



Procedural Bump Mapping and Noise

Code and images from Ebert, David S., editor, *Texturing and Modeling: a Procedural Approach*. 1994

Johns Hopkins Department of Computer Science
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



Bump Mapping - Computing N'

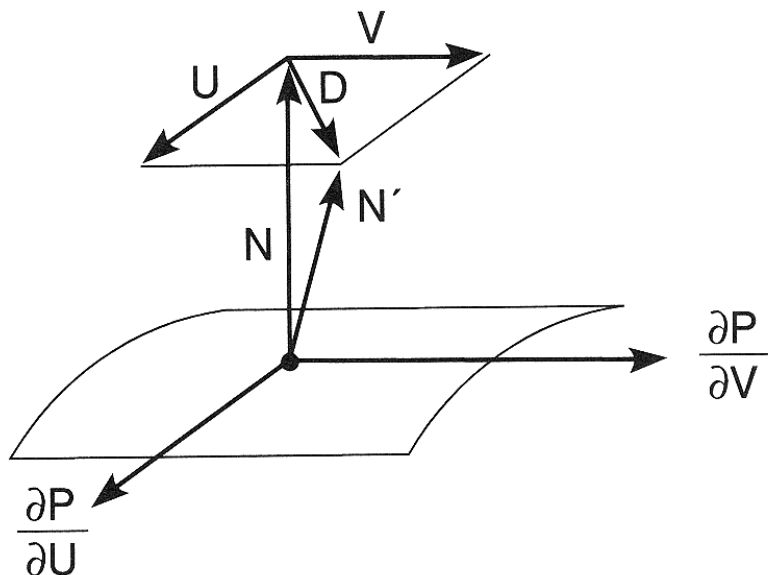


Figure 17. The geometry of bump mapping.

$F(u,v) =$
bump height function

$P(u,v) =$
surface position

$$U = \frac{\partial f}{\partial u} (\mathbf{N} \times \frac{\partial P}{\partial v})$$

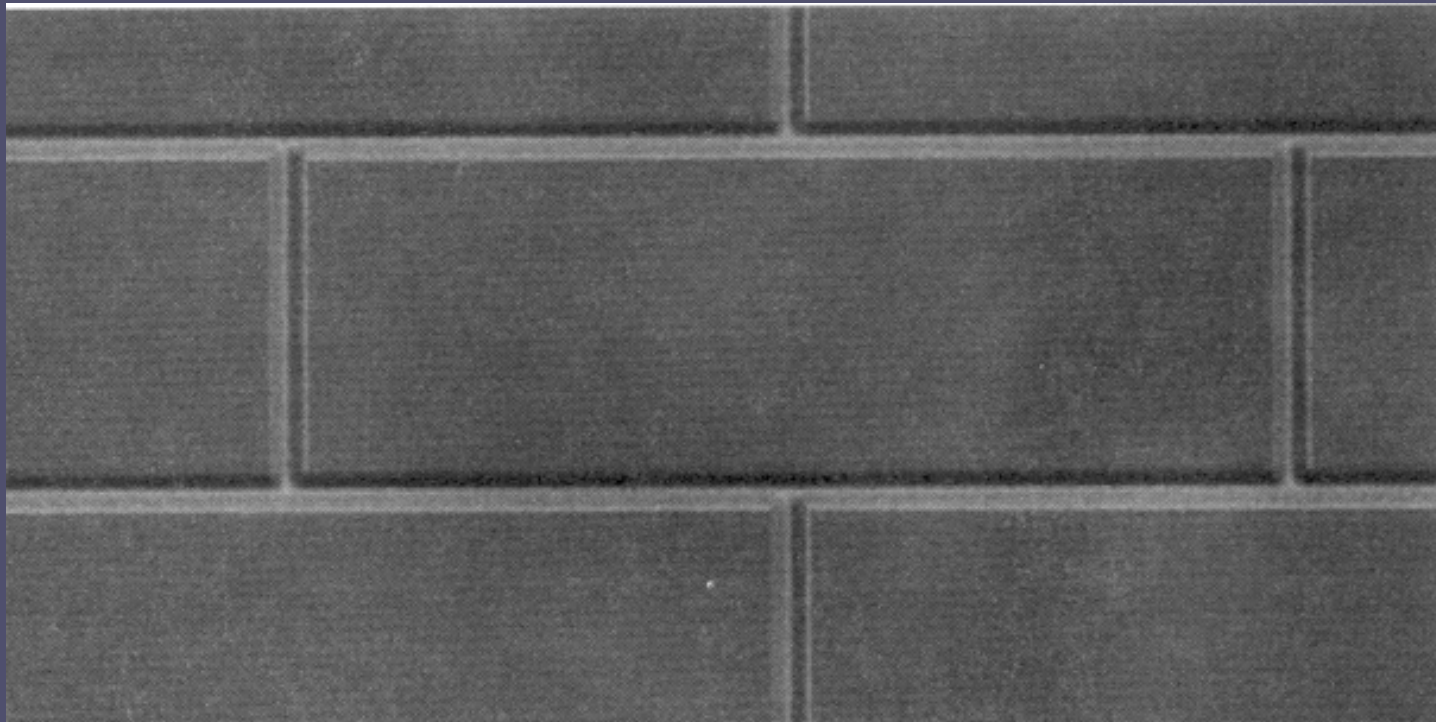
$$V = -\frac{\partial f}{\partial v} (\mathbf{N} \times \frac{\partial P}{\partial u})$$

