Materials and Texture

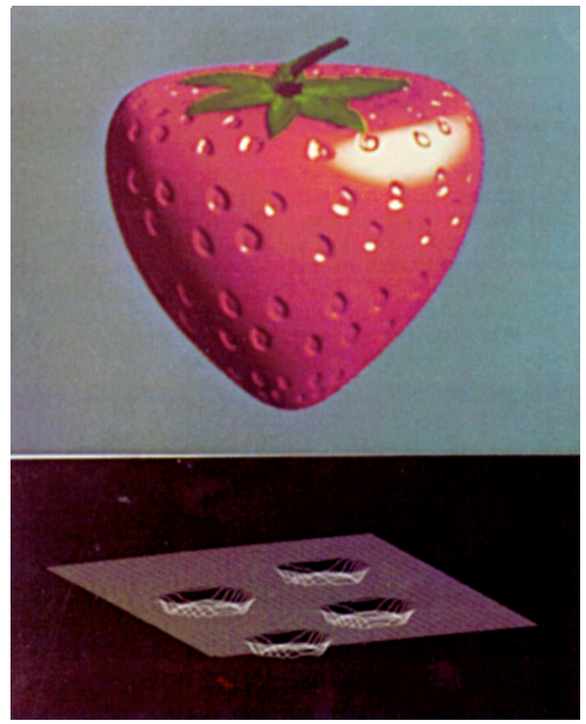
cs348b
Matt Pharr

Overview

- Spatially-varying BRDFs: BTFs
- Texture to drive BRDF spatial variation
- Image-based representations of light, texture, and beyond
- Procedural texturing
  - Shade trees
  - Shading languages

Bump Mapping

- Perturb surface normal to approximate geometric detail
- Can treat as a modification to BRDF
- Very efficient
- Breaks down at silhouettes
- Self shadowing tricky
  - Horizon mapping, ...



Blinn 1976

Displacement Mapping

- Actually modify the shape of the geometry
- Tessellation versus direct approaches
- Need ~pixel sized geometry
- How many triangles?
  - Scanline: ~nPixels * depthComplex.
  - Ray-tracing: scanline + ??
- System-design implications

BTFs

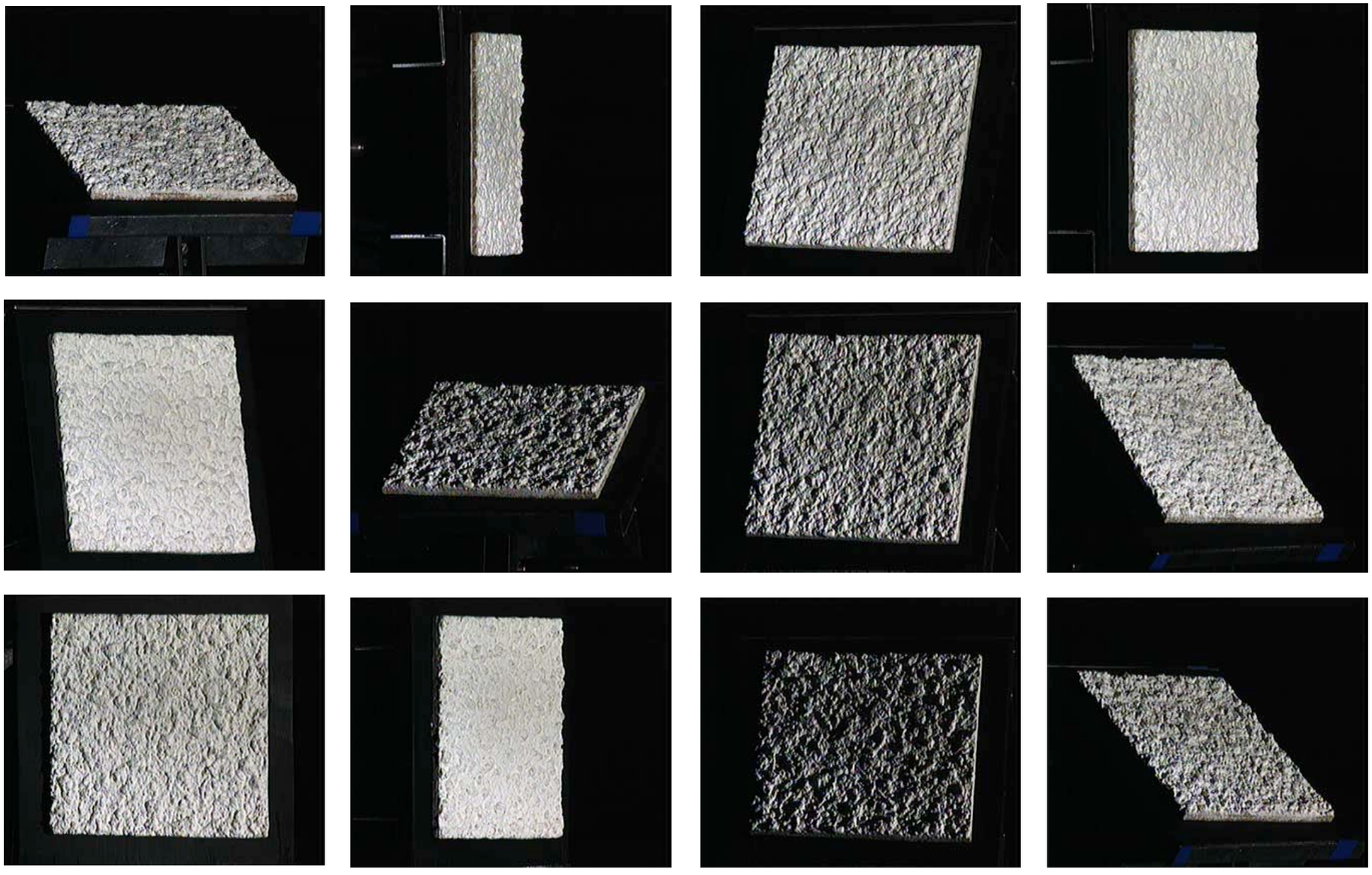
- Reflection and transmission properties vary over points on the surface
  - 4D BRDF + 2D surface position = 6D BTF

