CMSC 435
Texture

Penny Rheingans

UMBC

Texture Mapping

- Def: mapping a function onto a surface; function can be:
  - 1, 2, or 3D
  - sampled (image) or mathematical function

Mapped Parameters

- Surface color (Catmull 74)
- Specular reflection (Blinn and Newell 76)
- Normal vector perturbation (Blinn 78)
- Specularity (Blinn 78)
- Transparency (Gardner 85)
- Diffuse Reflection (Miller and Hoffman 84)
- Shadows, displacements, etc (Cook 84)
- Local coord system (Kajiya 85)

Map Indices

- Surface parameters
- Ray direction
  - reflection/environment mapping
- Surface normal direction
  - diffuse reflection mapping
  - transparency/refraction mapping

Key Challenges

- Mapping function determination
- Resolution issues
- Texture design/capture

Mapping Functions

- Standard projecting functions
  - planar
  - cylindrical
  - spherical
- Mechanism
  - Two-stage mapping
  - Reverse projection
- Arbitrary

Two-stage Mapping

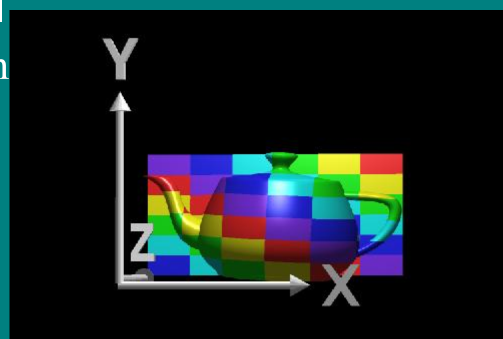
- S-mapping
  - map to simple 3D shape
  - intermediate surfs: plane, cylinder, cube, sphere
- O-mapping
  - map 3D texture onto surface
  - map entities: reflected view ray, surface normal, line through centroid, intermediate surface normal

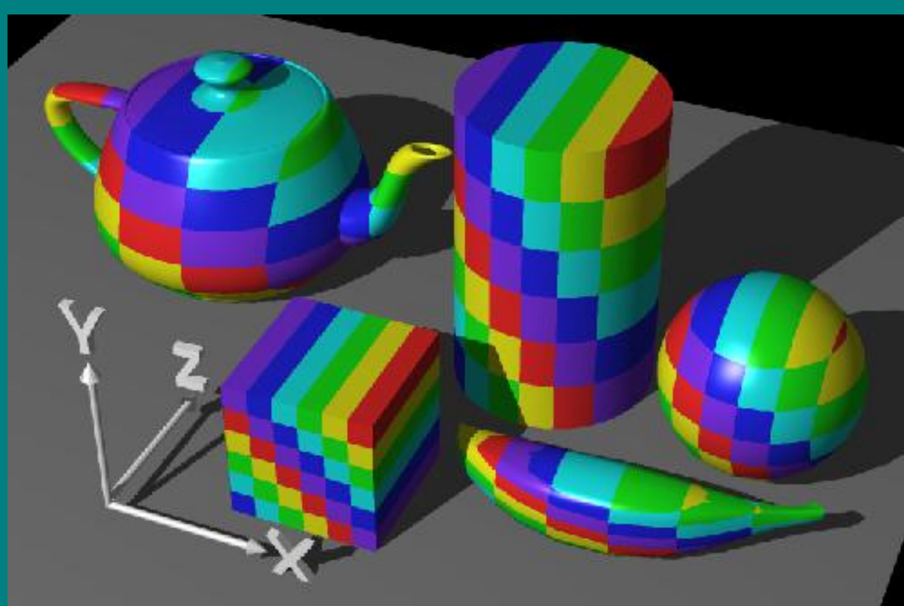
Planar Mapping

- For xy aligned plane

$$(u,v) = \left(\frac{x - x_1}{x_r - x_1}, \frac{y - y_1}{y_r - y_1} \right)$$

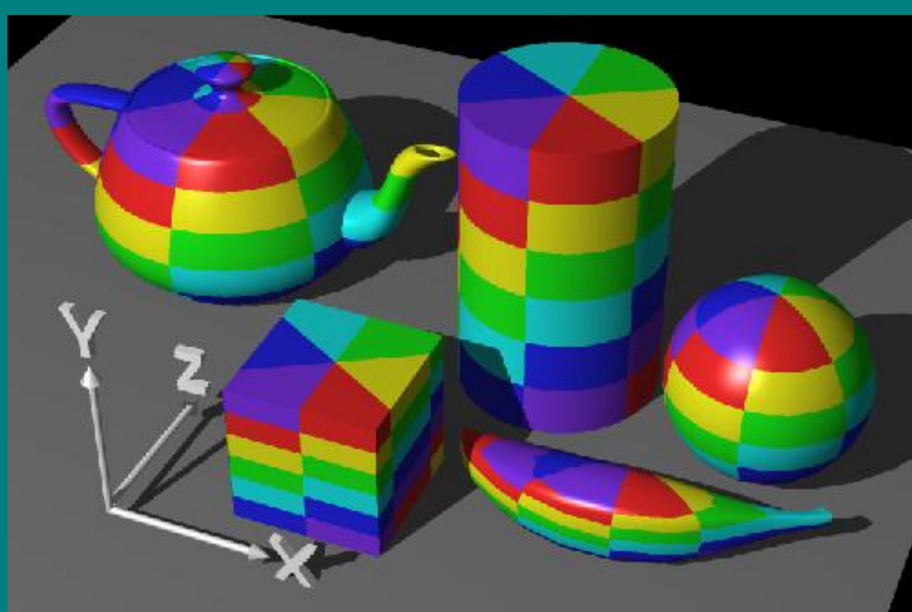
- Reverse projection





Cylindrical Mapping

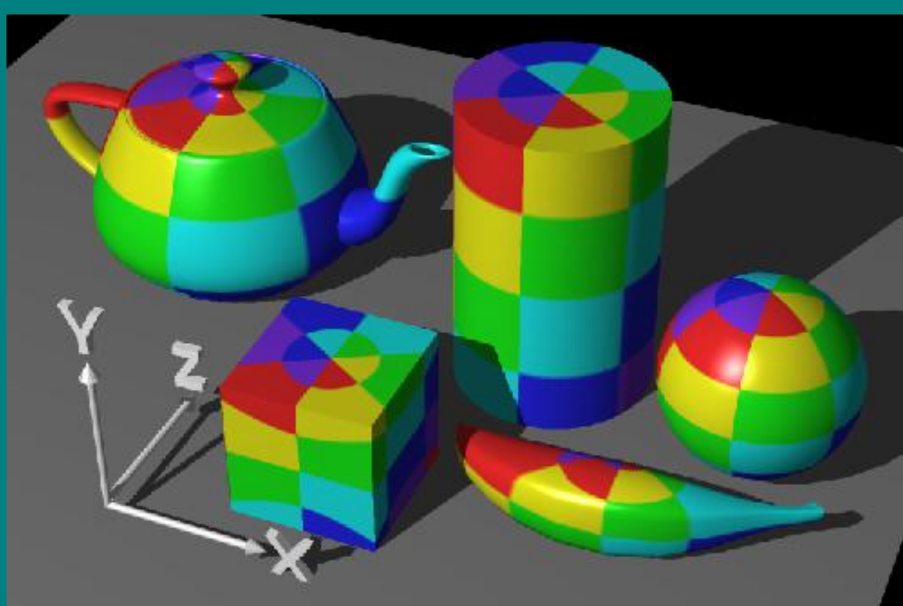
- For cylinder with point
  (rcosθ, rsinθ, hz)
- Texture coords
  (u,v) =(θ/2π, z)



Spherical Mapping

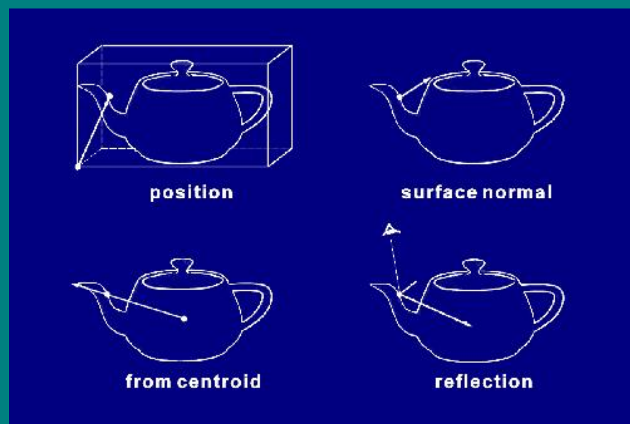
- For sphere with point
  $(r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)$
- Texture coords

