Texture

Procedural shading and texturing

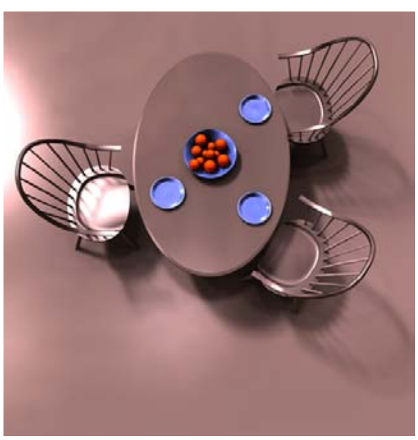
Applied and projected textures

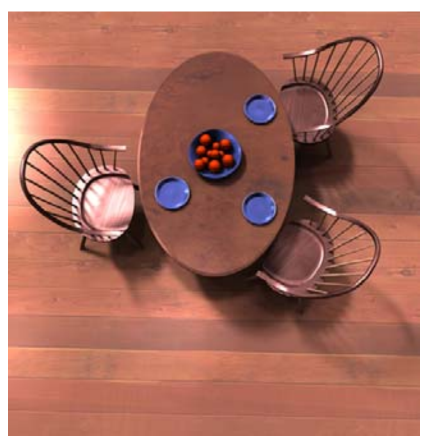
- Material / light properties
- Shadow maps

Spherical and higher order textures

- Spherical mappings
- Environment and irradiance maps
- Reflectance maps

Detail Representation





Texture Maps

What does the texture represent?

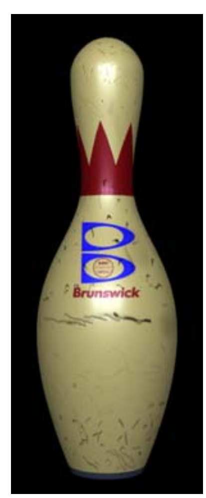
- Dimensionality: 1D, 2D, 3D, …
- Surface color and opacity
- Illumination functions: environment maps, shadow maps
- Geometry: bump and displacement maps
- Reflection functions: reflectance maps

How is it mapped onto surfaces?

- Decal: surface parameterization (u,v)
- Direction vectors: reflection R, normal N, halfway H
- Projection: cylinder, slide-projector

Surface Color and Transparency

Tom Porter's Bowling Pin

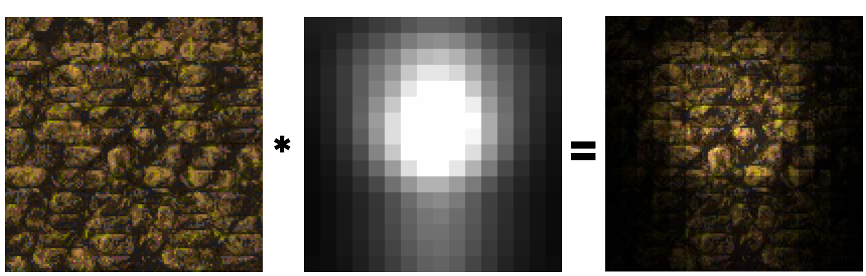


Source: RenderMan Companion, Pls. 12 & 13

Procedural Surface Shading

```
surface shader float4 bowling pin (texref base, texref bruns,
  texref circle, texref coated, texref marks, float4 uv)
{
  ... // Omitted texture coordinate generation code
  float4 Base = texture(base, m_base * uv_wrap);
  float4 Bruns = front * texture(bruns, m_bruns * uv_label);
  float4 Circle = front * texture(circle, m_circle * uv_label);
  float4 Coated = (1 - front) * texture(coated, m_coated * uv_label);
  float4 Marks = texture(marks, m_marks * uv_wrap);
  // Invoke lighting models from lightmodels.h
  float4 Cd = lightmodel diffuse({0.4, 0.4, 0.4, 1}, {0.5, 0.5, 0.5, 1});
  float4 Cs = lightmodel specular({0.35, 0.35, 0.35, 1}, {0, 0, 0, 0}, 20);
  // Composite textures, apply lighting, and return final color
  return (Circle over (Bruns over (Coated over Base))) * Marks * Cd + Cs;
}
```