$$f_r(x, \omega_i \rightarrow \omega_o)$$

- Or

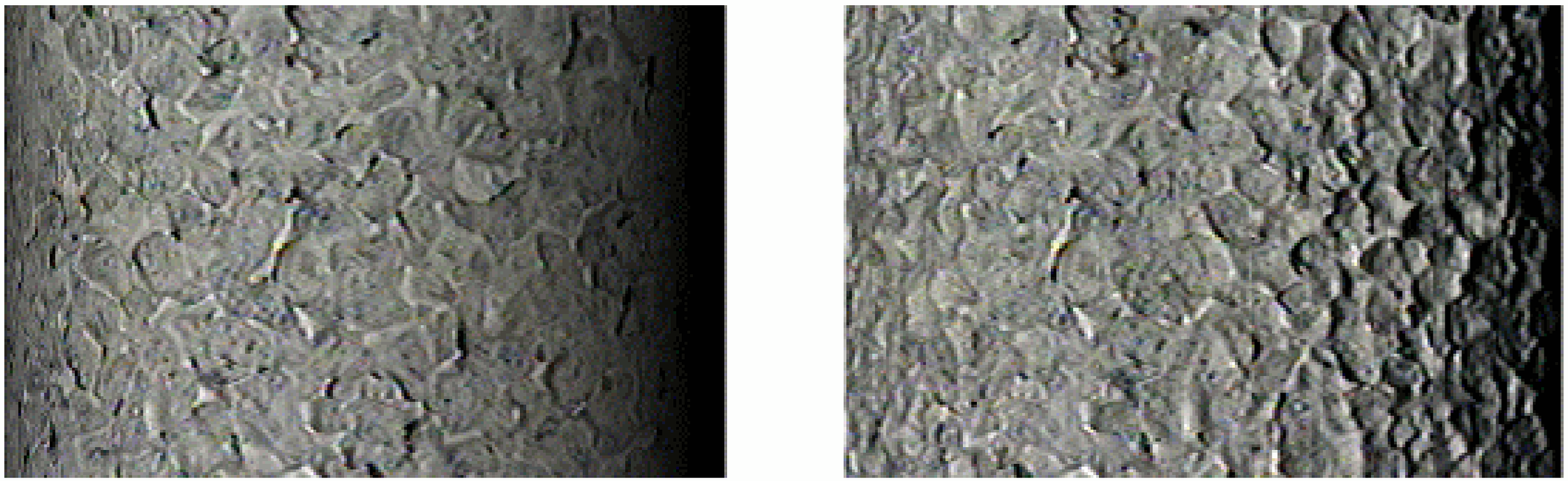
$$M : x \rightarrow f_r(\omega_i \rightarrow \omega_o)$$