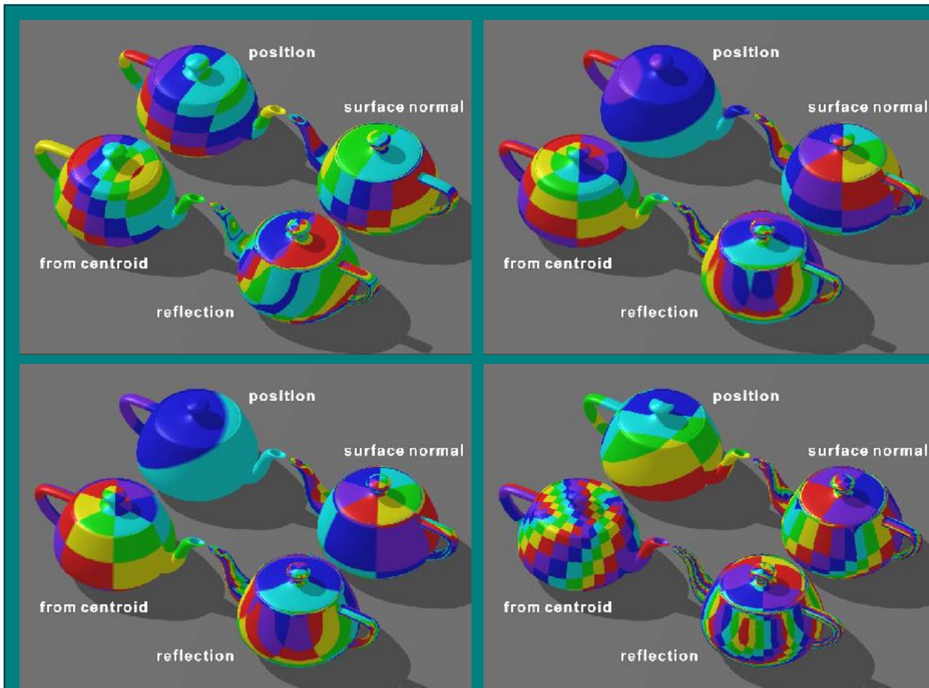
$$(u,v) = \left(\frac{\theta}{\pi/2}, \frac{\pi/2 - \phi}{\pi/4}\right)$$



Map Entity

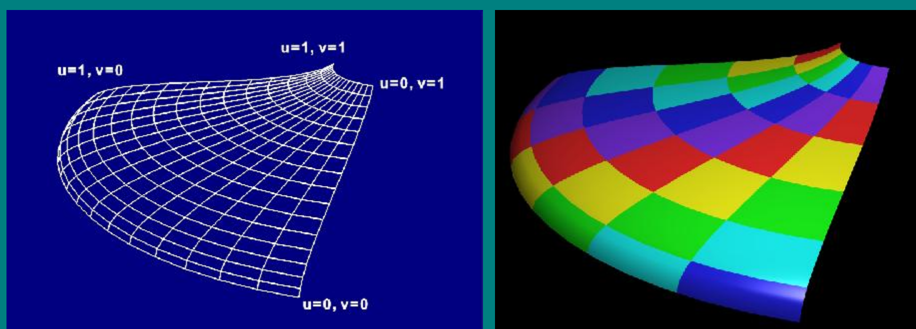
- What entity gives coords in map shape?

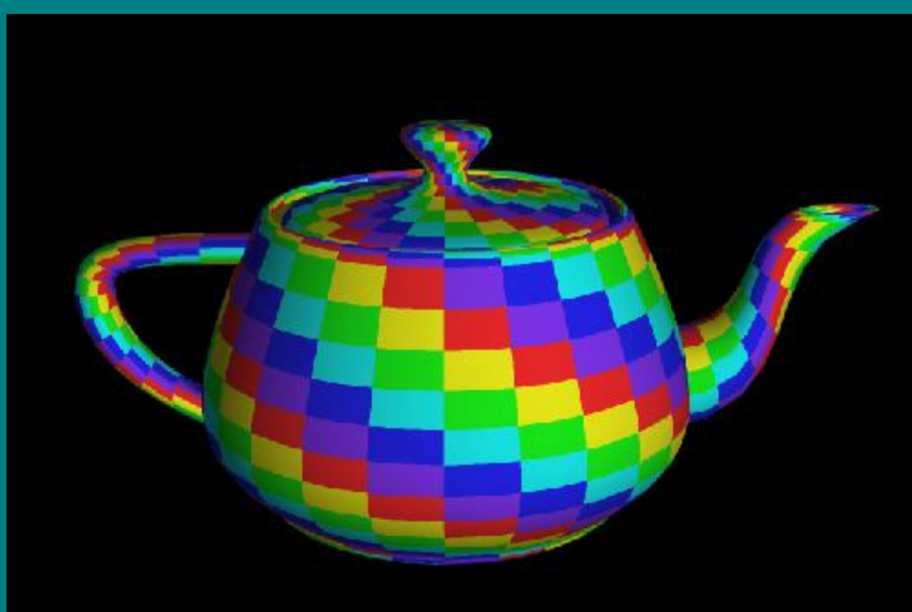




Mapping onto Parametric Patches

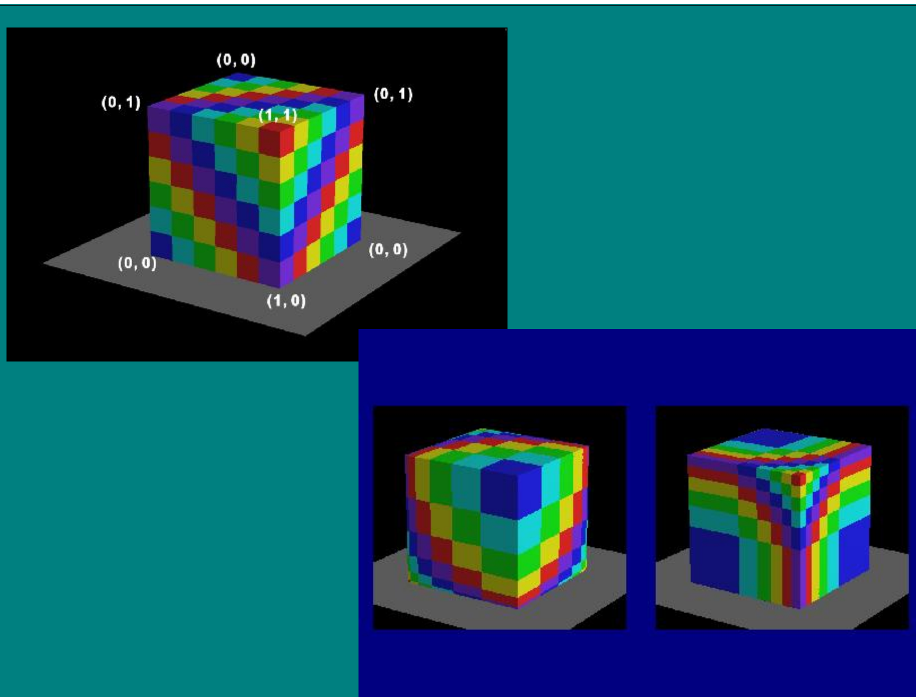
- Use scaled surface u,v parameters for texture u,v





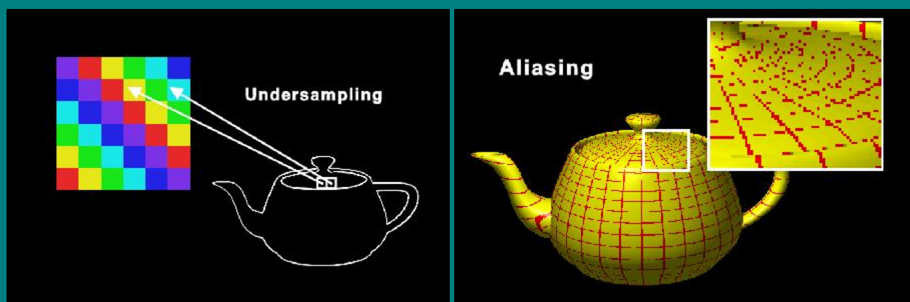
Mapping onto Polygons

- Like parametric surfaces, but use explicit vertex texture coords
- Interpolation issues
  - screen space interp results in errors from nonlinearity and lack of rotational invariance
  - use small pgons to minimize artifacts
- Correct solution: actual projection at each pixel



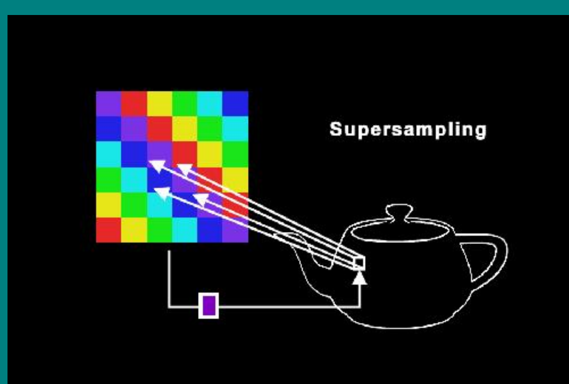
Texture Aliasing

- Undersampling of texture map leads to texture aliasing
- Oversampling can show limited texture res



Supersampling

- Sample texture multiple times per pixel and reconstruct

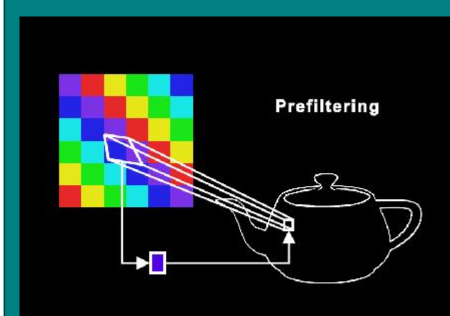


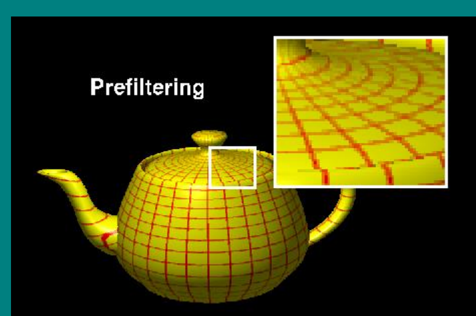
Filtering

- Steps
  - reconstruct continuous signal from samples
  - warp signal
  - low pass filter signal using convolution
  - resample at new resolution
- Filters
  - space-invariate
  - space-variant

Filtering

- Basic method (Catmull 78)
  - Project pixel pgon onto texture map
  - Average color over projected area





