Procedural Shading

So many options for shading, how to represent?

Write a procedure!

- Simple function
- Specialized high-level *shading language*

Shading Languages

Shade Trees [Cook 84]

- Simple expressions: surface, light, atmosphere
- Built-in vector math & common shading functions

Image Synthesizer [Perlin 86]

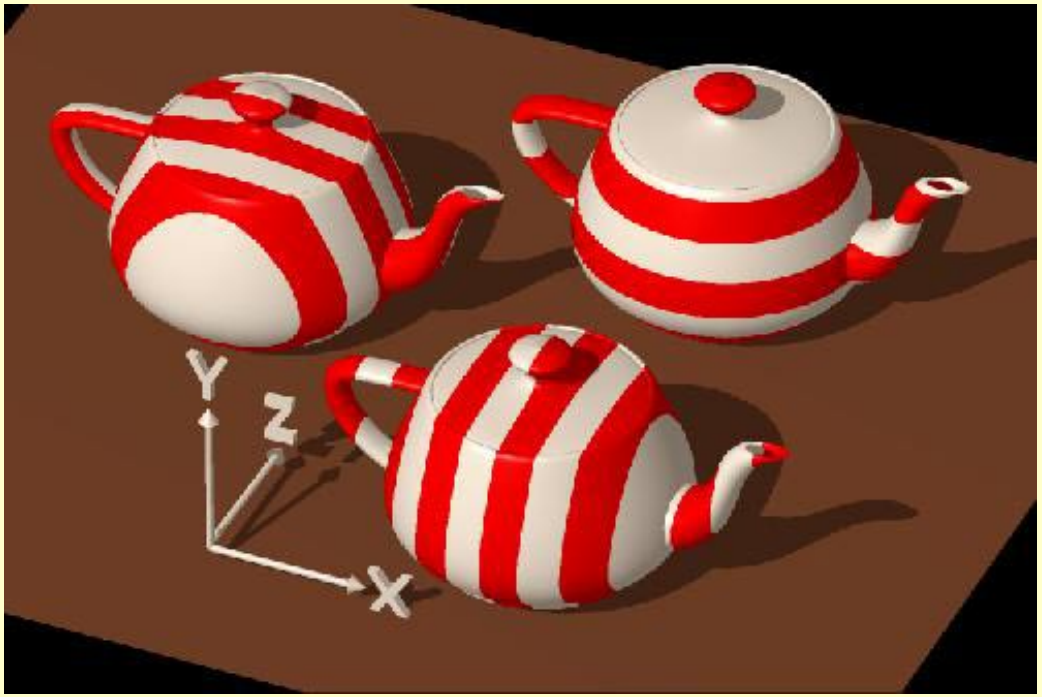
- Full language with branch & loop
- Band-limited noise function (more on this in a minute)

RenderMan [Hanrahan & Lawson 90/Pixar]

- C-like language
- Designed to work with many rendering algorithms
- Surface, light, displacement, volume/atmosphere

Shading Example

```
if (mod(trunc(zcomp(P)),2)==0)
  Ci = color(1,0,0);
else
  Ci = color(1,1,1);
```



RenderMan Surface Shaders

Input

- Cs, Os
- u, v, du, dv, s, t
- time, dtime
- P, N, Ng, dPdu, dPdv, dPdtime
- E, I
- L, Cl, Ol (In *illuminance*)

Output

- Ci, Oi

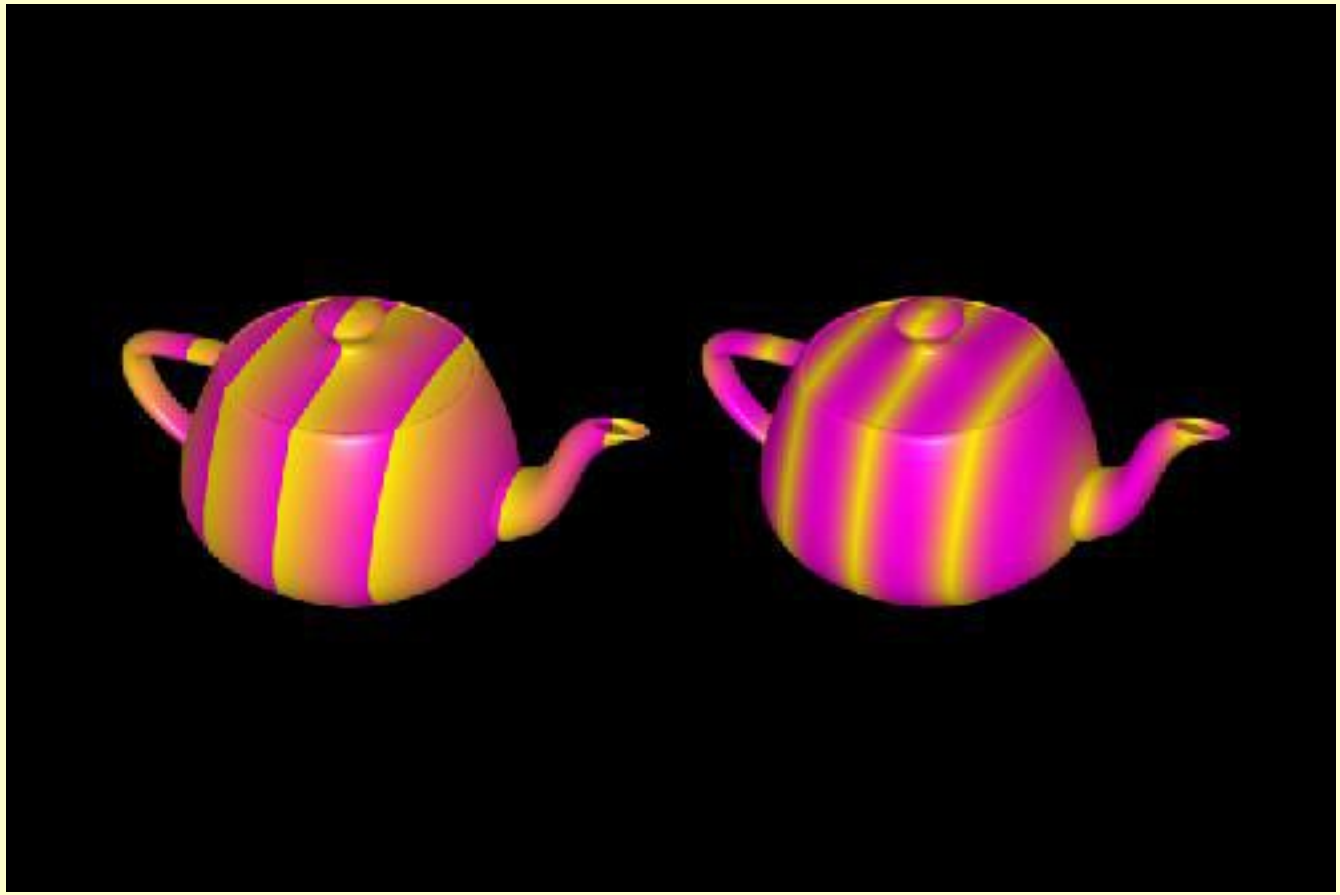
More Complex Example

```
float d = sqrt(
    xcomp(P)*xcomp(P) + ycomp(P)*ycomp(P));
if (mod(trunc(d),2)==0)
  Ci = color(1,0,0);
else
  Ci = color(1,1,1);
```