$$\mathbf{D} = \mathbf{U} + \mathbf{V}$$

$$\mathbf{N}' = (\mathbf{N} + \mathbf{D}) / |\mathbf{N} + \mathbf{D}|$$



Bump-Mapped Brick





Example - Bumped Brick

Describe height function in terms of texture coordinates

Using built-in RenderMan functions:

- **displace point along normal according to height**
 - **find partial derivatives of new surface with respect to texture coordinates**
 - **cross the partials to get vector normal to new surface**
-



Without Special Assistance

Compute $\partial P/\partial u$ and $\partial P/\partial v$ analytically according to surface geometry (e.g. sphere)

OR

- Evaluate P at 4 nearby points by varying u and v slightly, then approximate partial using differences

Compute $\partial f/\partial u$ and $\partial f/\partial v$ analytically according to height function

Apply preceding formulas



Bevelling Effects

Nice ridges along edges of geometric figures

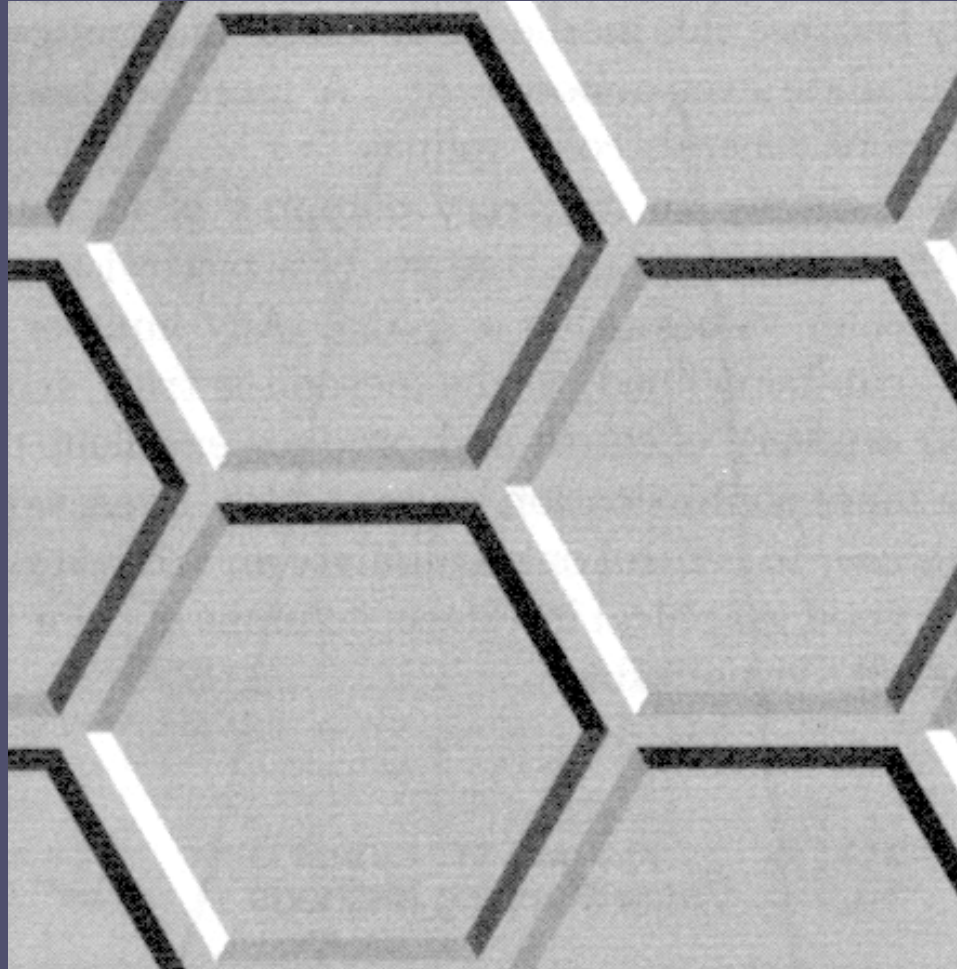
Parameters:

- **Total ridge and plateau widths**
- **slope at top and bottom of ridge**

Use perpendicular direction to closest edge as D (to add to normal), and scale according to ridge function



Bevelling





Noise Functions

Break up regularity

Enable modelling of irregular phenomena



White Noise

Sequence of random numbers

Uniformly distributed

Totally uncorrelated

- no correlation between successive values

Not desirable for texture generation

- Too sensitive to sampling problems
 - Arbitrarily high frequency content
-



Ideal Noise for Texture Generation

Repeatable pseudorandom function of inputs

Known range $[-1, 1]$

Band-limited (maximum freq. about 1)

No obvious periodicities

Stationary and isotropic

- statistical properties invariant under translation and rotation



Lattice Noise

Low pass filtered version of white noise

- Random values associated with integer positions in noise space
- Intermediate values generated by some form of interpolation
- Frequency content limited by spacing of lattice



Generating a Lattice

Generate a fixed-size table of random numbers

Hashing function indexes into the table to get value at any lattice point



Example Lattice Indexing

```
#define TABSIZE      256

#define TABMASK      ( TABSIZE - 1 )

#define PERM(x)      perm[ (x) & TABMASK ]

#define INDEX(ix,iy,iz) \
    PERM((ix)+PERM((iy)+PERM(iz)))
```

perm contains random permutation of integers in
[0, TABSIZE - 1]



Value Noise

**Create additional table of random values
(in range $[-1,1]$)**

**Index table according to permutation-based
INDEX function just presented**

(see sample code handout)