BTF Mapping



Plaster (Dana et al ‘99)

BTF Mapping



BTF Data: http://www.cs.columbia.edu/CAVE/curet/

Texture

- Think of as any of a family of functions that maps from some domain to some range
- Uses for Texture
  - Surface color & transparency
  - Illumination: radiance and irradiance
  - Reflection: pre-integrated reflection
  - Shadow information
  - Displacement and bump offsets
- *Texture gives parameter values for parameterized BRDF models*

Characterizing Texture

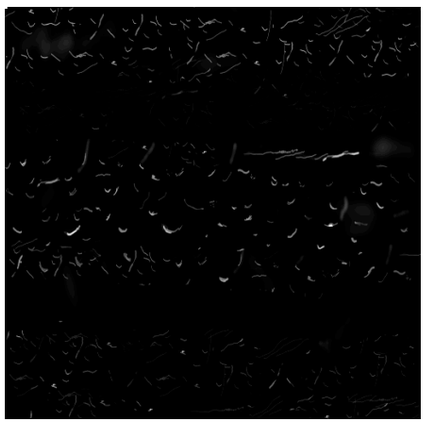
- What is the domain?
  - Dimensionality
  - Parameterization/mapping from surface pts
- What is the range?
  - Scalar values
  - Colors
  - ...
- How is the function represented?
  - Tabularized (point-samples)
  - Procedure
  - ...

Texture Domains

- Dimensionality
  - 1D, 2D, 3D, ...
- Mappings
  - Texture coordinates (s,t)
  - Surface parameters (u,v)
  - Direction vectors (N, R, H, ...)
  - Projection: sphere, cylinder,

