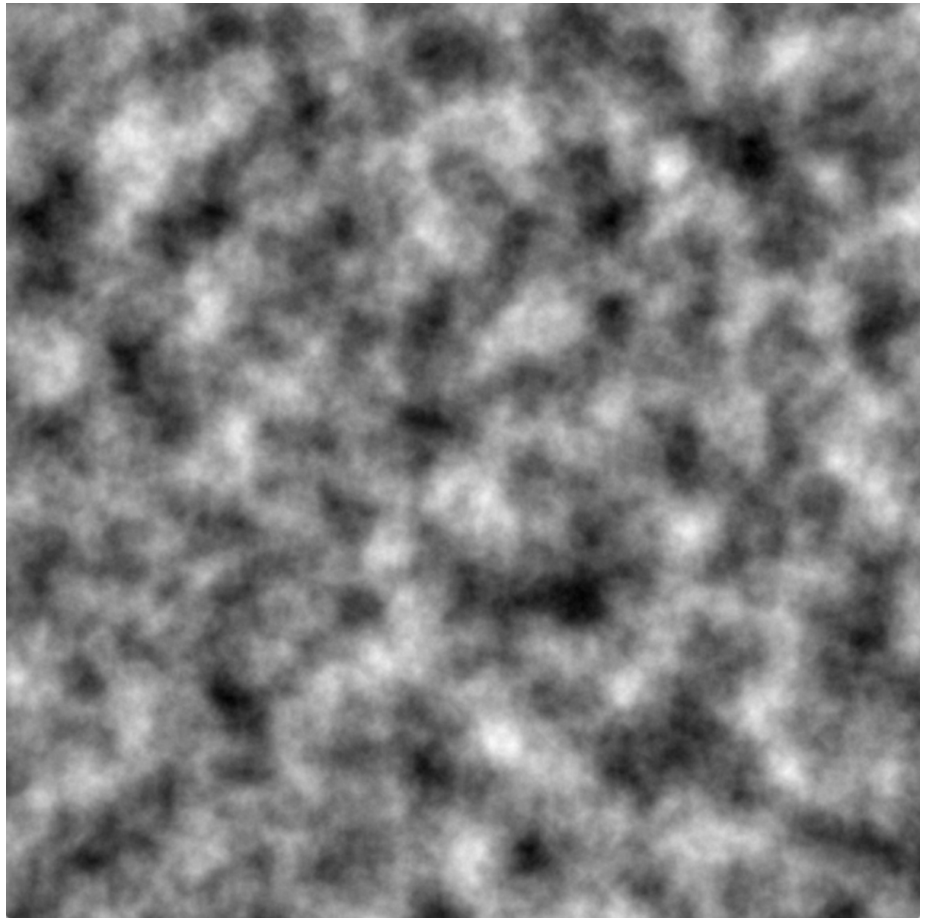
Perlin noise summary

- 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/

Unity: Scripting Perlin => Terrain

```
float[,] heights = new float[width, height];

for (int i = 0; i < width; i++) {
  for (int k = 0; k < height; k++) {
      heights [i,k] = baseHeight +  (float)hillHeight *
              (Mathf.PerlinNoise (
              ((float)i / (float)width) * tileSize,
              ((float)k / (float)height) * tileSize));
              }
       }


terrain.terrainData.SetHeights (0, 0, heights);
```

https://forum.unity.com/threads/perlin-noise-based-terrain-hill-generator-working-script.214701/

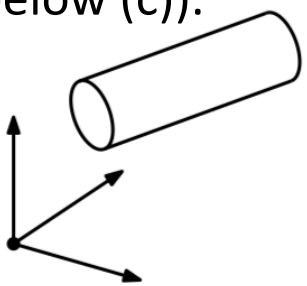
Question

- How would the idea of multiple scales apply to …
- Generating plants for a game
- Generating cities/towns/etc for a game
- Creating plot variations/bosses

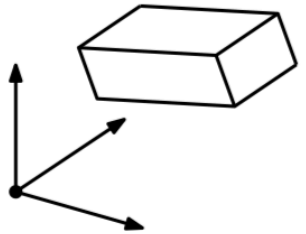
- 1. Metrics for best path on map
- 2. Navmesh process (R_D_P algorithm, triangulation)
- 3. Walkable terrain
- 4. Find paths on triangulated space
- 5. Configuration spaces
- 6. Quality of path
- 7. C-obstacles
- 8. Minkowski sums
- 9. Navmesh - grid, mulitresolution grid
- 10. Visibility graph
- ~~11. Medial axis~~
- 12. Randomized placement
- 13. Rapidly-expanded Random Trees (RRTs)
- 14. L-system plus turtle
- 15. Fractal dimension
- 16. Randomized and 3D L-systems
- 17. Particle systems
- 18. Flocking
- 19. Mandelbrot sets
- 20. Constructive solid geometry
- 21. Shading equation
- 22. Bump mapping
- 23. Polygonal meshes - basics, Euler's formula
- 24. DECL data structures
- 25. Perlin noise
- 26. A*
- 27. Admissible heuristic

Problem – configuration spaces

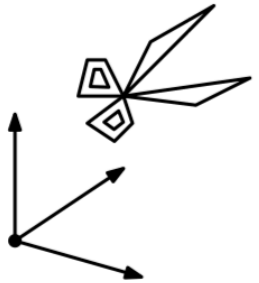
- How many dimensions are there in the configuration spaces for each of the following motion-planning problems. Justify your answer in each case by explaining what each coordinate of the space corresponds to.
- (i) Moving a cylindrical shape in 3-dimensional space, which may be translated and rotated (see the figure below (a)).
- (ii) Moving a brick in 3-dimensional space, which may be translated and rotated (see the figure below (b)).
- (iii) Moving a pair of scissors in 3-dimensional space, which may be translated, rotated, and swung open and closed (see the figure below (c)).