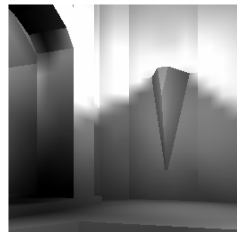
Illumination Maps



Reflectance	Irradiance	Radiosity
$\rho(x)$	$E(x)$	$B(x)$

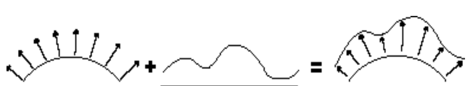
Quake Light Maps

*



Lower resolution

Displacement/Bump Mapping



$$\mathbf{P}(u,v)$$

$$\mathbf{S}(u,v)=\frac{\partial \mathbf{P}(u,v)}{\partial u} \qquad \mathbf{T}(u,v)=\frac{\partial \mathbf{P}(u,v)}{\partial v}$$

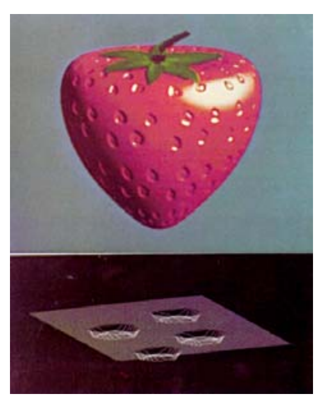
$$\mathbf{N}(u,v)=\mathbf{S}\times\mathbf{T}$$

- **Displacement**

$$\mathbf{P}'(u,v)=\mathbf{P}(u,v)+h(u,v)\mathbf{N}(u,v)$$

- **Perturbed normal**

$$\mathbf{N}'(u,v)=\mathbf{P}'_u\times\mathbf{P}'_v$$
$$=\mathbf{N}+h_u(\mathbf{T}\times\mathbf{N})+h_v(\mathbf{S}\times\mathbf{N})$$



From Blinn 1976

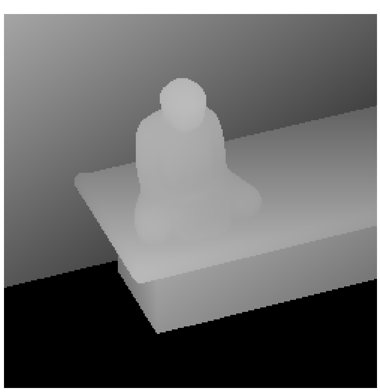
Procedural Light Shading

Barzel's *UberLight.sl*

```
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
}
```


Shadow Maps

Shadow maps = depth maps from light source



Correct Shadow Maps

Step 1:

Create z-buffer of scene as seen from light source

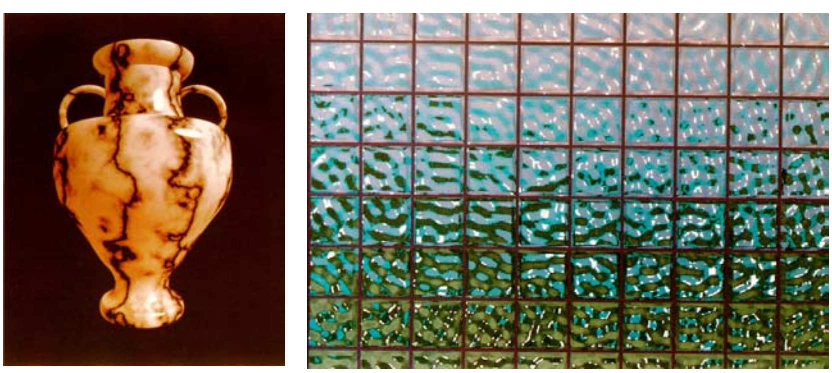
Step 2.

Render scene as seen from the eye

- For each light
  - Transform point into light coordinates
    - return (zl < zbuffer[xl][yl]) ? 1 : 0

CS348B Lecture 12 Ren Ng, Spring 2004

Image Synthesis with Noise



Perlin 1985

CS348B Lecture 12 Ren Ng, Spring 2004

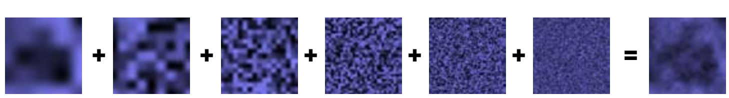
Perlin's Noise Function

1. Create a table of random 3D gradients
2. Hash a 3D lattice to a table entry
3. Perform cubic (or higher order) interpolation

Perlin 1985, 2002

Turbulence

```
// Compute fractal noise
for (i = 0; i < 6; i++) {
  turb += 1/freq * noise(freq*P);
  freq *= 2;
}
```