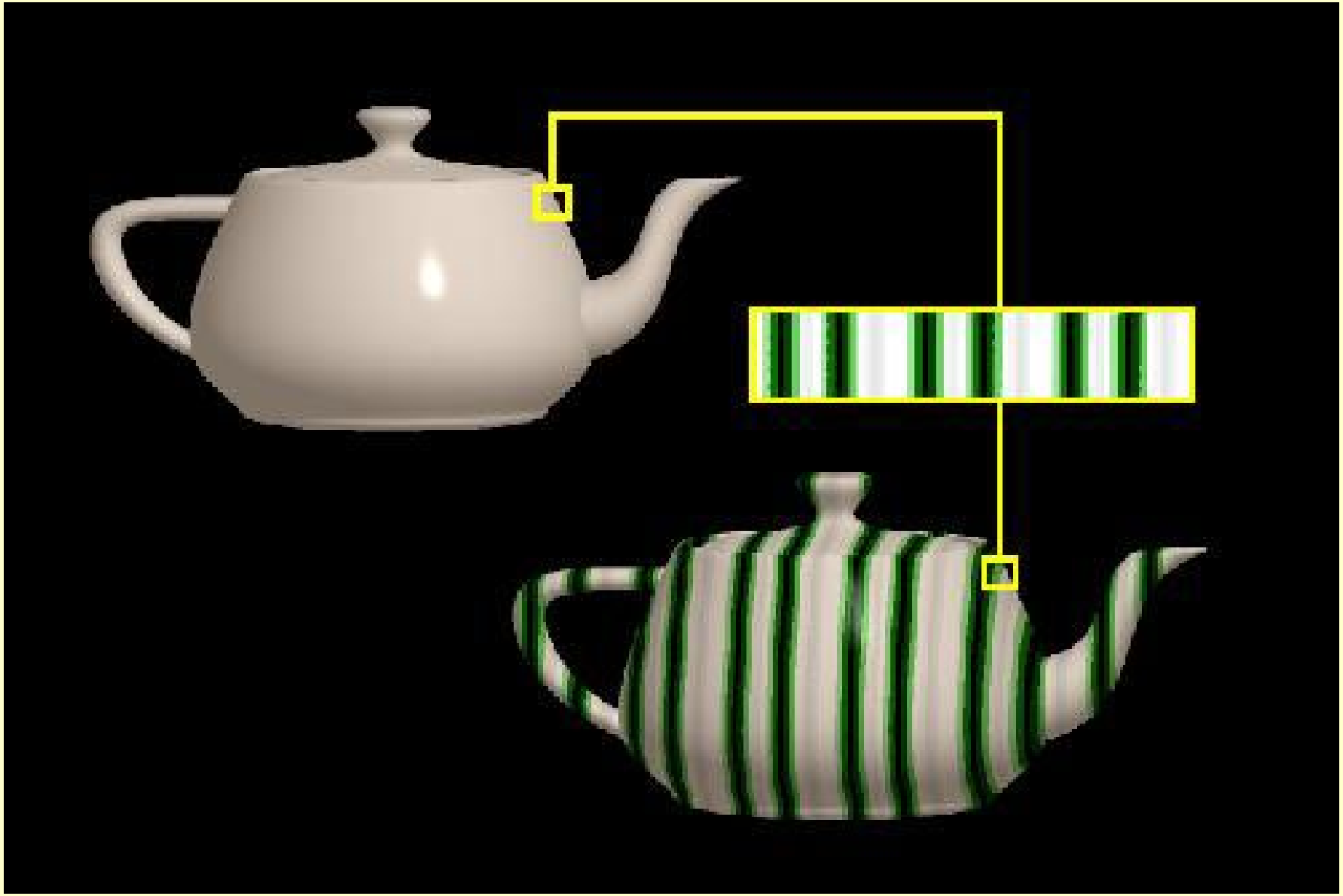
Repeating Patterns

```
float r1 = mod(x,2)/2;
Ci = mix(yellow,
  magenta, r1);
```

```
float r2 = sin(x);
Ci = mix(yellow,
  magenta, r2);
```