(a)



(b)



(c)

Problem 3. (20 points) Consider the collection of shaded rectangular obstacles shown in the figure below, all contained within a large enclosing rectangle. Also, consider the triangular robot, whose reference point is located at a point s. (You may take s to be the origin.)

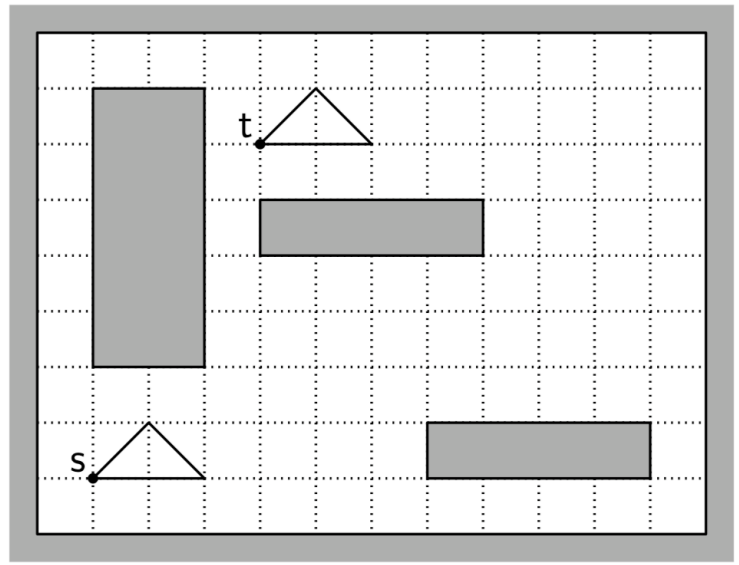


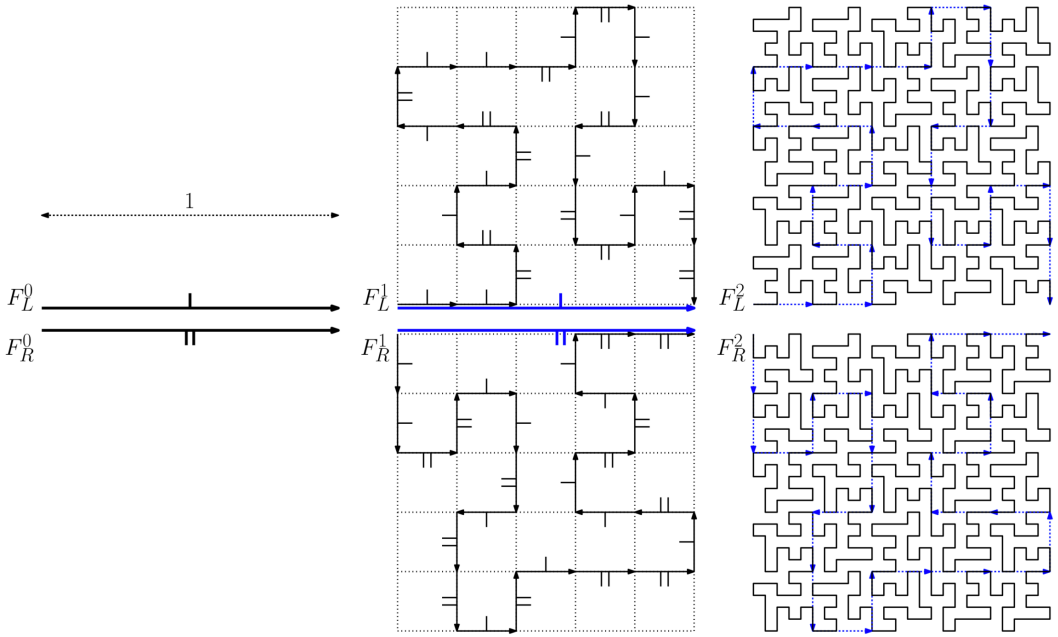
Figure 2: Problem 3.

(a) Draw the C-obstacles for the three rectangular obstacles, including the C-obstacle from region lying outside the large enclosing rectangle.

(b) Either draw an obstacle-avoiding path for the robot from s to t, or explain why it doesn't exist.

Problem – Fractal curve

- Derive an L-system that generates FL and FR. In particular, please provide the recursive rules for FL and FR.
- Consider the curve FL in the limit. Derive its fractal dimension.
- Each generation distances are scaled by σ = 1/5, and each individual segment of the basic length is replaced by 25 segments of the next smaller size.



Problem – DECL intersection

- Compute a list L = ⟨e1,e2,...,em⟩ of edges that intersect a line segment **ab**
- Given:
  - Faces fa and fb that contain a and b, respectively
  - Function e.cross(a,b) that returns true/false if edge e crosses **ab**