Filtering Types

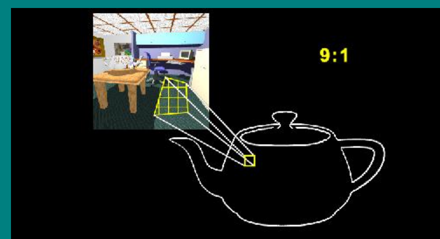
- Direct Convolution
  - average multiple samples from texture (usually selected in texture space)
- Prefiltering
  - construct multi-resolution copies of texture
- Fourier filtering
  - low pass filter texture in frequency space

Mipmappng

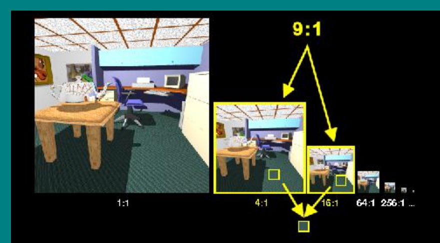
- Precalculate filtered maps at a range of resolutions (Williams 83)
- Higher memory requirements

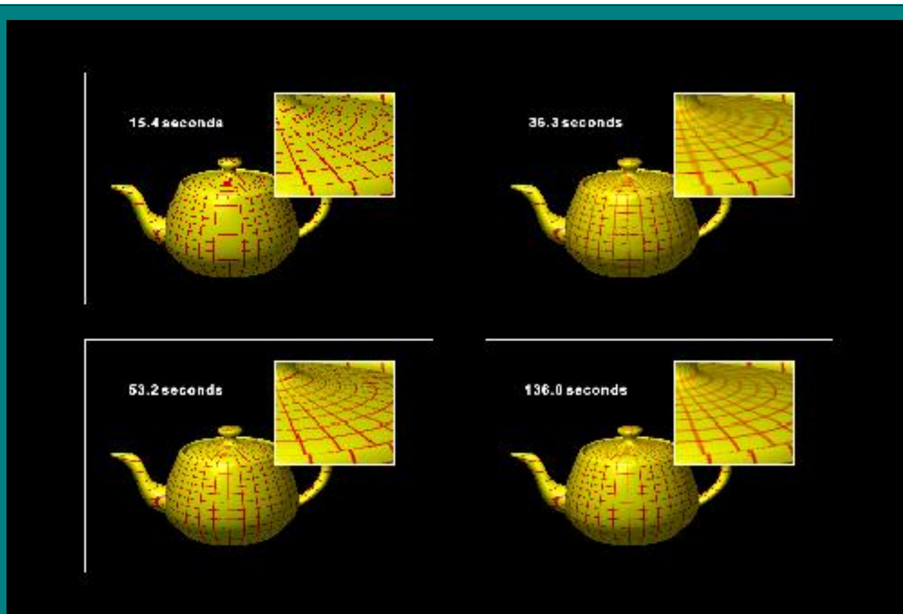
Mipmapping Process

- Compute pixel area in mipmap



- Average from two closest maps





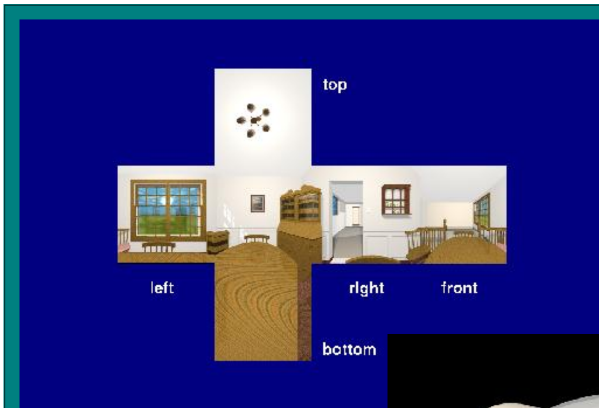
Anti-aliasing: none, mipmapped, supersampled, supersampling and mipmapping

Prefiltering Methods

- Pyramid/mipmaps
  – construct pyramid of different resolution maps
- Summed area tables
  – not constrained to square areas
- Repeated integration
  – generalization to higher order integration by repeated sampling

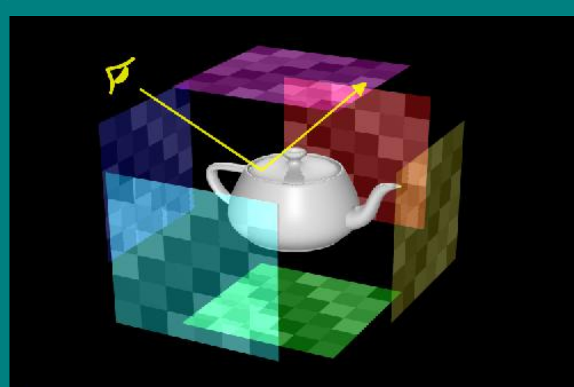
Reflection Mapping

- Look up reflections on an object from a map simulating surrounding environment

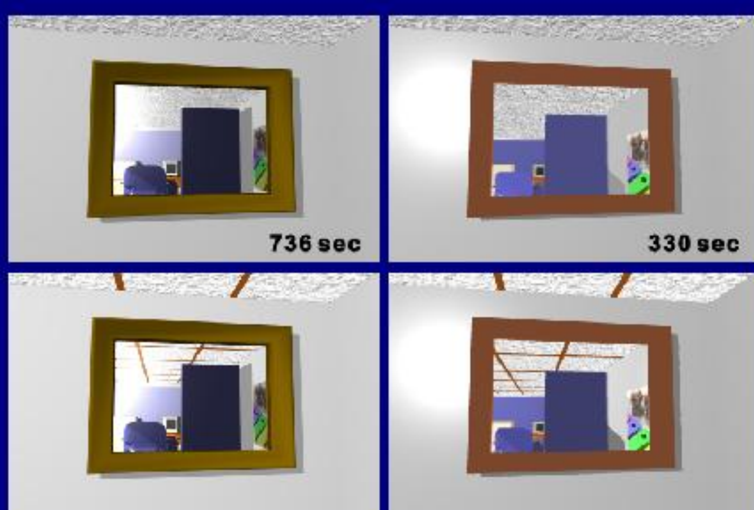


Environment Mapping

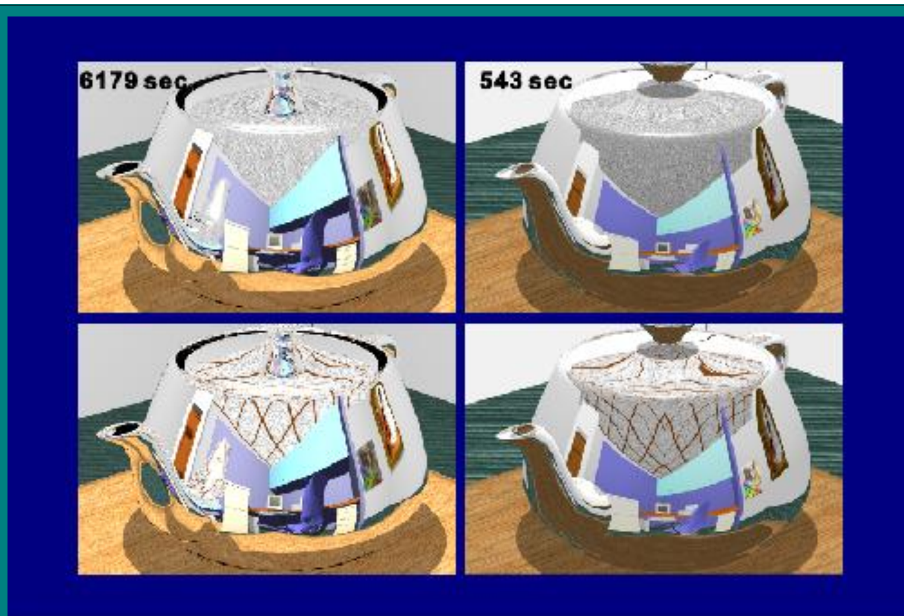
- Surround scene with maps simulating surrounding detail