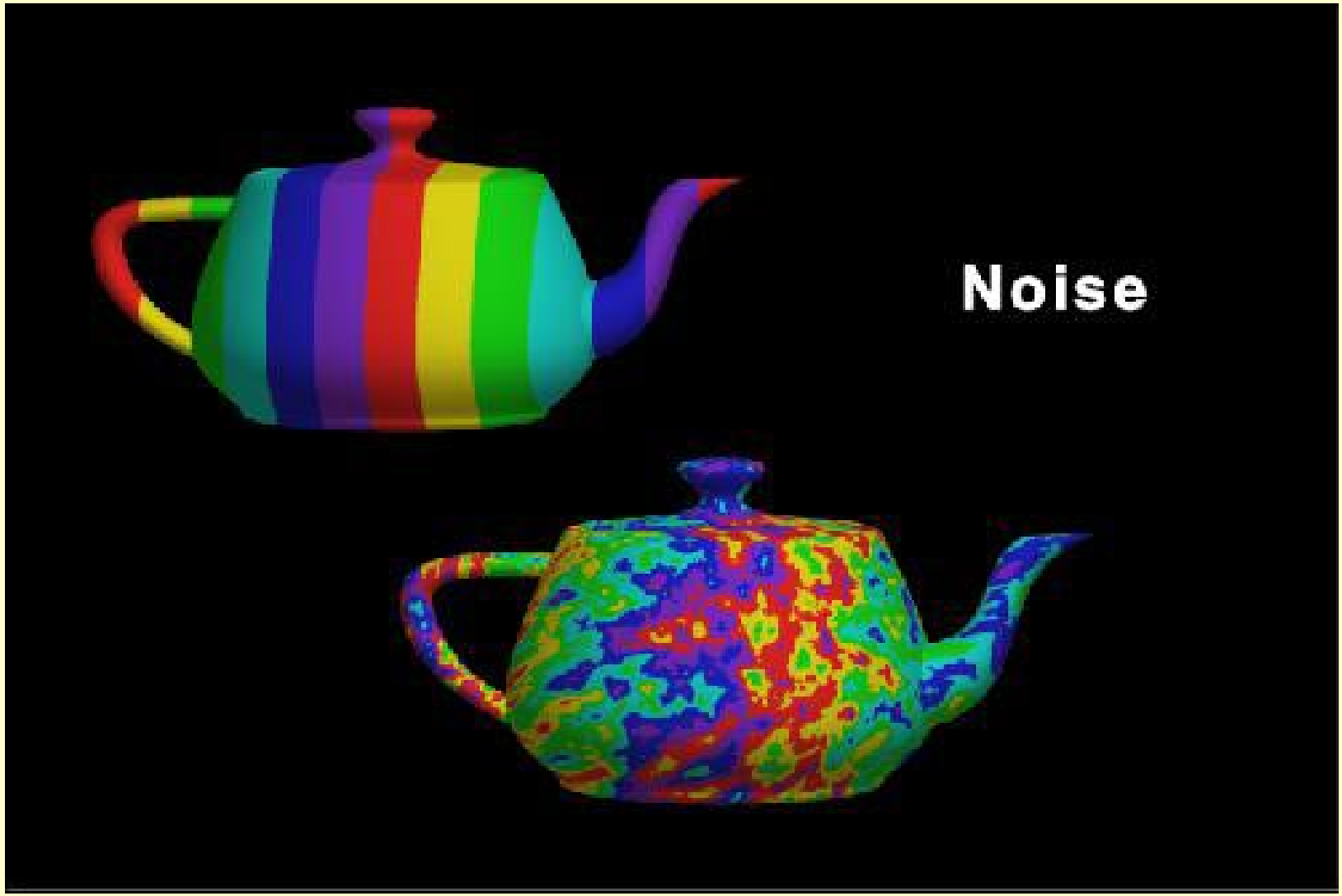


Color Tables



Perturbed Color Tables

```
Ci = texture("rainbow",x+noise(P));