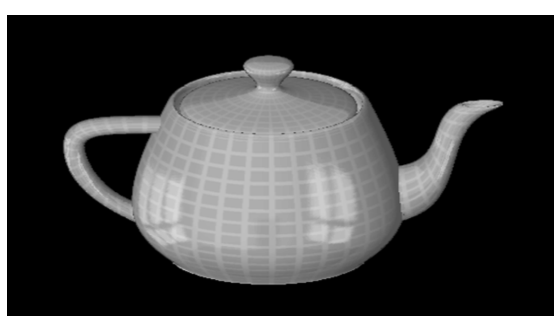
Images from http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

History

Catmull/Williams 1974 - basic idea
Blinn and Newell 1976 - basic idea, reflection maps
Blinn 1978 - bump mapping
Williams 1978, Reeves *et al.* 1987 - shadow maps
Smith 1980, Heckbert 1983 - texture mapped polygons
Williams 1983 - mipmaps
Miller and Hoffman 1984 - illumination and reflectance
Perlin 1985, Peachey 1985 - solid textures
Greene 1986 - environment maps/world projections
Akeley 1993 - Reality Engine

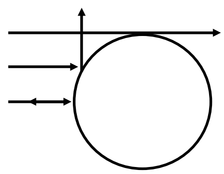
Reflection Maps

Blinn and Newell, 1976



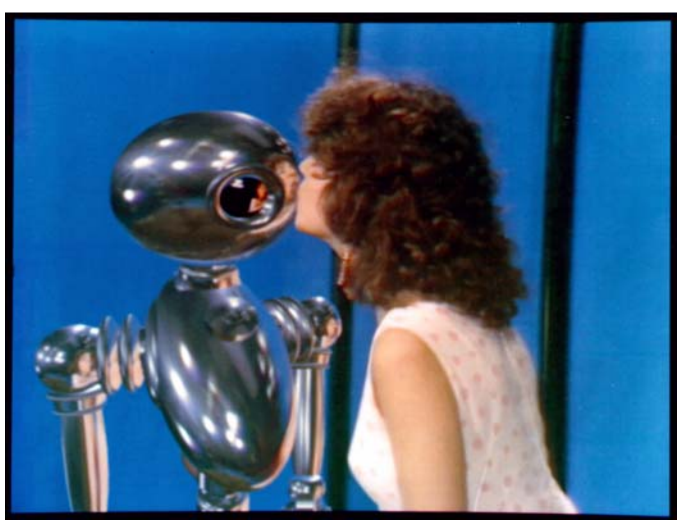
Gazing Ball

Miller and Hoffman, 1984



- **Photograph of mirror ball**
- **Maps all directions to a to circle**
- **Resolution function of orientation**
- ***Reflection indexed by normal***

Environment Maps



Interface, Chou and Williams (ca. 1985)

Environment Map Approximation

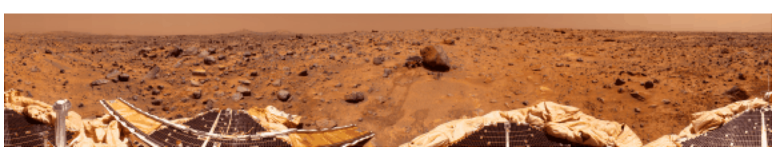
Ray Traced

Environment Map

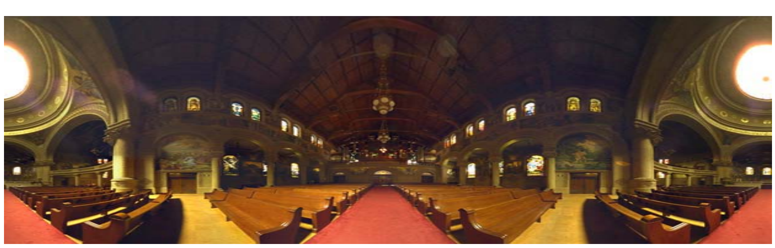
Self reflections are missing in the environment map

Cylindrical Panoramas

QuickTime VR



Mars Pathfinder

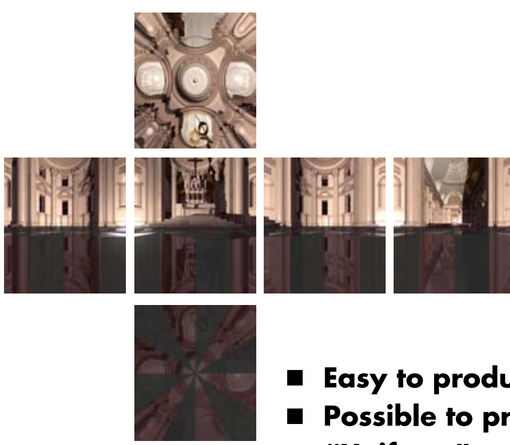


Memorial Church (Ken Turkowski)