Ray tracing Environment Mapping



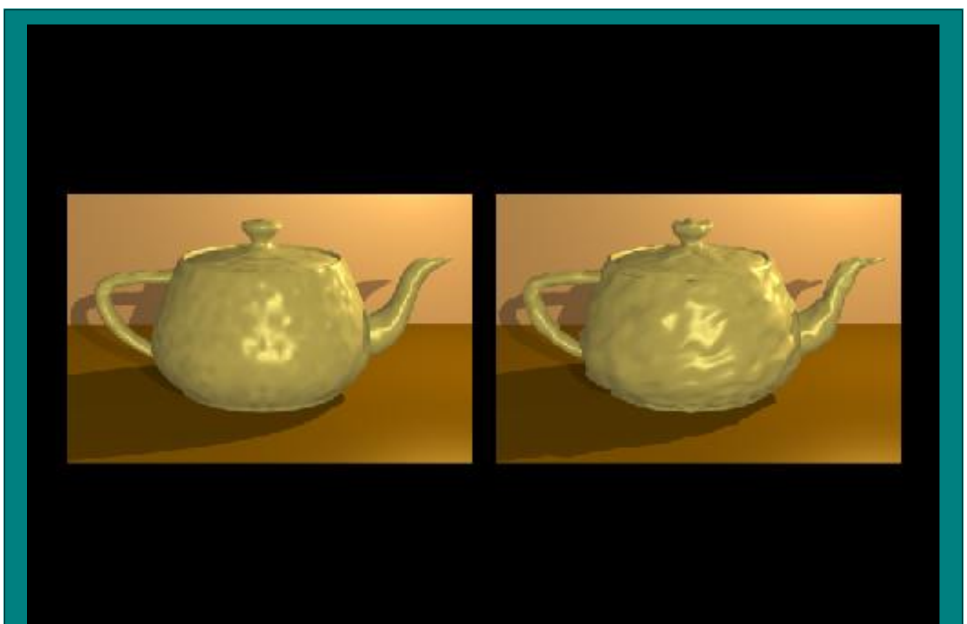
Ray tracing Environment mapping



Ray tracing Environment Mapping

Bump Mapping

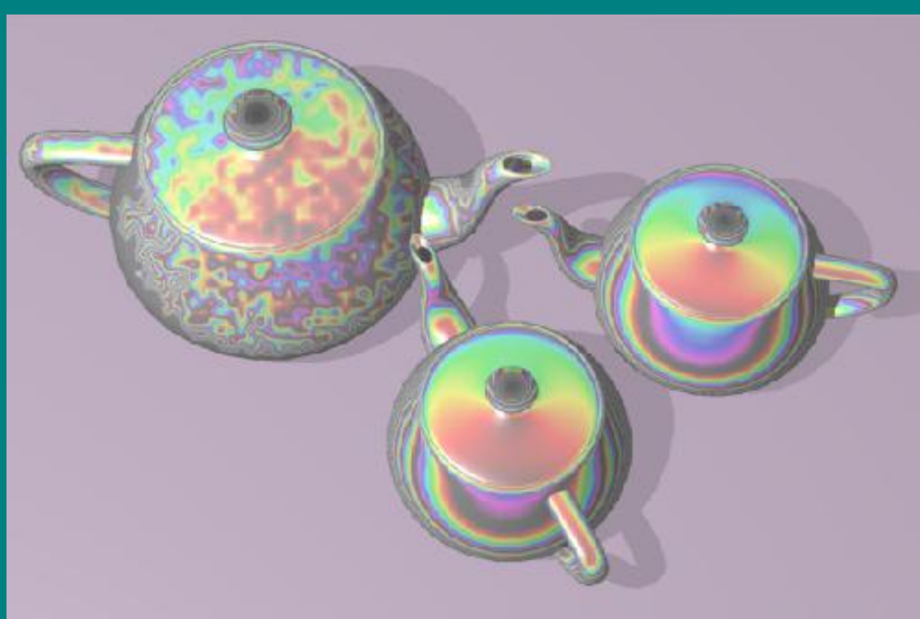
- Perturb surface normals to simulate shape variations



Bump mapping Displacement Mapping

Refraction Mapping

- Perturb refraction rays through transparent surface by disruption of surface normal

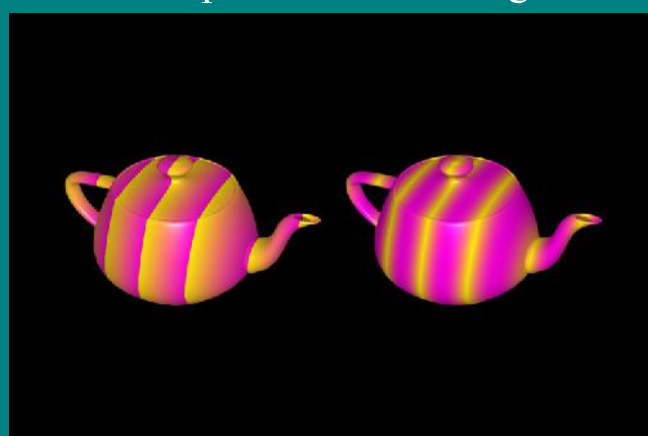


Procedural Approaches

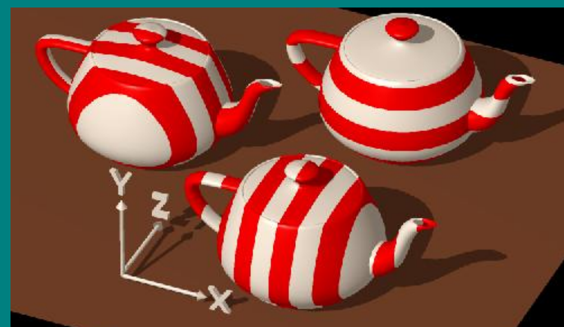
- Simple Functions
- Noise
- Statistical Synthesis from Samples
- Simulation
- Developmental

Simple Functions

- Simple functions can produce interesting effect
  - ramp
  - sine

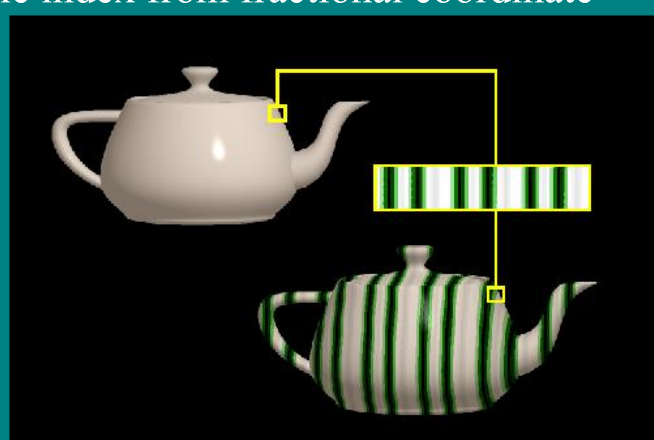


- Fractional coord



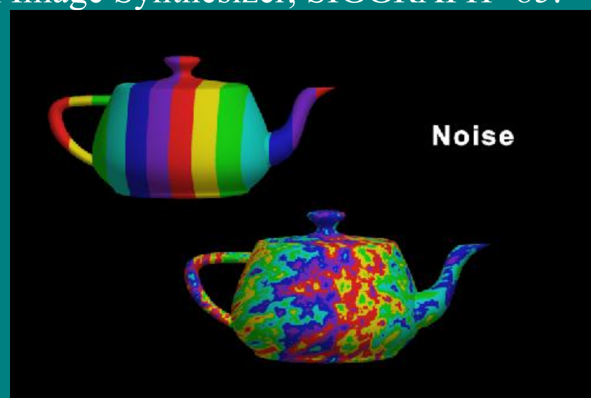
- 2D distance from center

- Color table index from fractional coordinate



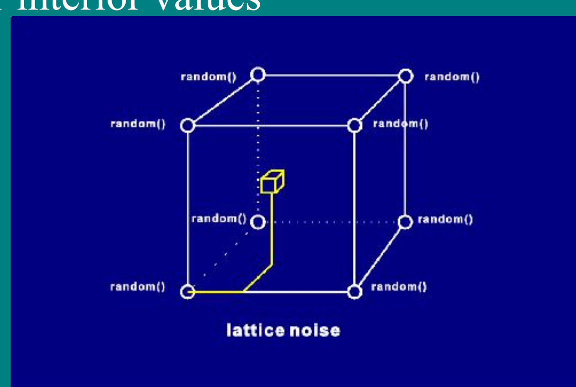
Noise Textures

- Randomness can create more natural textures
- Ken Perlin, An Image Synthesizer, SIGGRAPH '85.



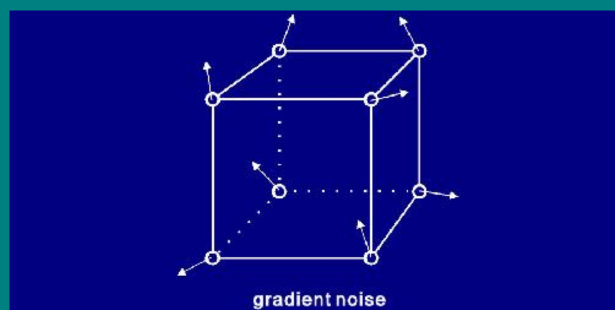
Lattice Noise

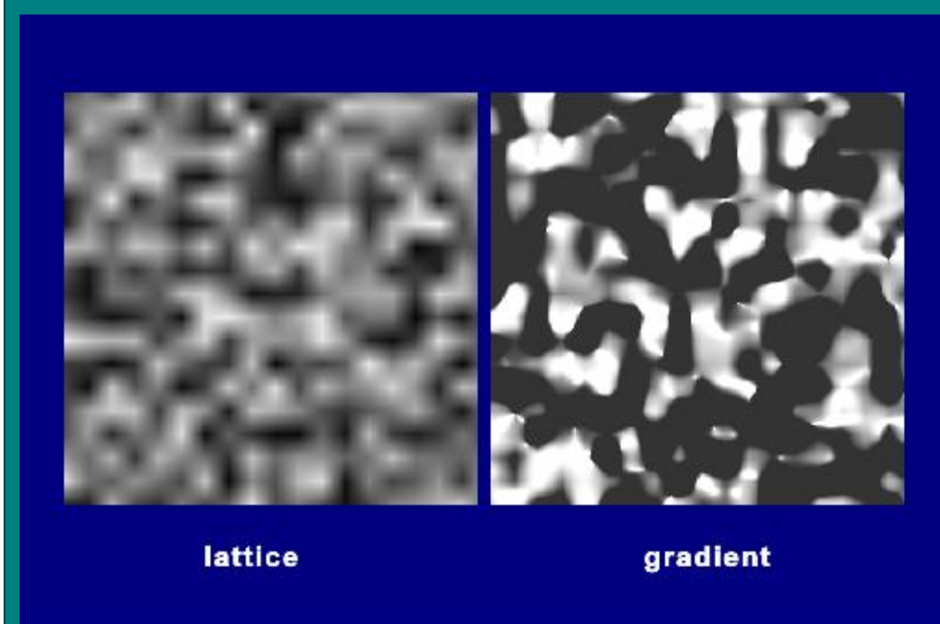
- Assign random numbers to lattice points
- Interpolate for interior values
- Perlin, '85