```

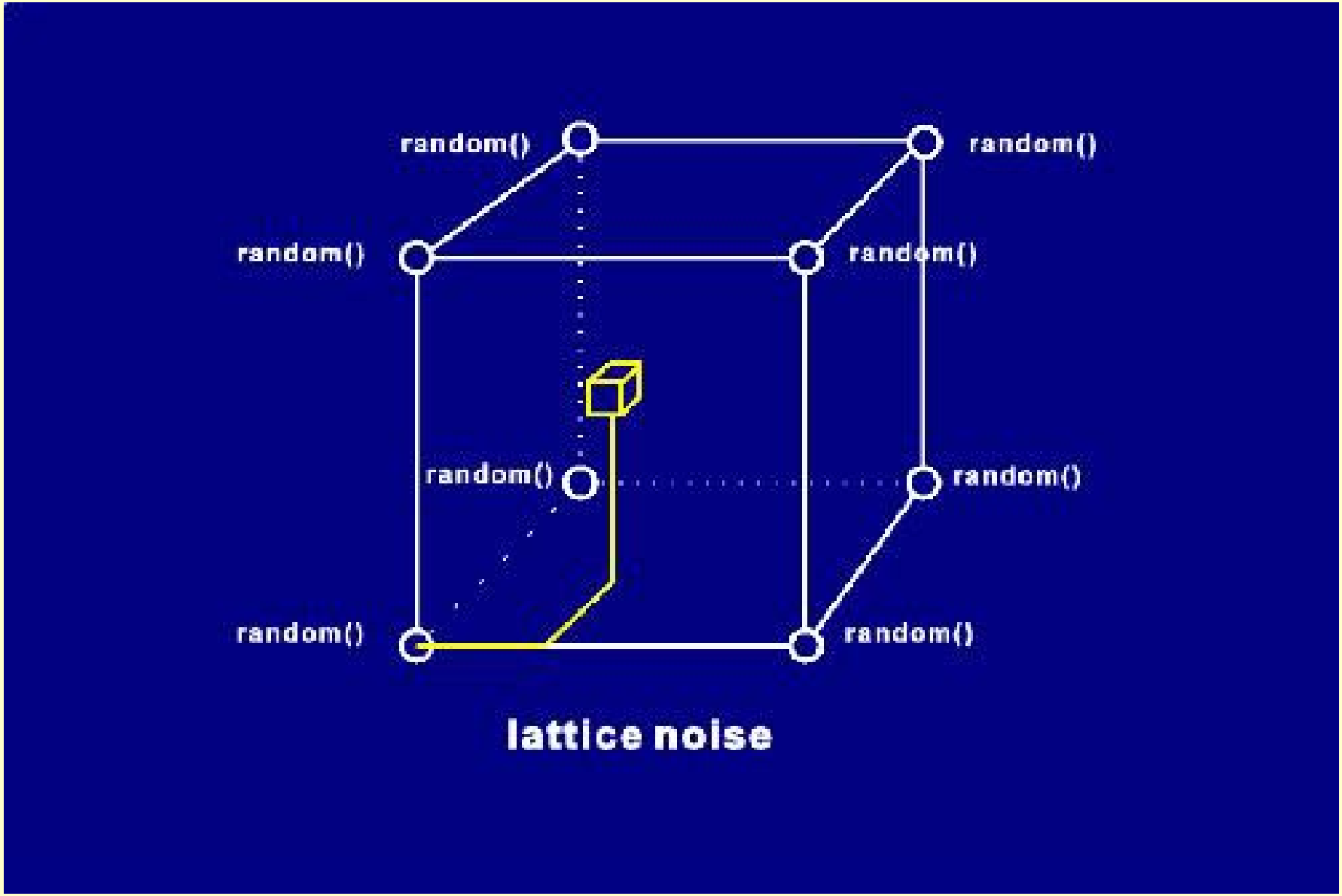


Noise

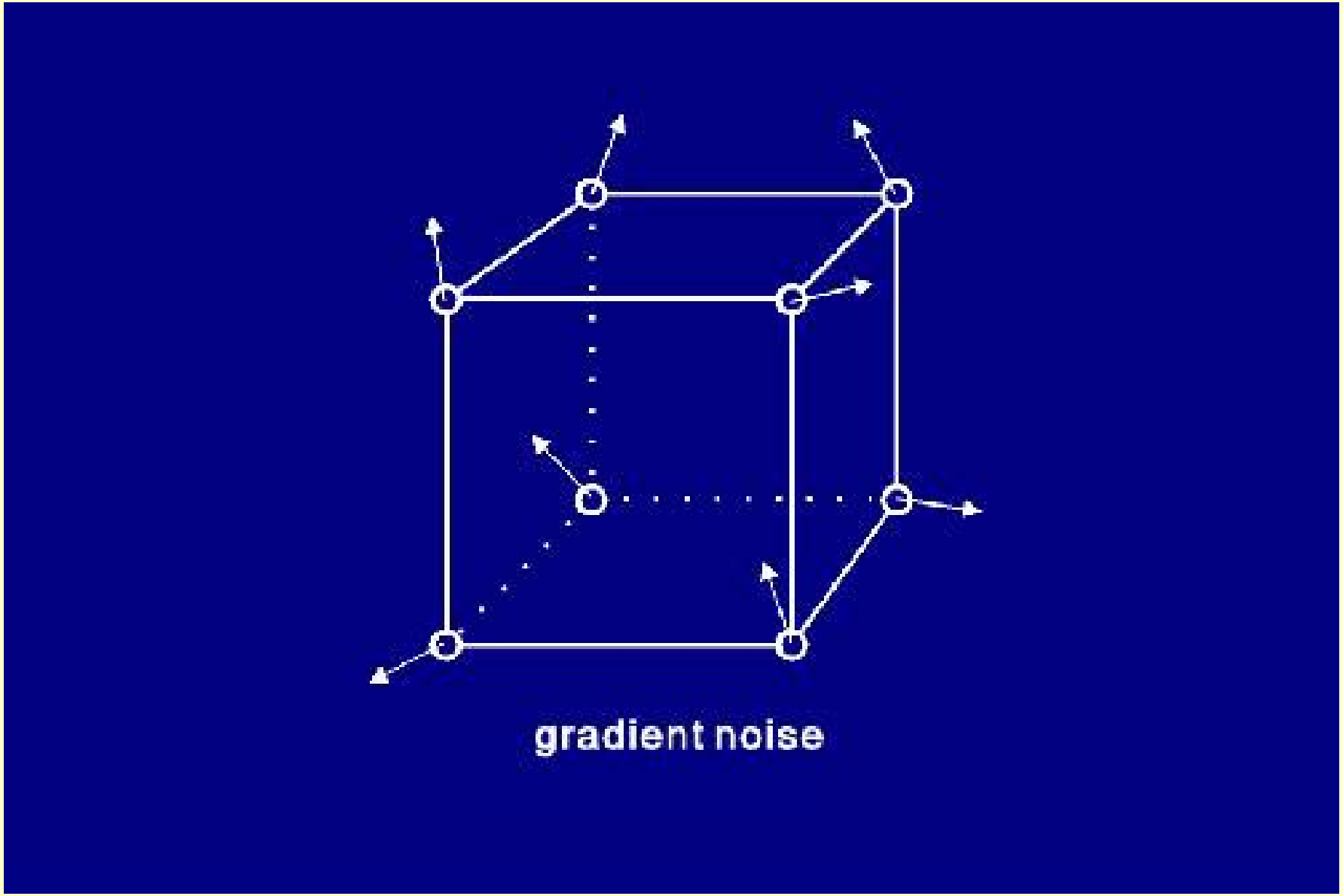
What is this *noise*?