Fisheye Lens

Pair of 180 degree fisheye
Photo by K. Turkowski

Cubical Environment Map



- Easy to produce with rendering system
- Possible to produce from photographs
- "Uniform" resolution
- Simple texture coordinates calculation

Direction Maps

Many ways to map directions to images…
Methods:

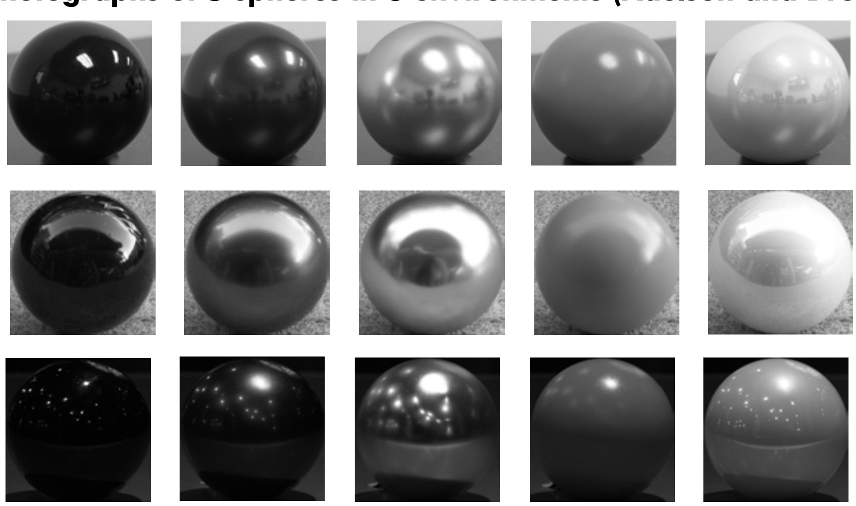
- Lattitude-Longitude (Map Projections) [Newell and Blinn]
  - Create by painting
- Gazing Ball (N) [Miller and Hoffman]
  - Create by photographing a reflective sphere
- Fisheye Lens
  - Standard camera lens
- Cubical Environment Map (R)
  - Create with a rendering program, photography…

Issues:

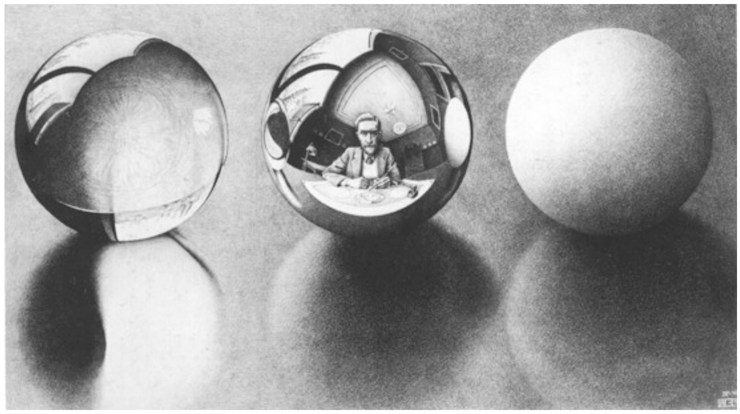
- Non-linear mapping - expensive, curved lines
- Area distortion - spatially varying resolution
- Convert between maps using image warp

Combining Reflectance & Illumination

Photographs of 5 spheres in 3 environments (Adelson and Dror)



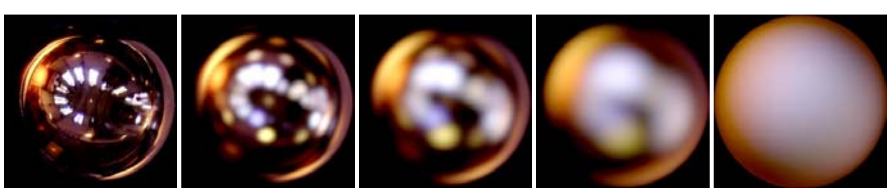
Reflectance Maps



For a given viewing direction
- For each normal direction
  - For each incoming direction (hemispherical integral)
    - Evaluate reflection equation