Tabularized/Image Textures

Color & Transparency



Tom Porter

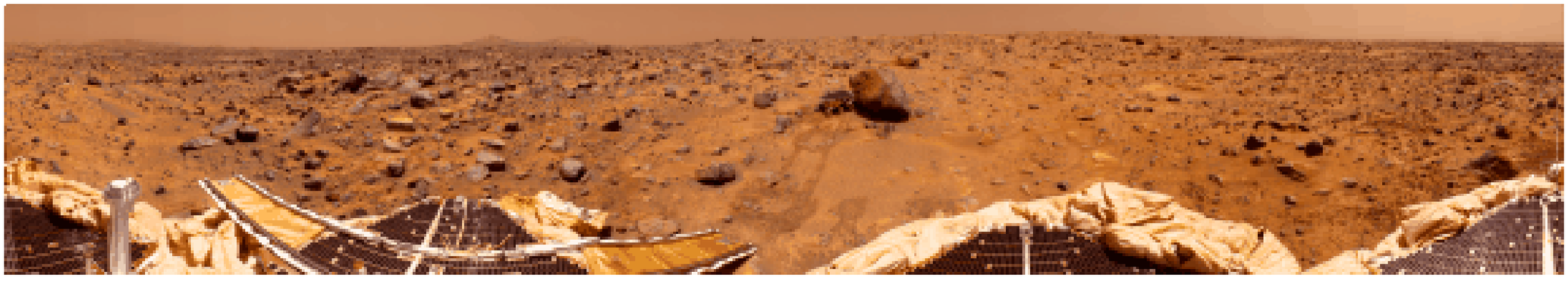
cs348b

Matt Pharr, Spring 2003

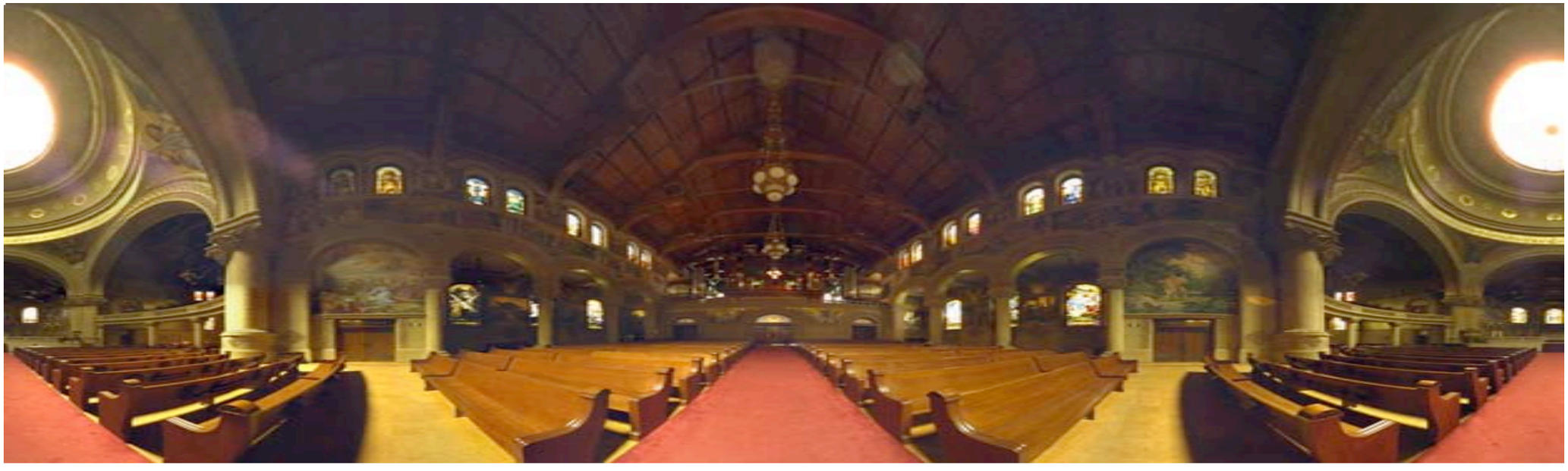
Reflection Maps

Blinn & Newell 1976

Environment Maps



Mars Pathfinder



Memorial Church (Ken Turkowski)