- "it provides seasoning to help you make things irregular enough so that you can make them look more interesting" – Ken Perlin
- Random but repeatable
  - Different arguments give different random values
  - Same argument gives the same value every time
  - Consistency in animation!
- Frequency band-limited (approximately one octave)
  - Control: can choose frequency range by combining several octaves of noise

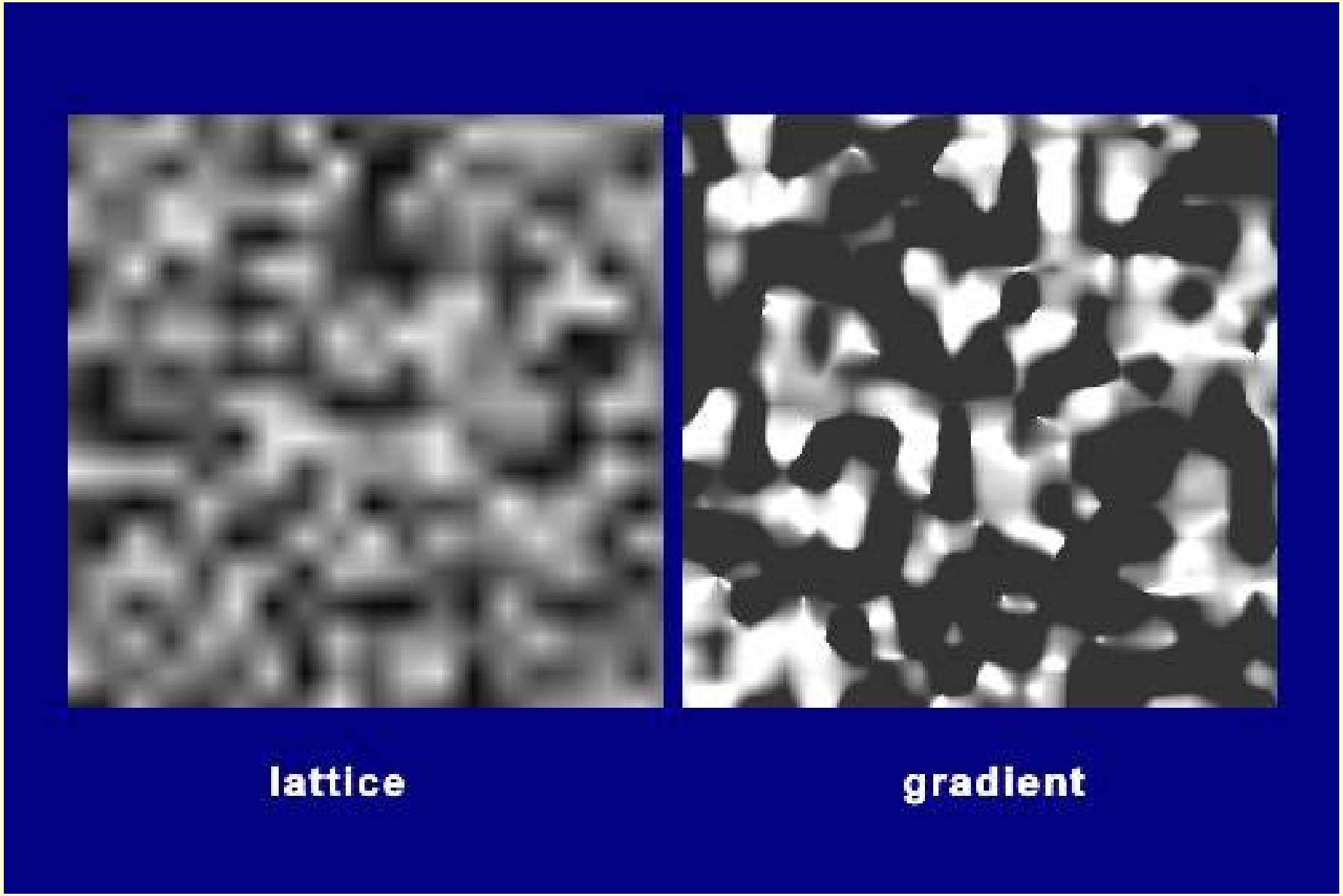
Lattice/Value Noise



Gradient Noise



Value vs. Gradient



RenderMan noise

Biased to give values centered at .5 rather than 0

float, color or vector output

- inferred, but can force: Ci = float noise(P);