Interpolation Schemes

Linear interpolation -

- not really smooth enough

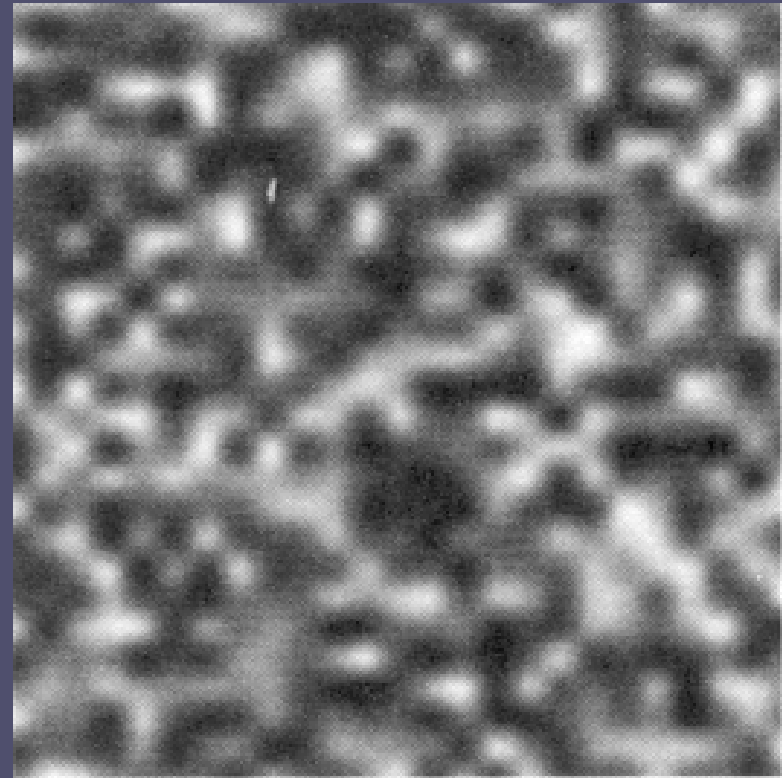
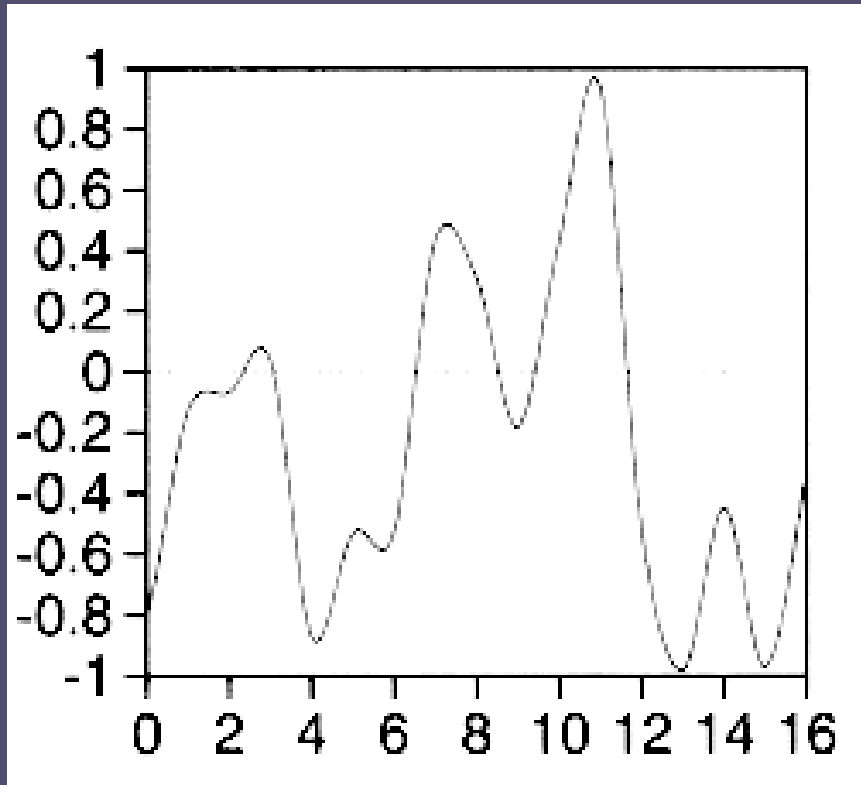
Quadratic or cubic spline interpolation

- may still have some artifacts resulting from grid layout

Convolution with radially symmetric filter kernel



1D and 2D Value Noise





Gradient Noise

Store direction vector at each lattice point

Noise values at lattice point is zero

Computing intermediate values:

For each neighboring lattice point

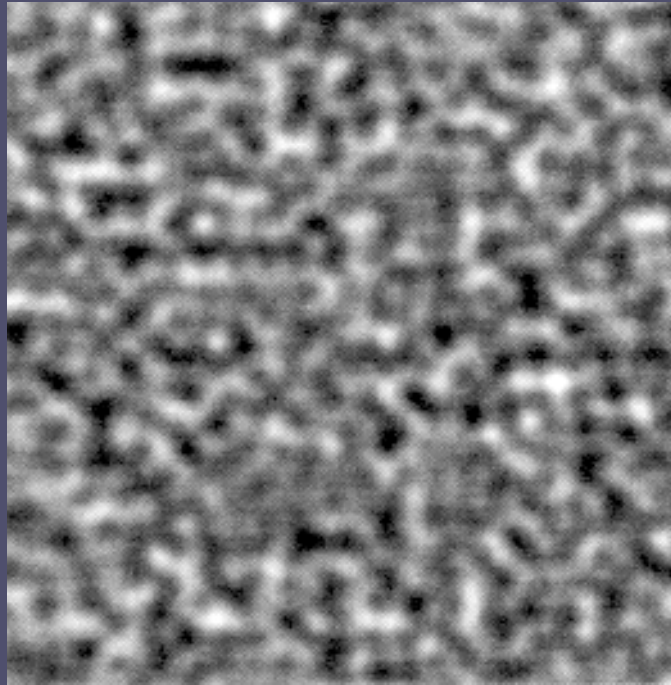
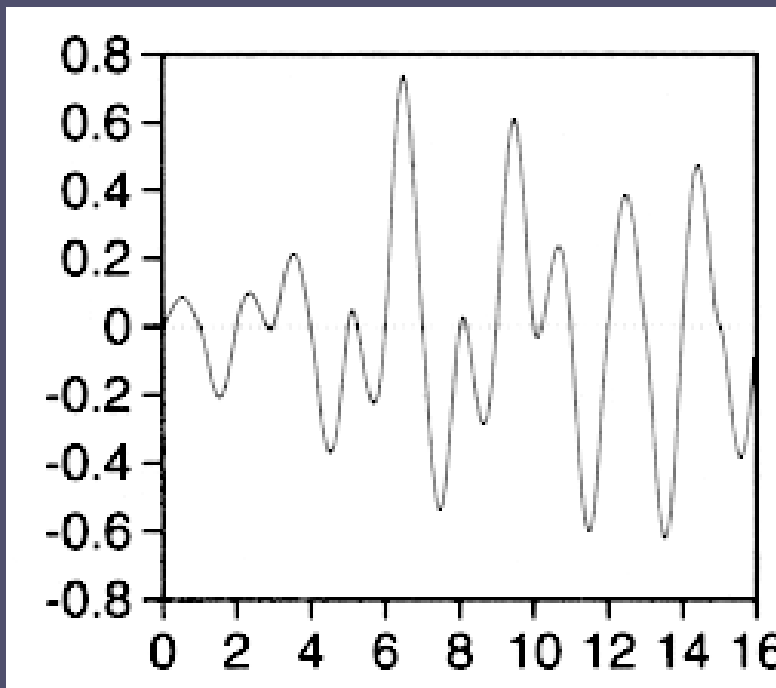
compute displacement along direction

Linearly interpolate between resulting 8 values
to get final value

(see sample code handout)