cs348b

Matt Pharr, Spring 2003

Fisheye Lens

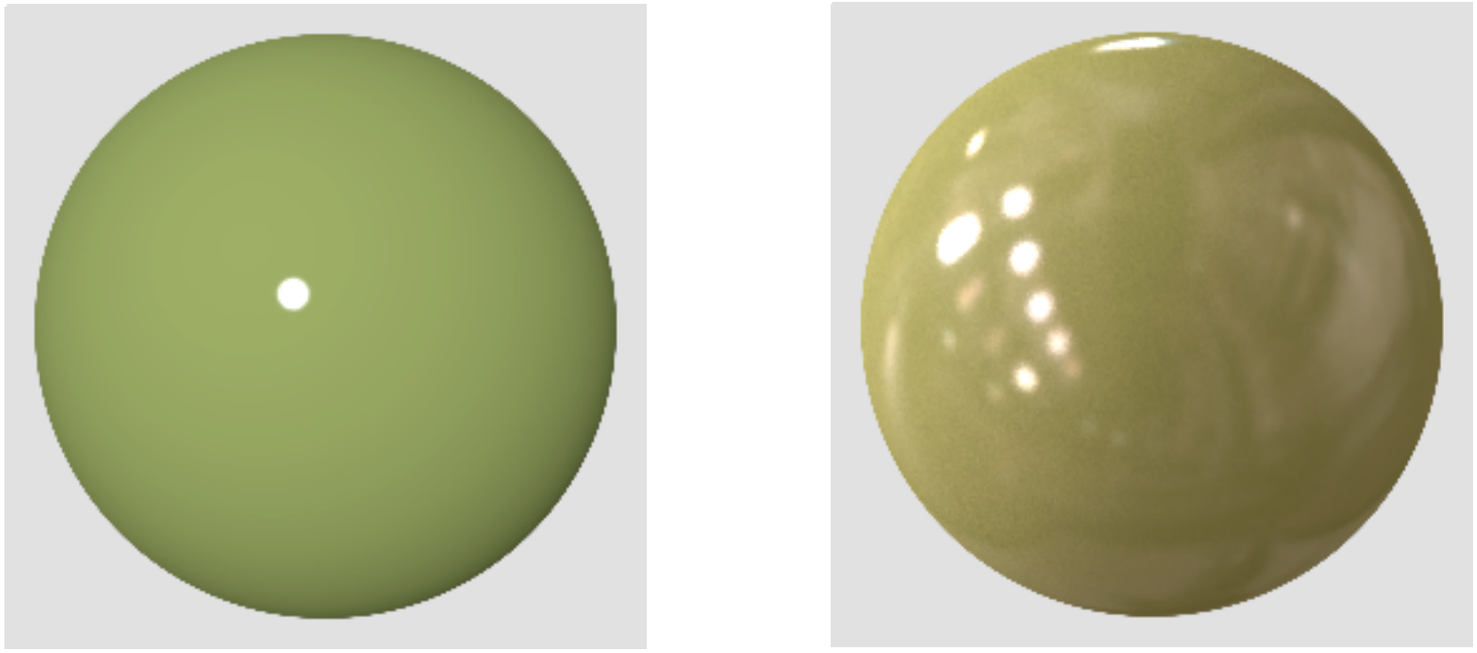
Ken Turkowski

Mapping Directions to Images

- Many choices
  - Latitude-longitude (map projections)
  - Gazing ball (N)
  - Fisheye lens
  - Cubical environment map
- Issues
  - Mapping efficiency
  - Area distortion
  - Can convert among representations with image warping

Material Recognition

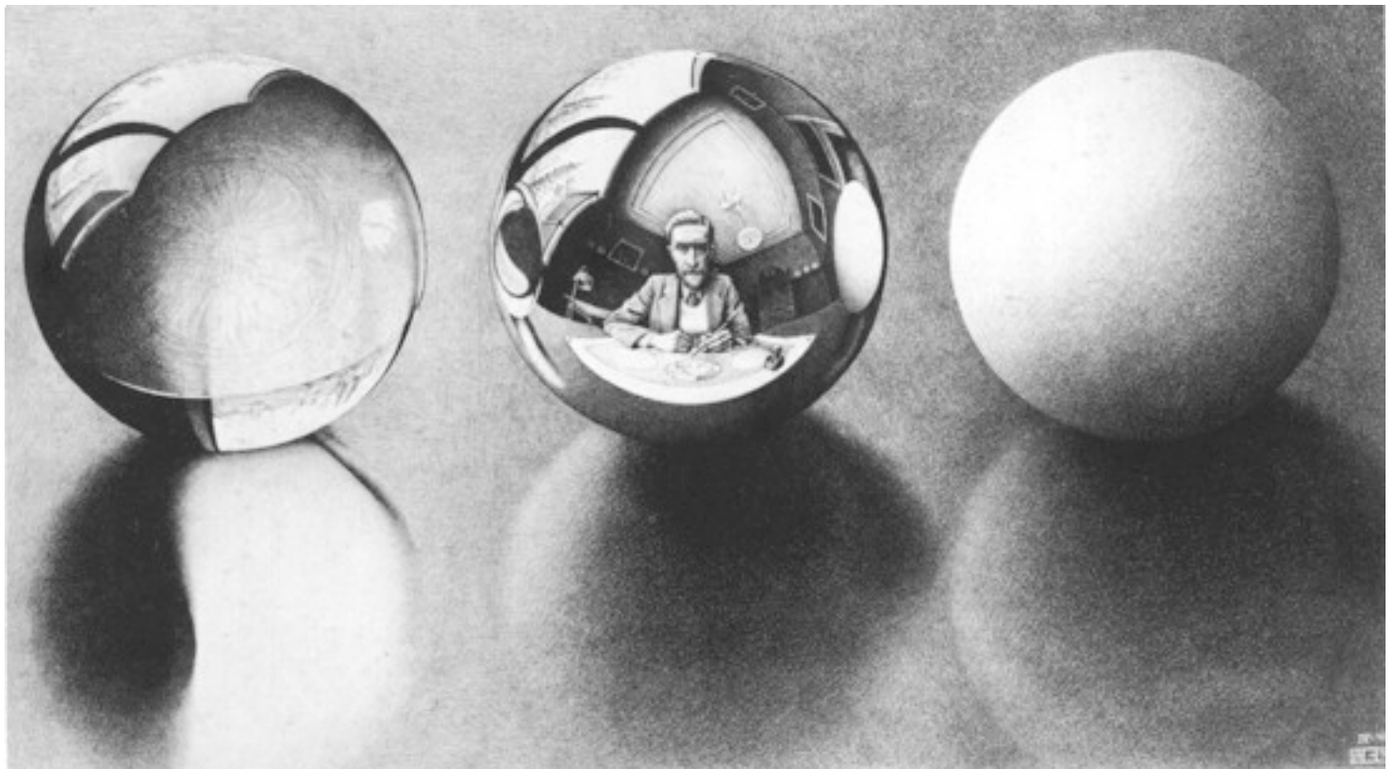
- Real illumination makes objects more recognizable than CG illumination