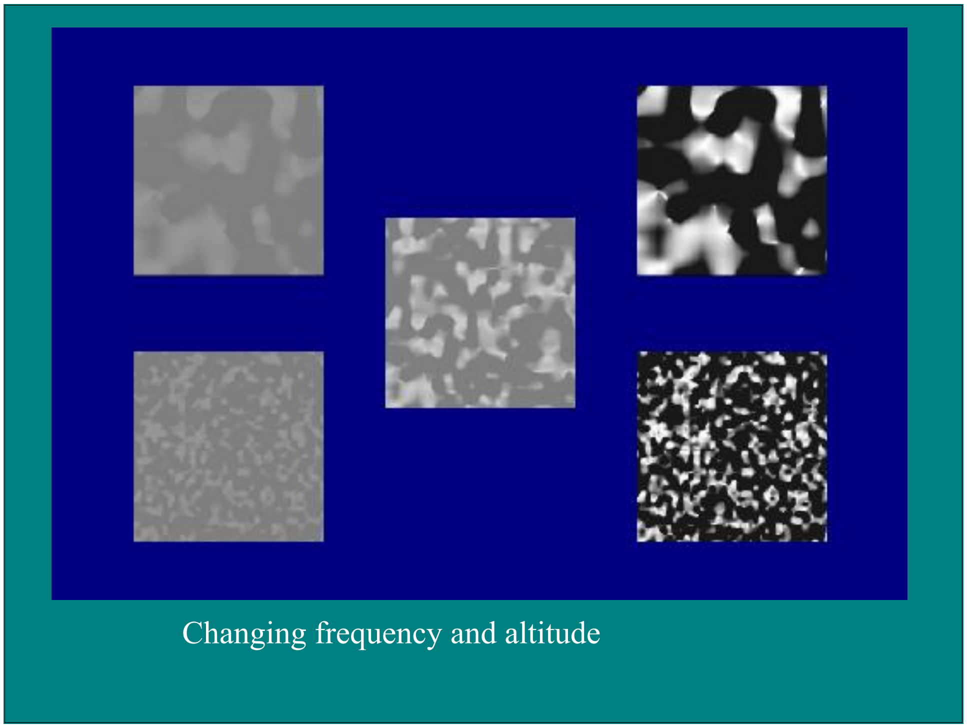


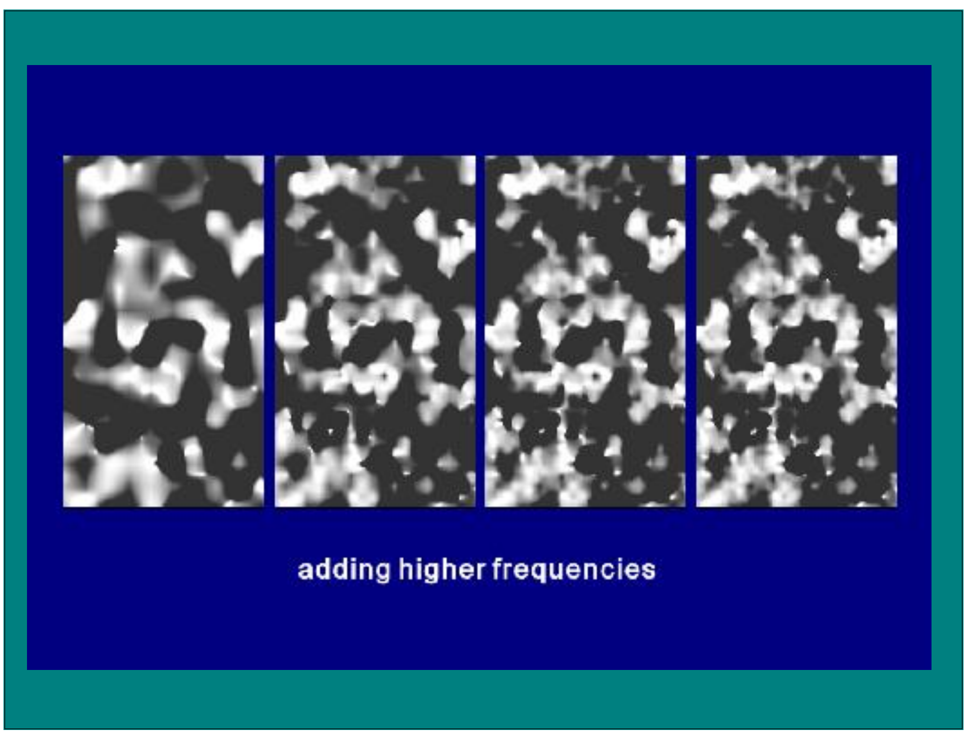
Gradient Noise

- Generates random vectors at lattice points
- Lattice points not perturbed
- Uses gradients for interior points
- Perlin '85






Changing frequency and altitude


adding higher frequencies

Texturing with Noise

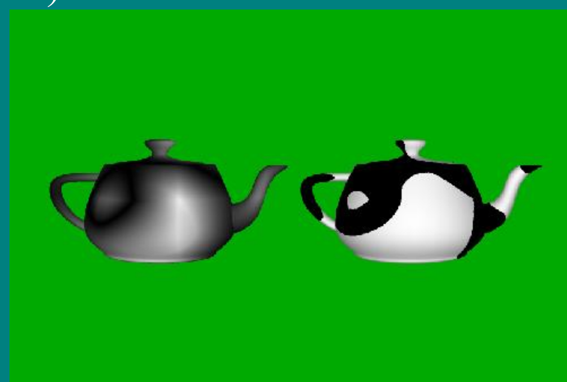
Grey = noise(x,y,z)

if (grey > threshold)

    white

else

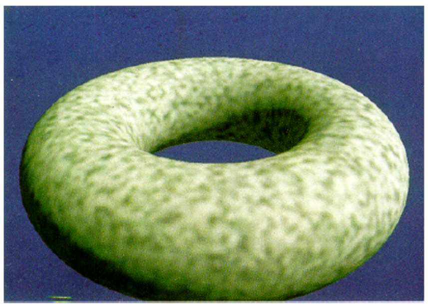
    black



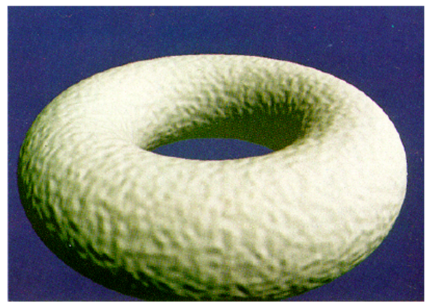
Using Noise to Perturb Textures

Procedural Properties

- Color
- Normal direction



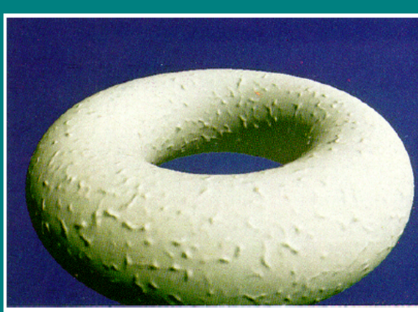
Spotted Donut



Bumpy Donut

Procedural Properties (cont.)

- Normal
- Color



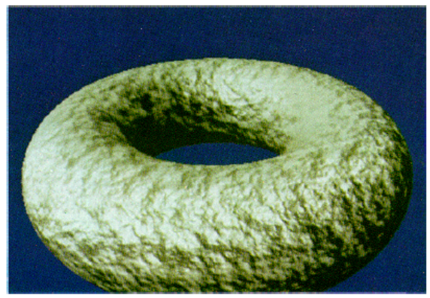
Stucco Donut

Disgusting Donut

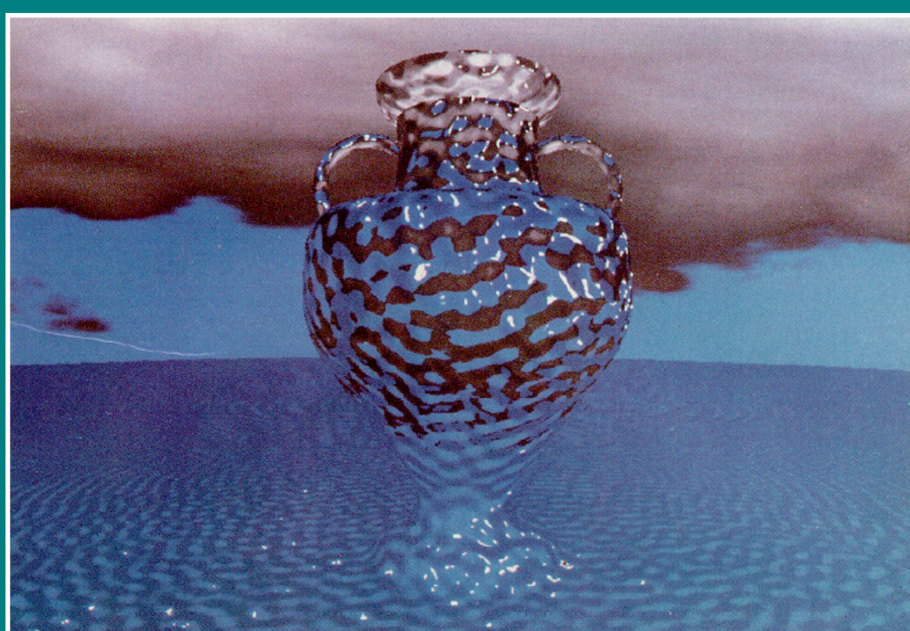
Procedural Properties (cont)