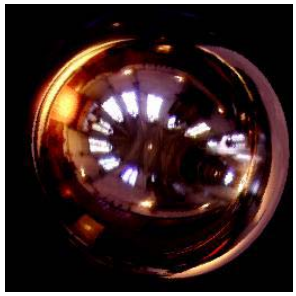
Example: Phong Model

Rough surfaces blur highlight



σ

Example: Lambertian Reflectance



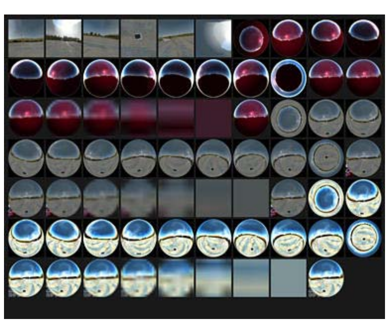
Incident Lighting

Reflected Light

$$B(\hat{\mathbf{N}}) = \rho E(\hat{\mathbf{N}})$$

Radiosity or Irradiance Map

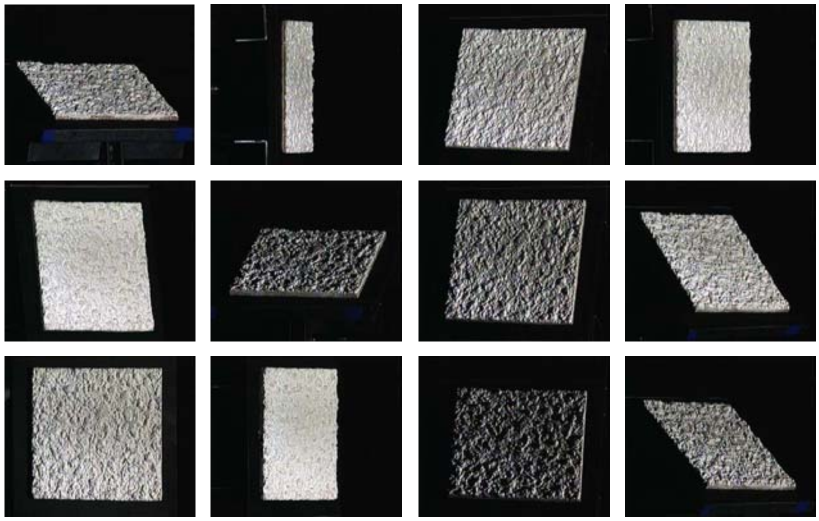
Reflectance Space Shading



12 directions

Cabral, Olano, Nemic 1999

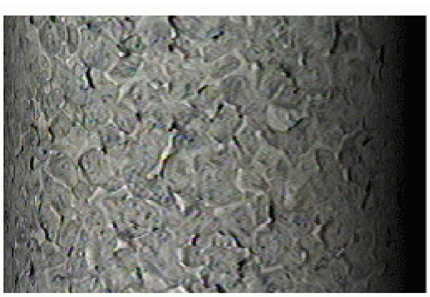
Bidirectional Texture Function (BTF)

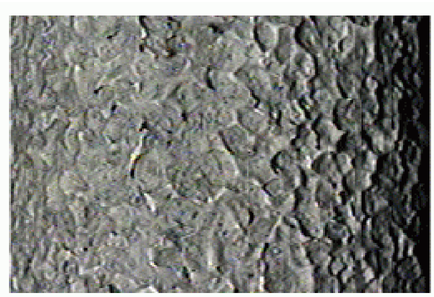


Plaster

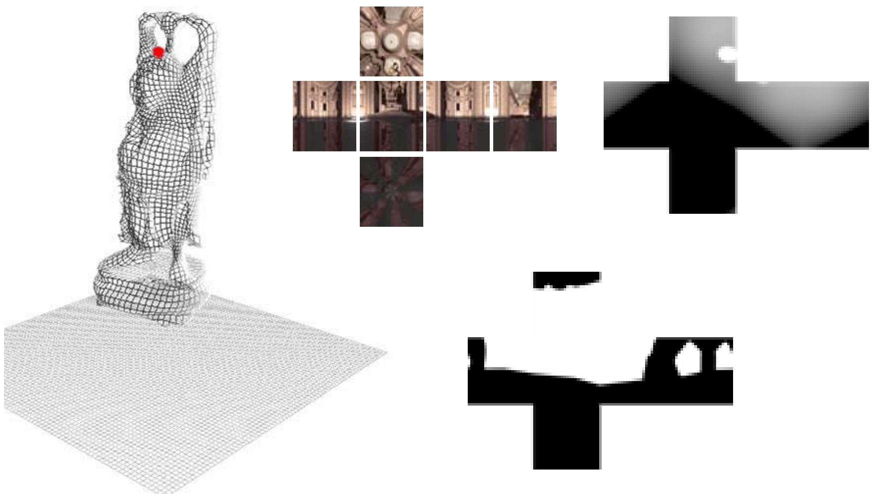
BTF Mapping

Complex interplay between texture and reflection





Relighting with Visibility Map Textures



CS348B Lecture 12 Ren Ng, Spring 2004