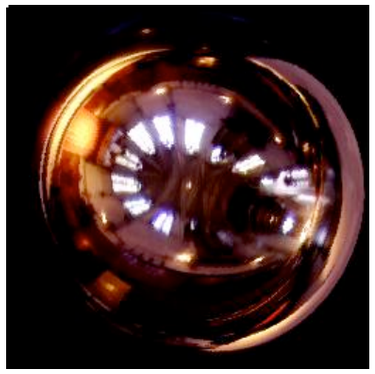


Ted Adelson

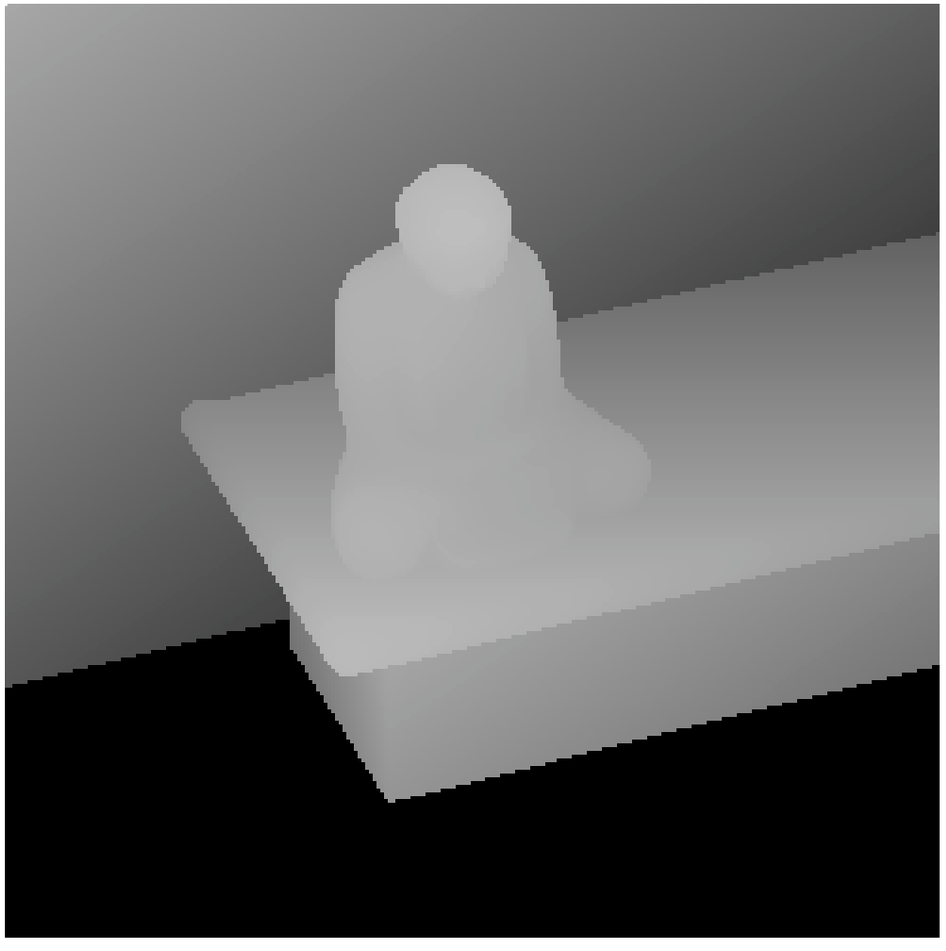
Reflectance Maps

- Compute value of reflection equation for each outgoing direction





Shadow Maps



Deep Framebuffers

- Store more than color/alpha in image:
  - Geometric info: position, surface normal, (u,v)
  - Object info: object id
  - Layers (layered depth images)
- Uses
  - Fancy image processing
  - Fast re-rendering
  - Image-based rendering (viewpoint changes)