1D, 2D, 3D or 4D

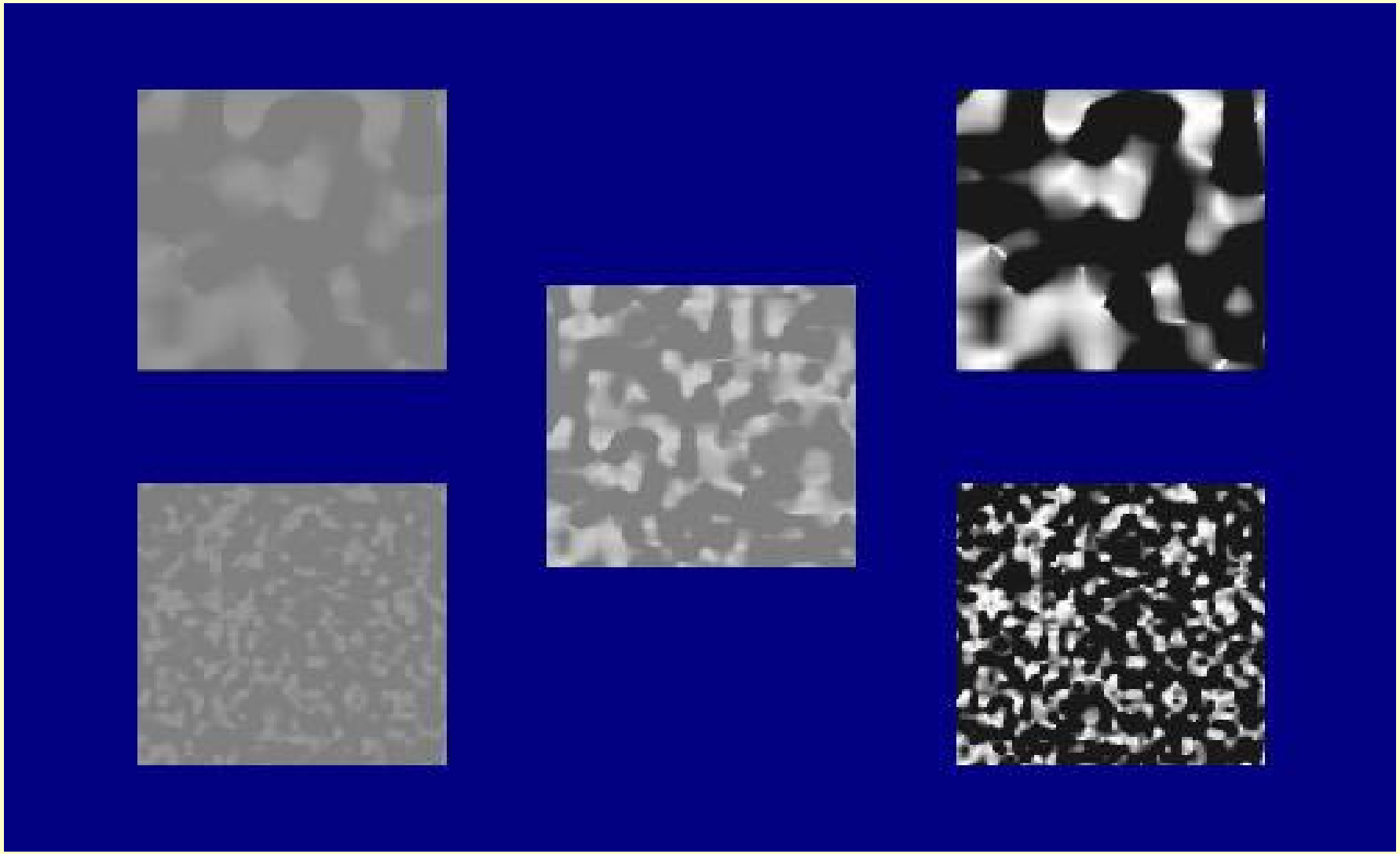
- noise(x)
- noise(x, y)
- float(P)
- noise(P, t)

Periodic version

- pnoise(x,y, x_period, y_period)

Noise Frequency & Amplitude

```
a*noise(f*P)
```



Simple RenderMan Example

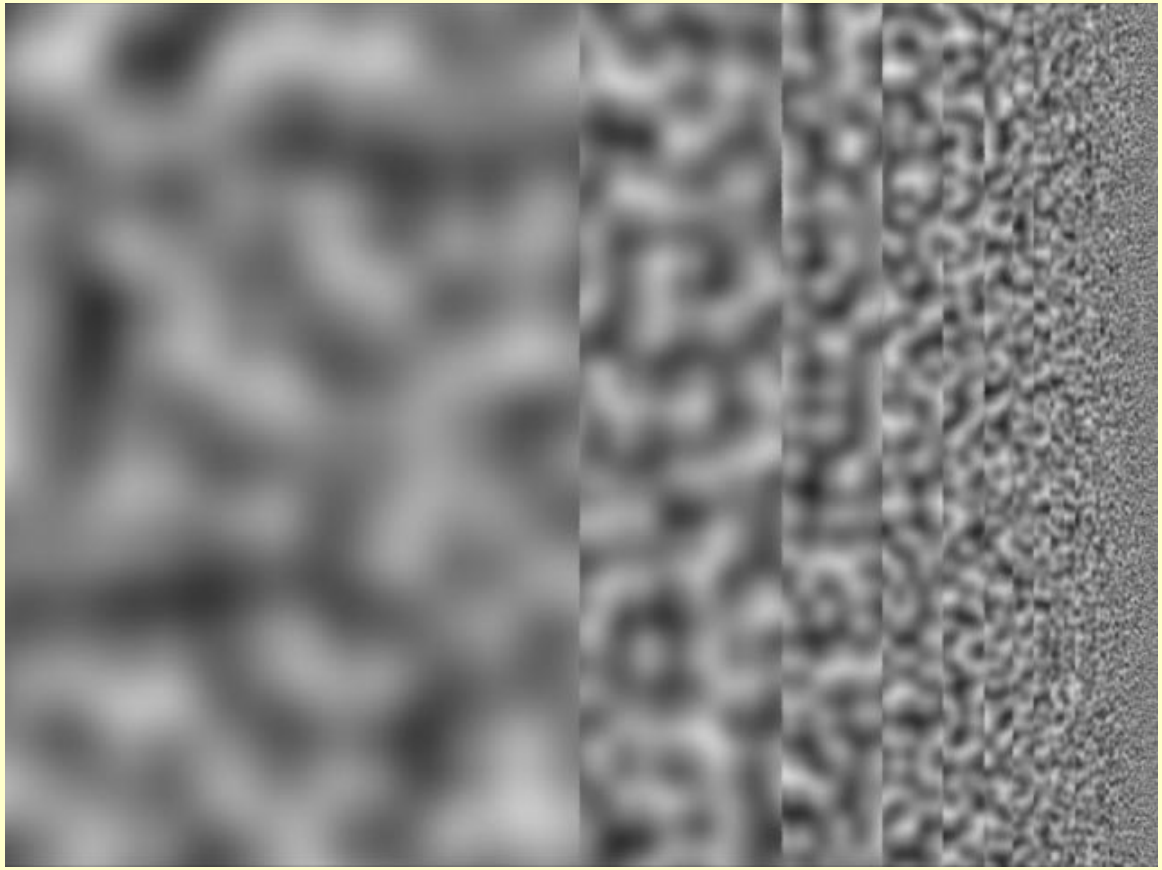
C Code

```
#include “ri.h”
RtPoint Square[4]={
    {1.4,1,1},{1.4,-1,1},{-1.4,1,1},{-1.4,-1,1}};
int main() {
  float noisescale = 4;
  RiBegin(“square.rib”);
  RiDeclare(“sc”,”uniform float”);
  RiWorldBegin();
  RiSurface (“noisetest”, “sc”, &noisescale, RI_NULL);
  RiPatch (RI_BILINEAR, RI_P, (RtPointer) Square, RI_NULL);
  RiWorldEnd();
    RiEnd();
    return 0;
}
```