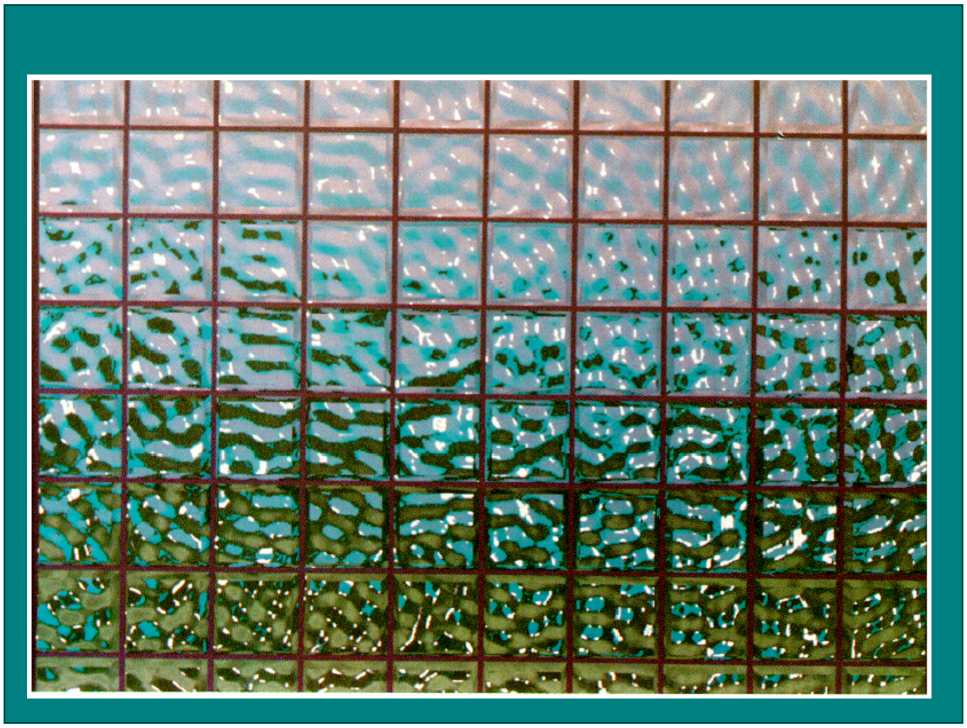
- Thresholded color
- Normal (turbulence)