BRDF Images

- Interactive relighting

Procedural Texturing

cs348b

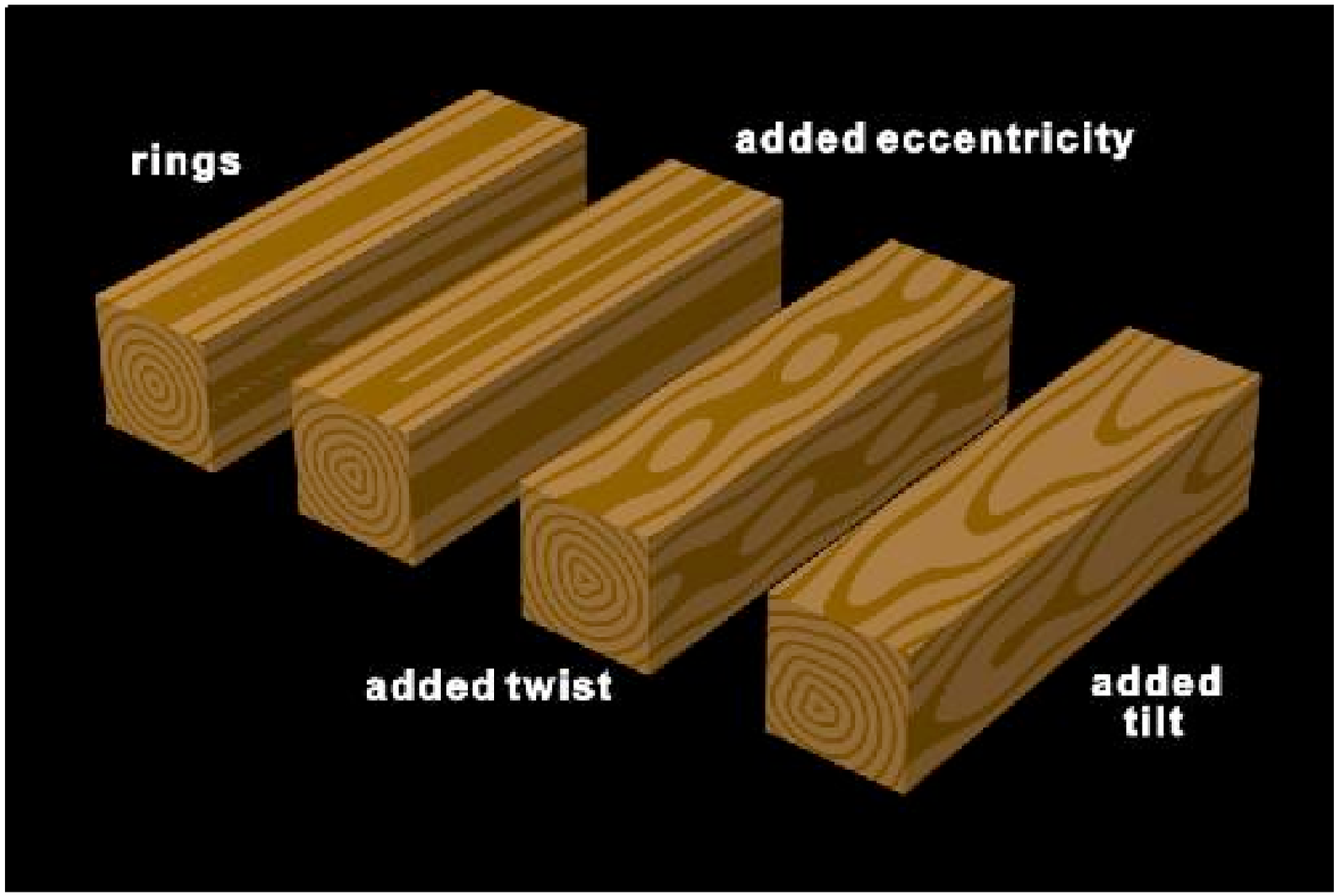
Matt Pharr, Spring 2003

Perlin's Marble Vase

cs348b

Matt Pharr, Spring 2003

Wood



Clouds

David Ebert

cs348b

Matt Pharr, Spring 2003

Approaches to Procedural Texture

- Shade Trees
  - Tree-based representation of an expression
- Shading Languages
  - "Little language" (Bentley) for shading computation

Shade Trees

- Expression tree takes set of inputs, computes single output
- Nodes: mix, min, max, texture lookup, ...
- No control flow
  - Not as big a limit as you might think
- Paradigm for many material design tools

Shading Language Constructs

- Useful data types
  - float, color, point, vector, normal, matrix, ...
- Useful "global" variables
  - P, N, Ng, u, v, dPdu, dPdv, ...
  - Automatically interpolated surface parameters
- C-like control flow, expressions, etc