1D and 2D Gradient Noise





Value vs. Gradient Noise

Both noises have limited frequencies

Value noise slightly simpler to compute

Gradient noise has most of the energy in the higher frequencies

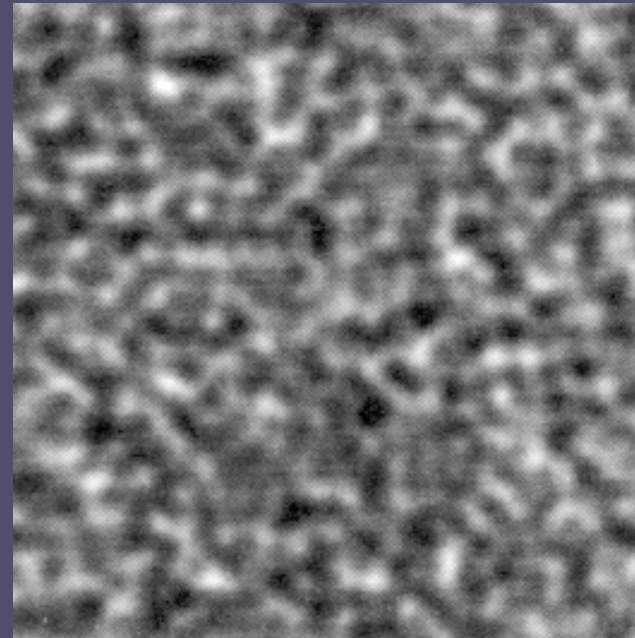
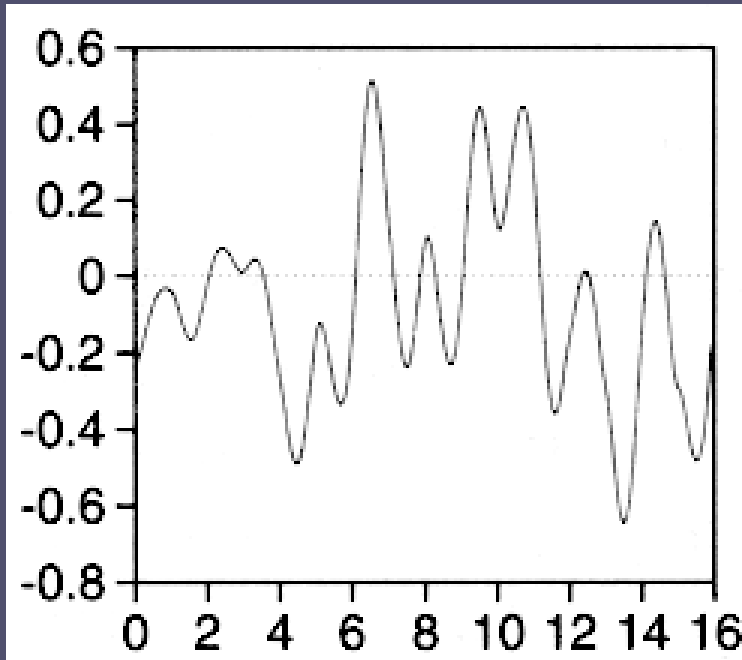
- **forced zero crossings**

Gradient noise has regularity because of zero crossings



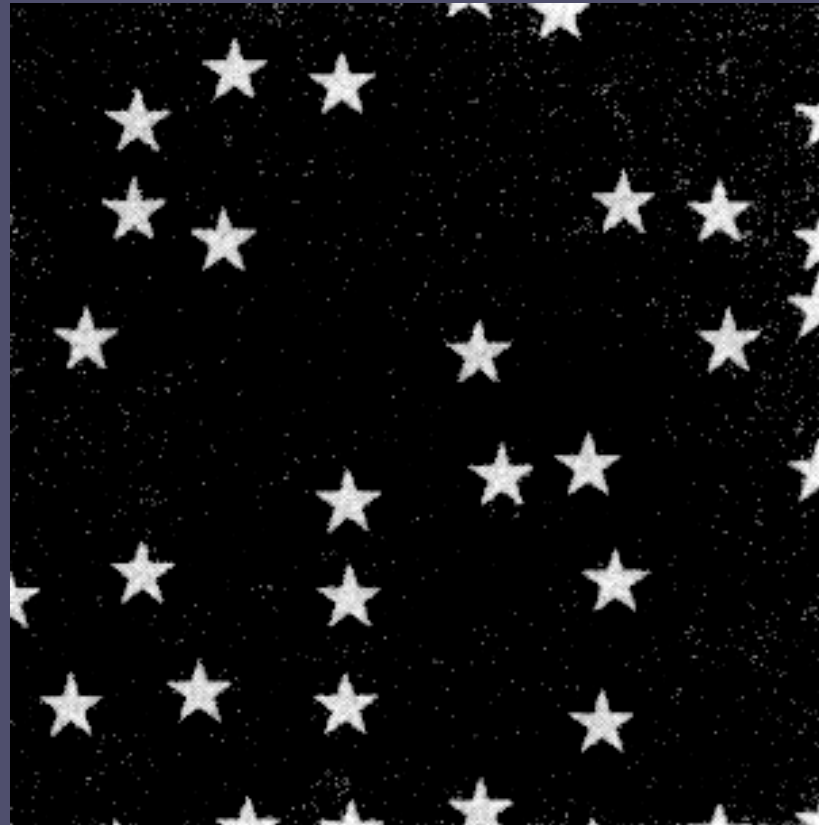
Value Gradient Noise

Weighted sum of value and gradient noises





Example - Star Wallpaper





Example - Star Wallpaper

Divide 2D texture space into uniform grid

Decide whether or not to place a star in each cell

Perturb position of star within each cell

To render a point on surface, check nearby cells for stars which may cover point

(see code handout)



Example - Perturbed Texture





Example - Perturbed Texture

Use noise function to apply perturbation to texture coordinates

Look up image texture (or generate procedural texture) using modified coordinates

(see code handout)



Example - Blue Marble



Marble vase (right) from Foley, van Dam, Feiner, and Hughes. *Computer Graphics: Principles and Practice*.