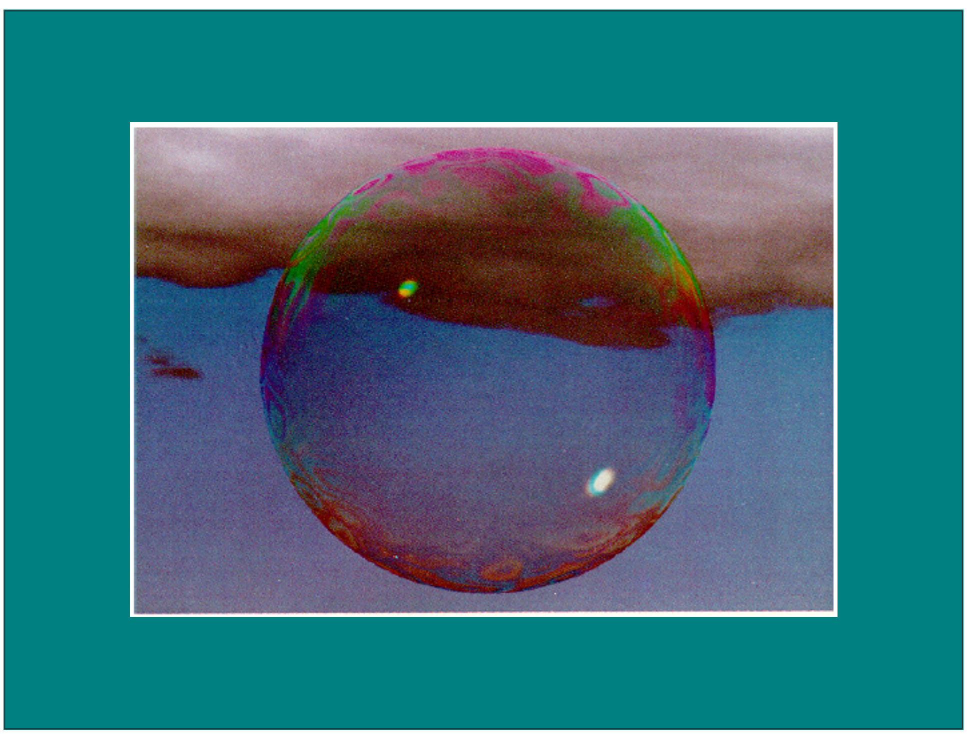
Bozo's Donut

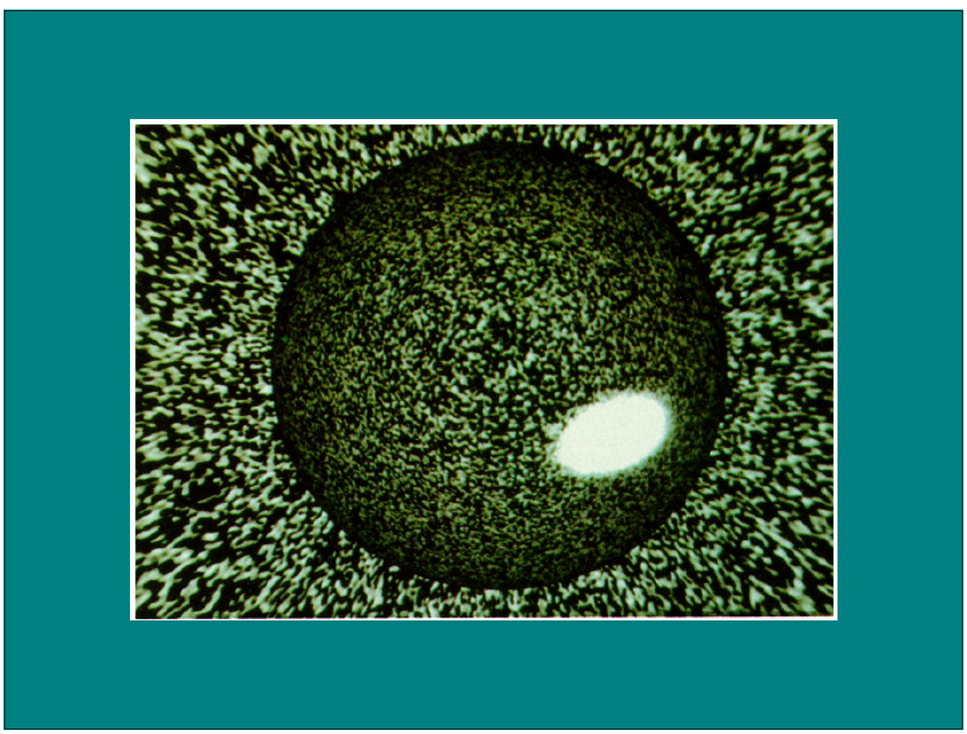


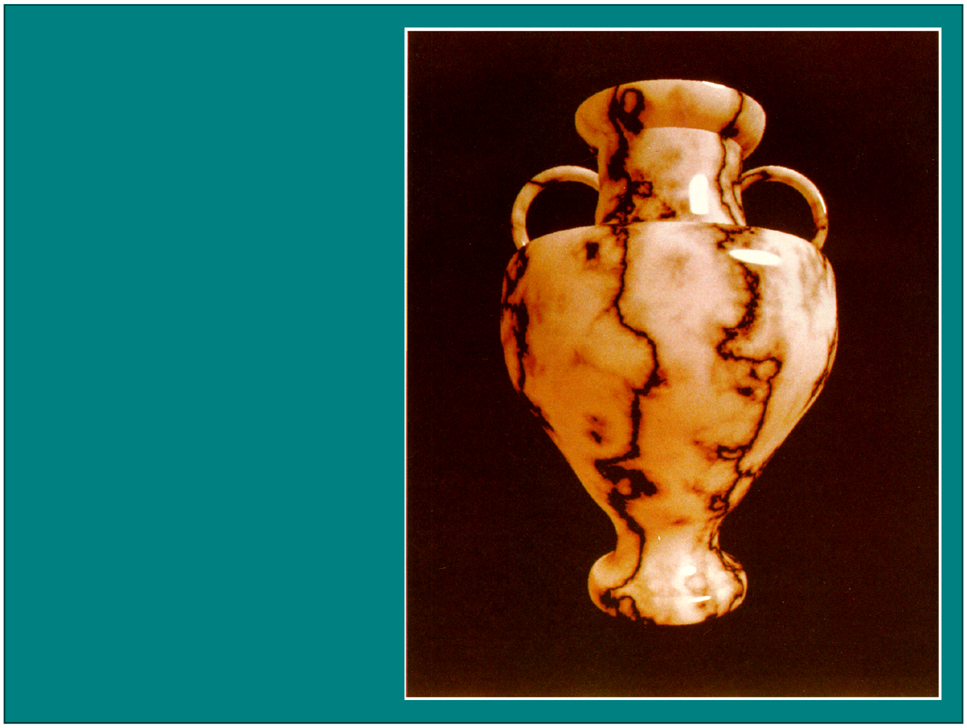
Wrinkled Donut

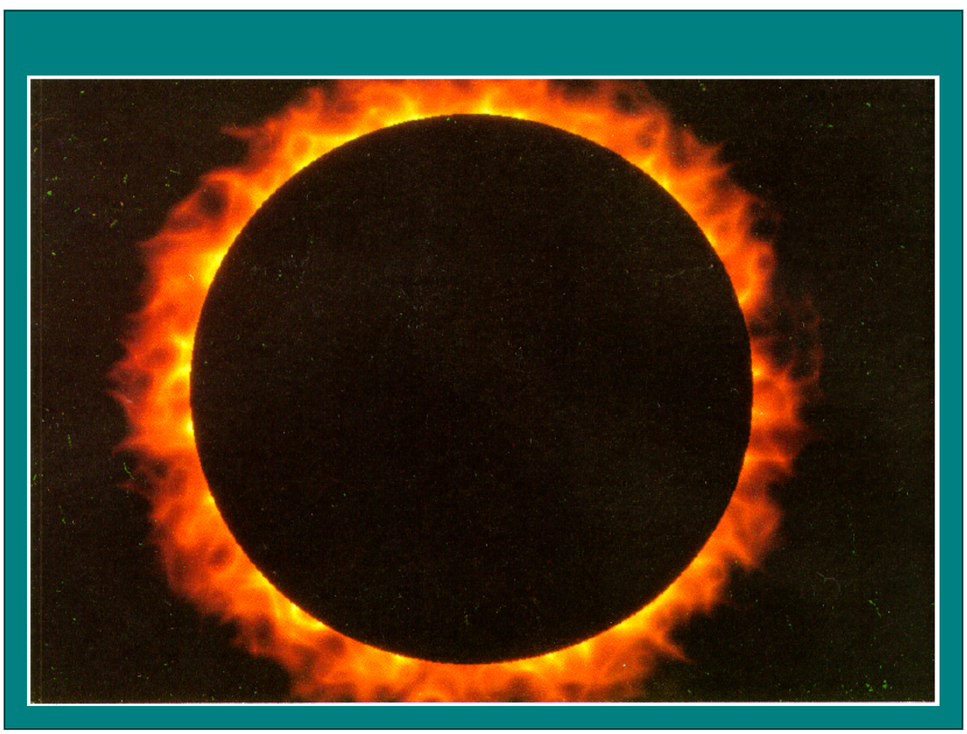


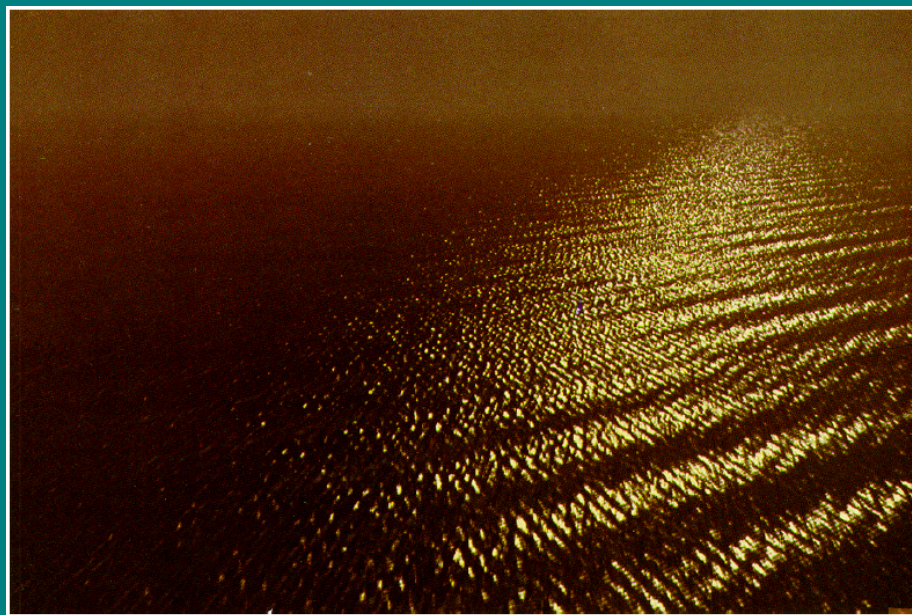












Hypertexture

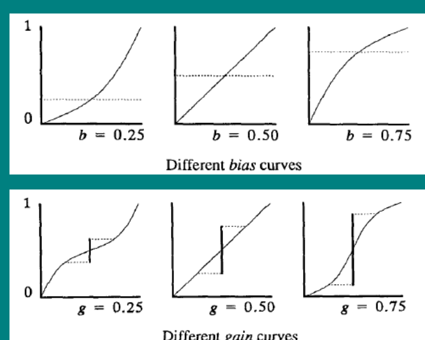
- Ken Perlin and Eric Hoffert, Hypertexture, SIGGRAPH '89.
- Extend 3D procedural noise textures to include opacity component to create volume models
  - object density function D(x)
  - Density modulation function (DMF) f_i

Boolean Operations

- Intersection
- Complement
- Difference
- Union

Base DMFs

- Bias
- Gain
- Noise
- Turbulence
- Arithmetic functions

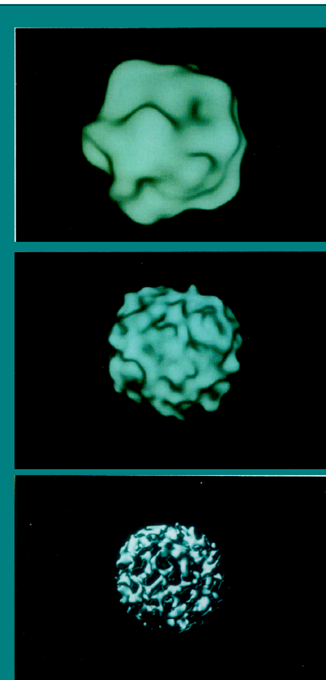


Basic Noise

- Basic noisy sphere

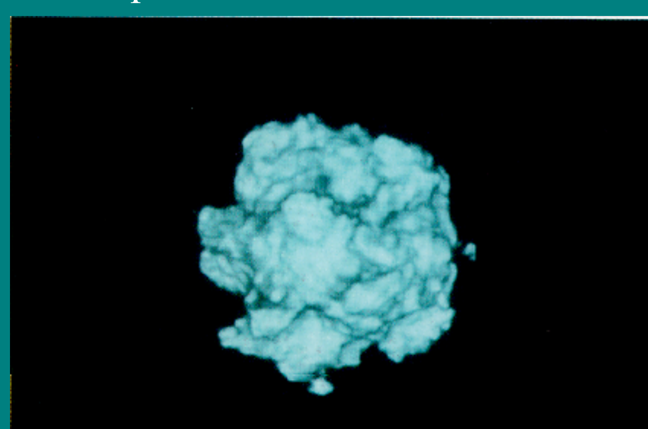
$$D(\mathbf{x}) = sphere(\mathbf{x}\,(1 + \frac{1}{f} noise(f\mathbf{x})))$$