Explicit vs. Implicit Pattern Description

- Explicit (e.g. Postscript)

```
0 0 1 0 360 arc closepath
0.5 setgray fill
```

- Implicit (e.g. RenderMan)

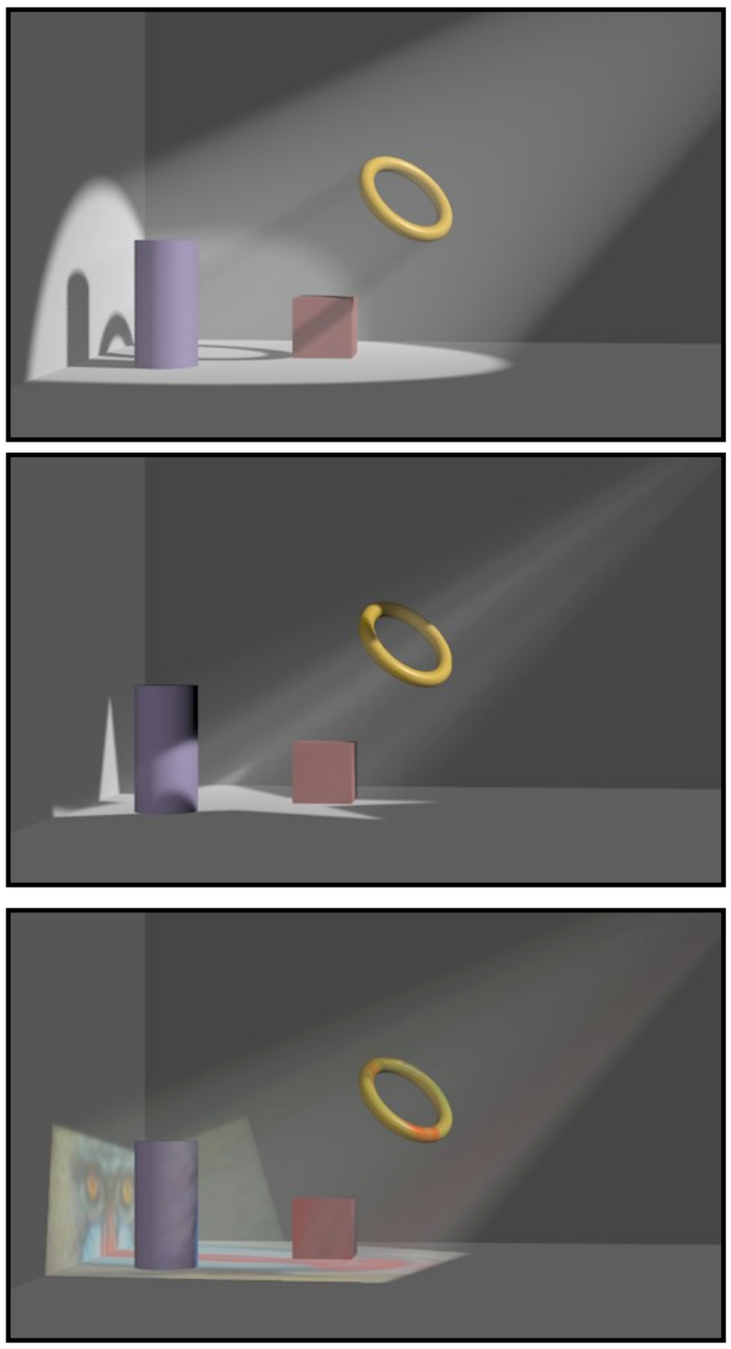
```
if (P.x*P.x + P.y*P.y < 1)
    C = color(0.5);
else
    C = color(1);
```

- Explicit easier for the user; implicit easier for the renderer. (Implicit wins.)

Derivatives

- RenderMan: SIMD model gives derivatives for free
- Area computation for anti-aliasing
  - Free with SIMD shading
  - Du(expr), Dv(expr)
  - texture(name, s, t)

Barzel's Uberlight



```
Clip to near/far planes
Clip to shape boundary
foreach superelliptical blocker
  atten *= …
foreach cookie texture
  atten *= …
foreach slide texture
  color *= …
foreach noise texture
  atten, color *= …
foreach shadow map
  atten, color *= …
Calculate intensity fall-off
Calculate beam distribution
```

The Role of A Shader

- Compute color (RenderMan)
- Compute BSDF (toro)
- Irt Material
  - In: Differential Geometry
  - Out: BSDF