Simple RenderMan Example

SL Code

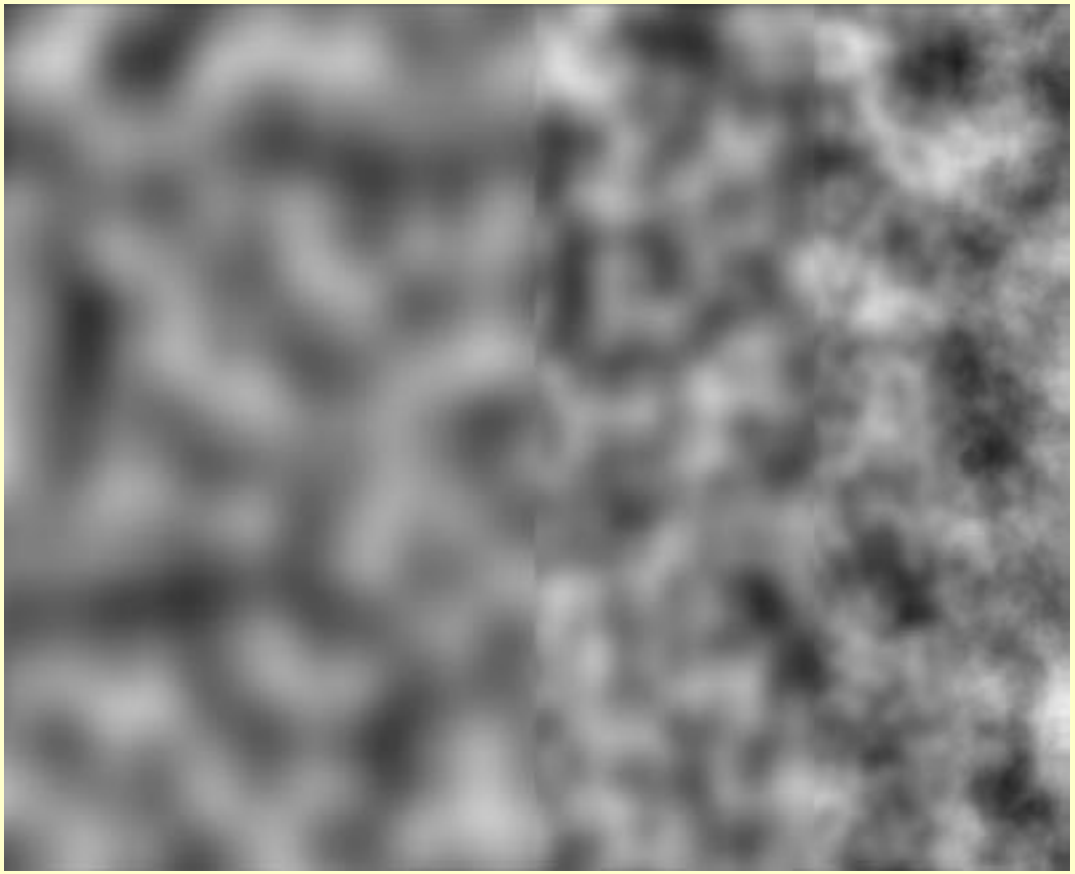
```
surface noisetest(float sc=1) {
  Ci = float noise(floor(1/t)*sc*P);
}
```



Fractional Brownian Motion (fBm)

Combine octaves, scaled by 1/f

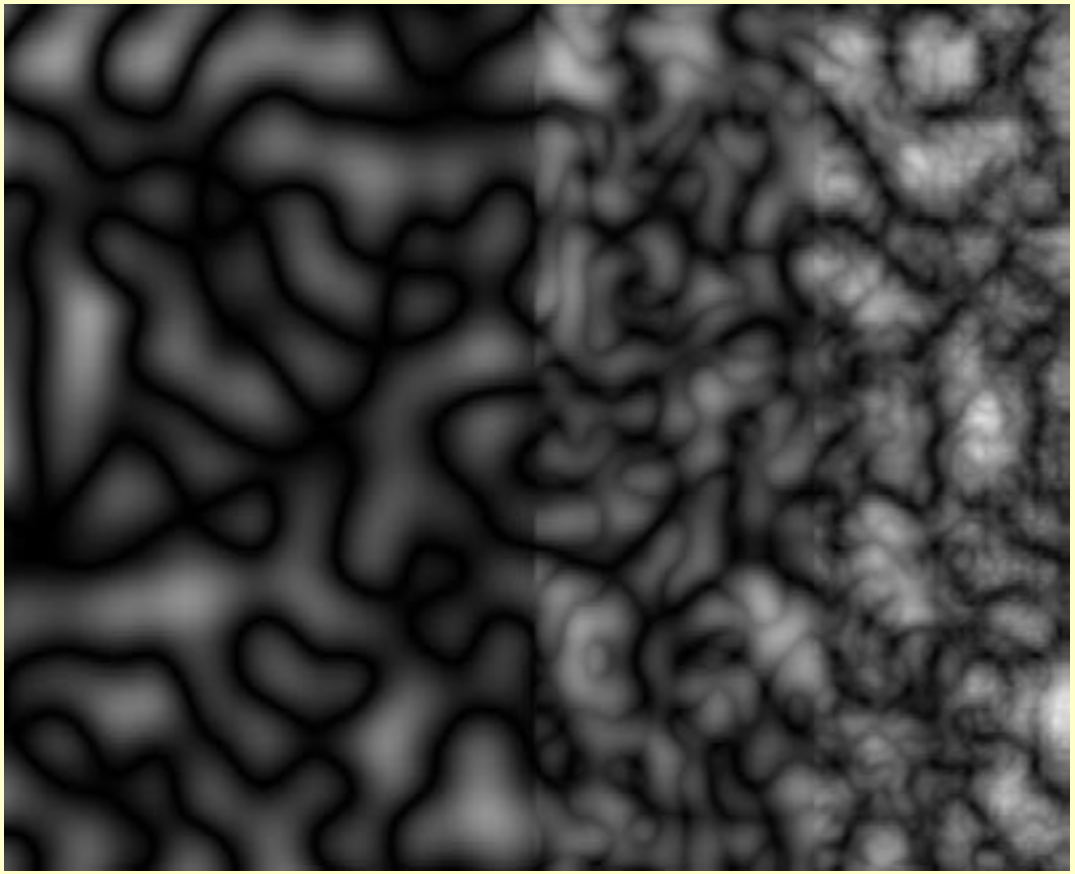
```
for(f=1; f<=floor(1/t); f*=2)
  Ci += (float noise(f*sc*P)-.5)/f;
Ci += .5;
```