Example - Blue Marble

Use 3D position to compute 3D texture coordinates

Accumulate noise functions at several frequencies

- one type of **spectral synthesis**

Use sum of noise to determine marble color

- using spline interpolation between colors



Modelling Gases

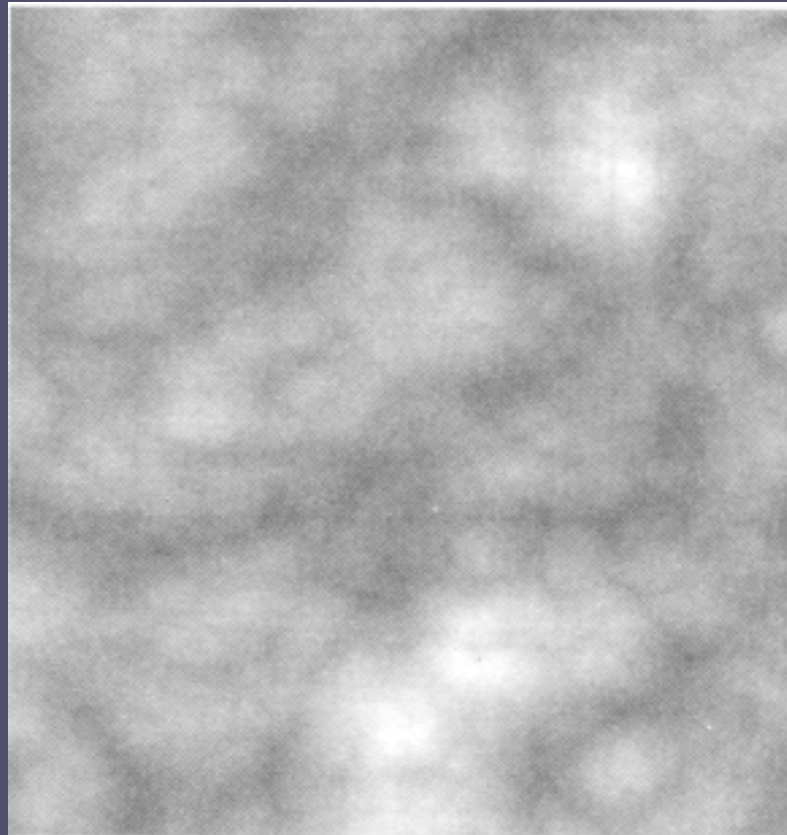
Represent 3D gas as density volume

Use turbulence function as basic gas description

Adjust turbulence by raising it to a power, taking the sine, etc.



Turbulence





Turbulence

```
float turbulence(point Q)
```

```
{
```

```
    float value = 0;
```

```
    for (f= MINFREQ; f < MAXFREQ; f *= 2)
```

```
        value += abs(noise(Q*f))/f;
```

```
    return value;
```

```
}
```

(in practice, don't use a round number like 2)



Basic gas

```
float gas(point P, float max_density,  
          float exponent)  
{  
    float turb, density;  
    turb = turbulence(pt);  
    /* or turb = (1 + sin(turbulence(pt)*PI))/2 */  
    density =  
        pow(turb*max_density, exponent);  
    return density;  
}
```



Placing and Shaping Gas

Place some primitive shape to contain density volume

Attenuate density to account for dissipation

Steaming teacup example

- **attenuate according to distance from center of tea surface**
- **attenuate according to height above tea surface**



Steaming Tea Cup



Johns Hopkins Department of Computer Science
Course 600.456: Rendering Techniques, Professor: Jonathan Cohen



More Turbulence Uses

Add variation to color of surface textures

Use as bump mapping function to add variety to normals