- Vary
  - Frequency
  - Amplitude



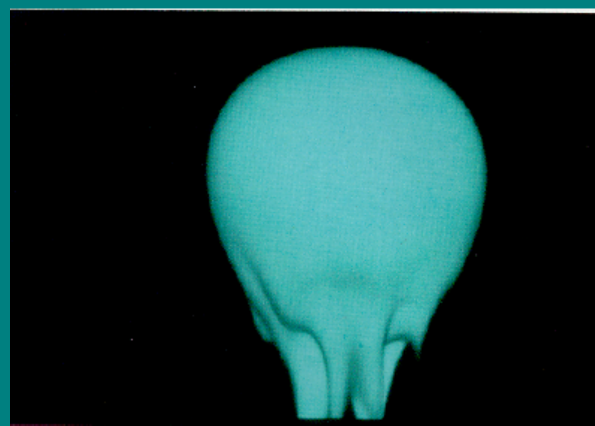
Turblence

- Mix different frequencies of noise



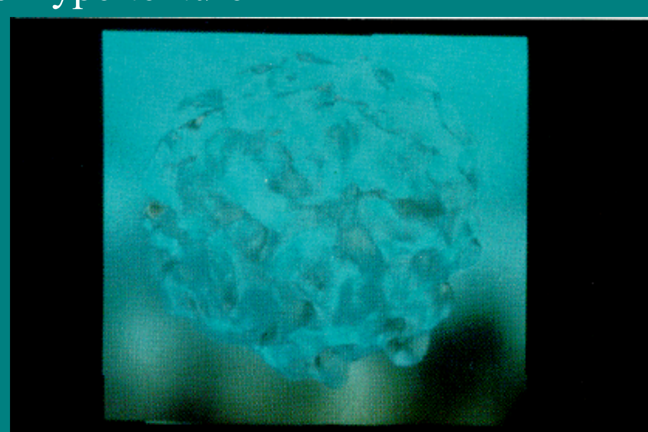
Shaped Noise

- Vary only single component



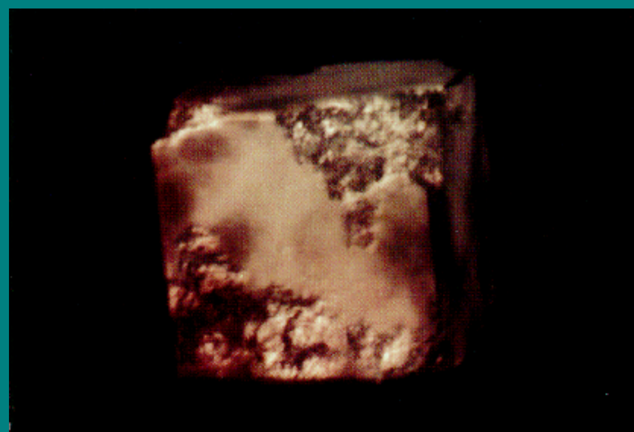
Transparency

- Refractive Hypertexture



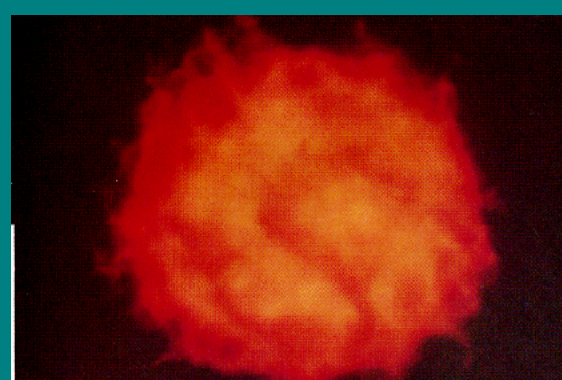
Erosion

- Boolean intersection of fractal sphere with cube



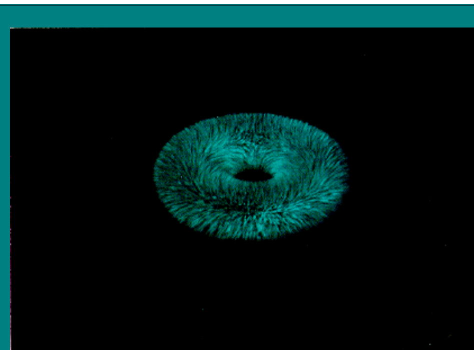
Fire

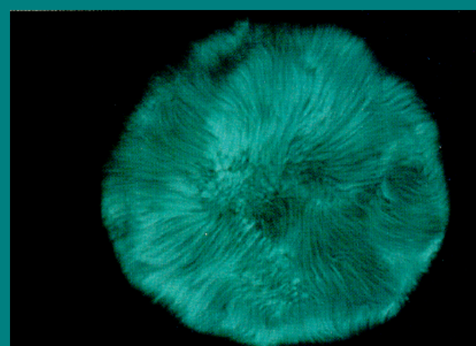
- Density func: $D(\mathbf{x}) = sphere(\mathbf{x}\,(1 + turbulence(\mathbf{x})))$
- Colormap

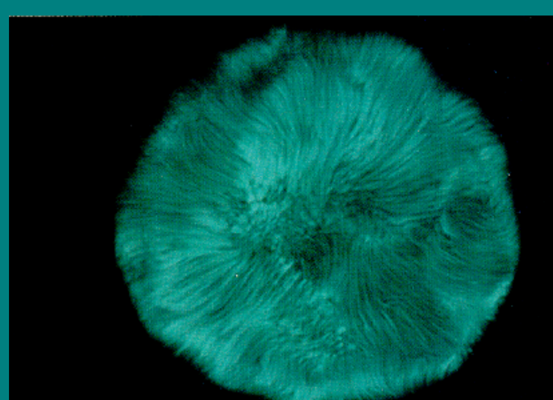


Fur

- Project points to create hairs
- Modulate density
- Control bias and gain
- Add noise in growth direction

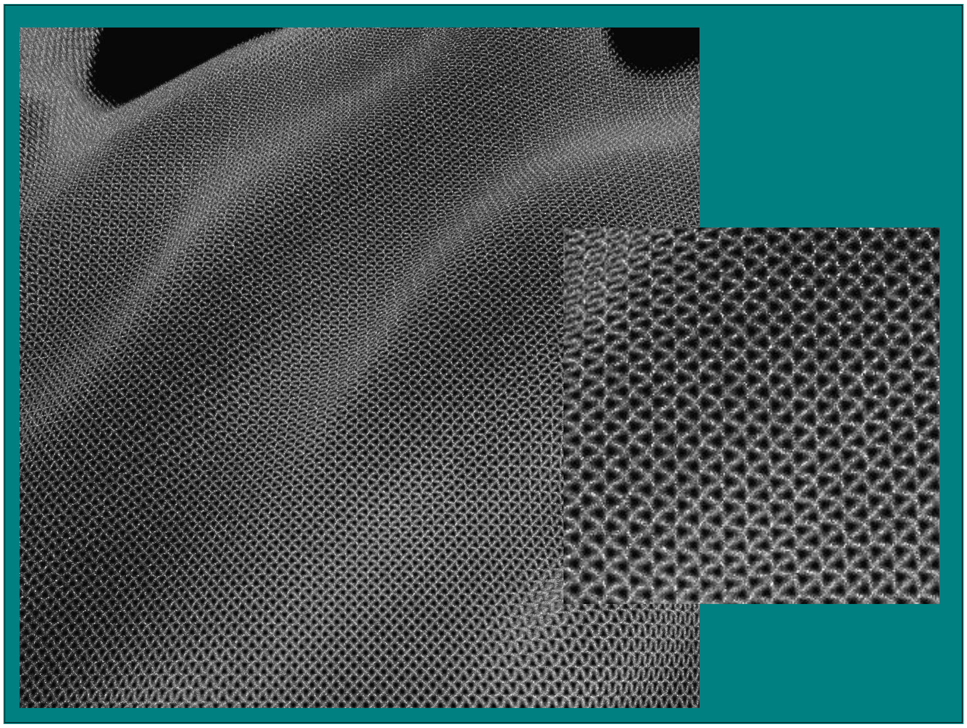


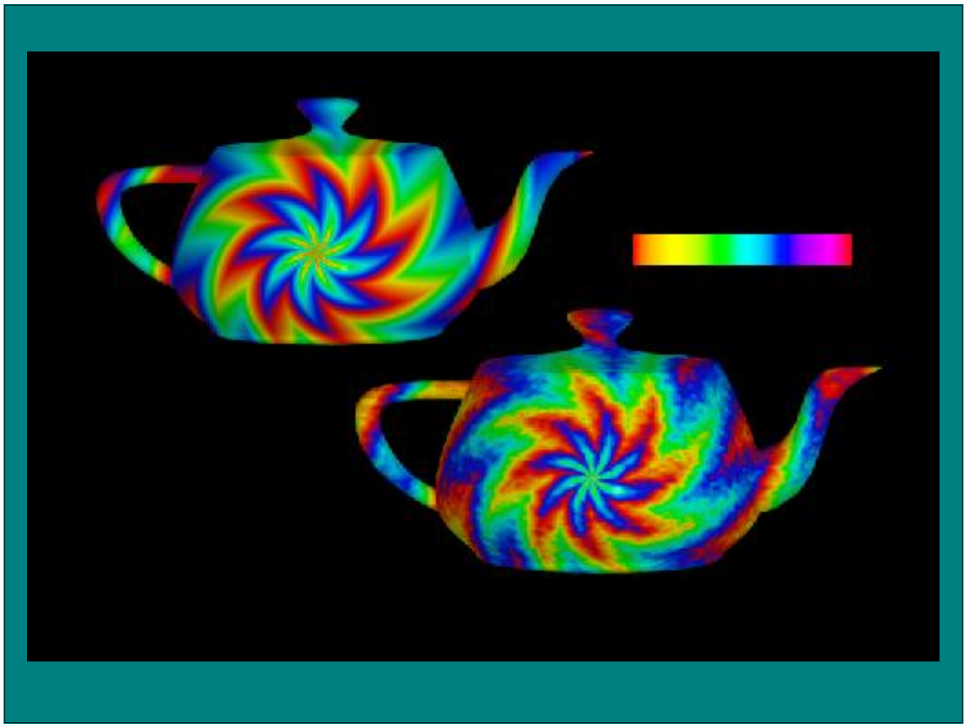


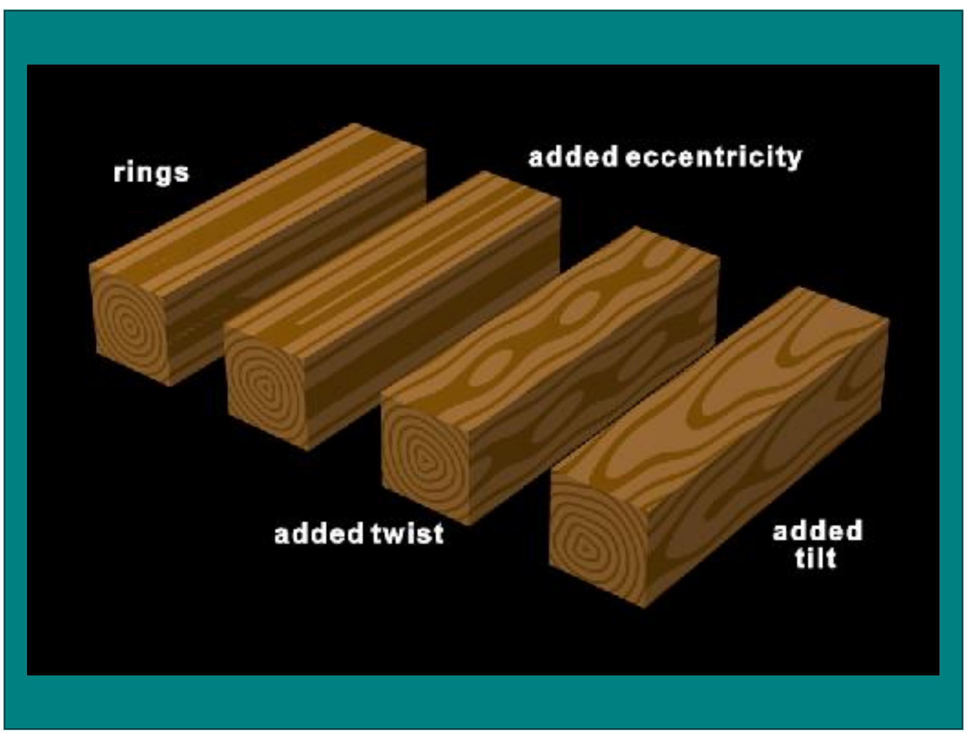


Noisy Things

- Color
- Specularity
- Opacity/Density
- Normals
- Displacements
- Index of Refraction
- Procedural Texture Parameters






rings
added eccentricity
added twist
added
tilt