Turbulence

fBm using abs(noise)

```
for(f=1; f<=floor(1/t); f*=2)
  Ci += abs(float noise(f*sc*P)-.5)/f;
Ci += .5;
```



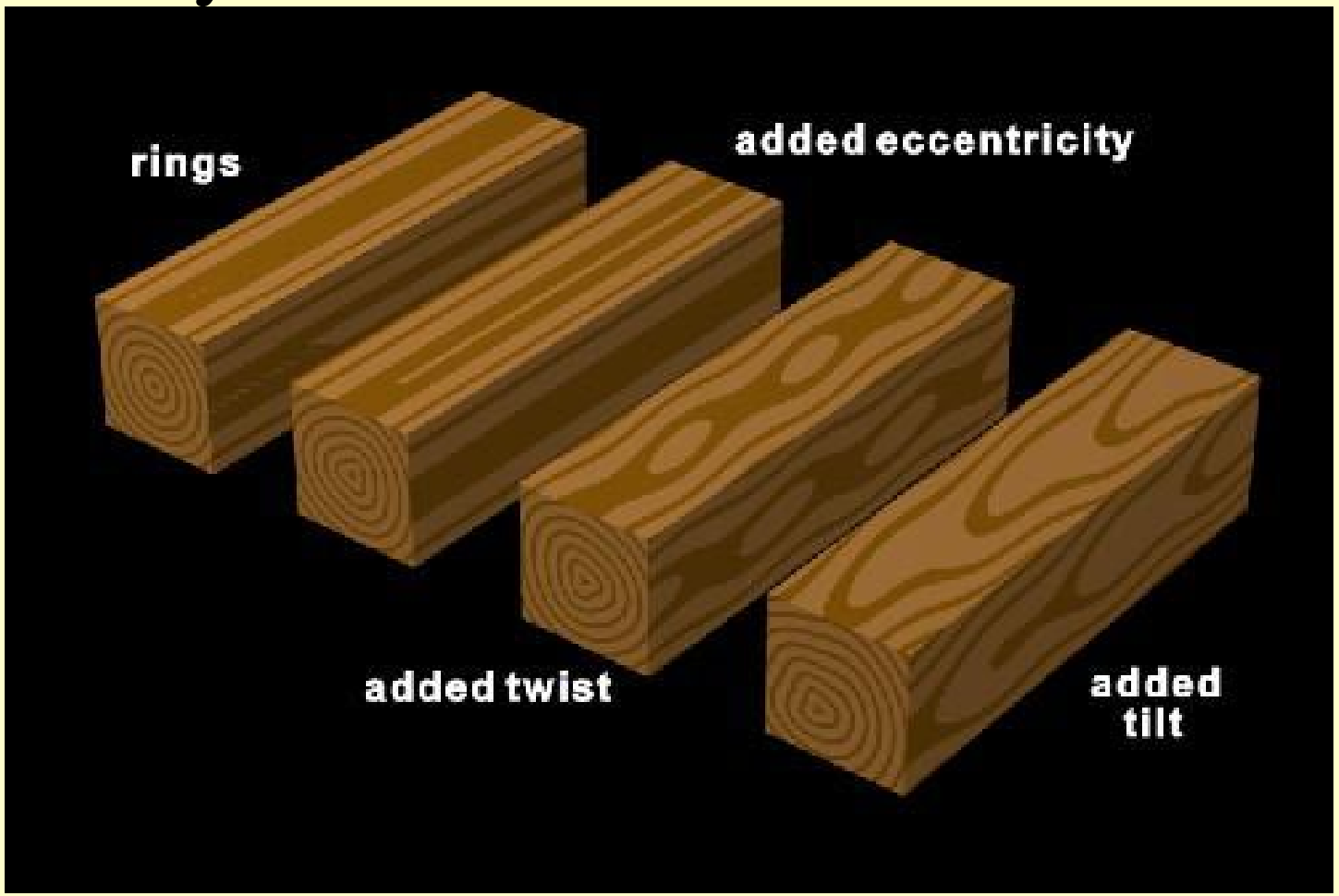
Marble

```
texture(xcomp(P)+turbulence(P))
```

Wood

Concentric rings of dark & light wood

Perturbed by noise



Volume/Cloud

Solid ellipse

Noise modifies density

More Atmospheric Effects