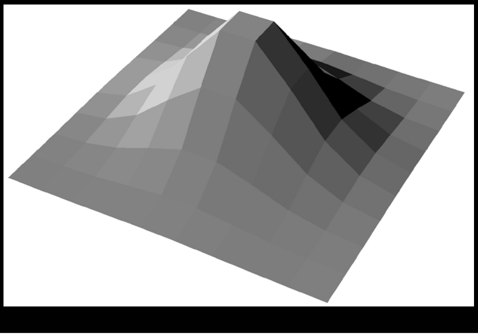
Texture and other Mappings

Texture Mapping
Bump Mapping
Displacement Mapping
Environment Mapping

Angel Chapter 7

Last Time - Shading

- Flat shading: Inexpensive to compute
- Appropriate for objects with flat faces
- Less pleasant for smooth surfaces



Flat Shading and Perception

- *Lateral inhibition*: exaggerates perceived intensity
- *Mach bands*: perceived “stripes” along edges

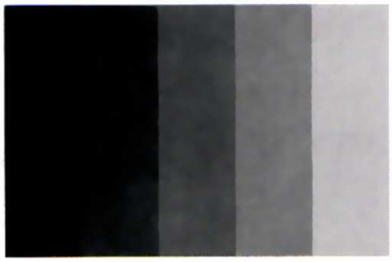


Figure 6.28 Step chart.

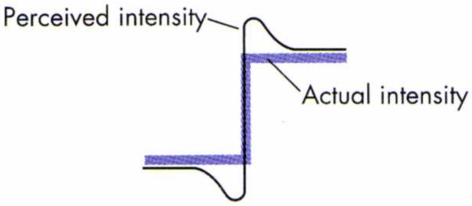
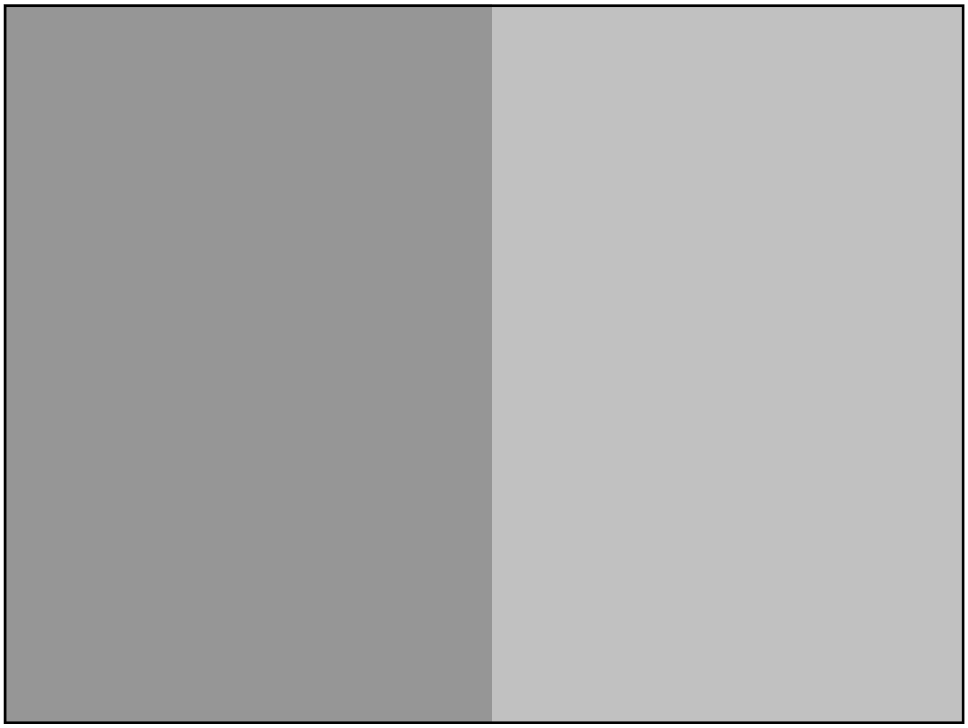
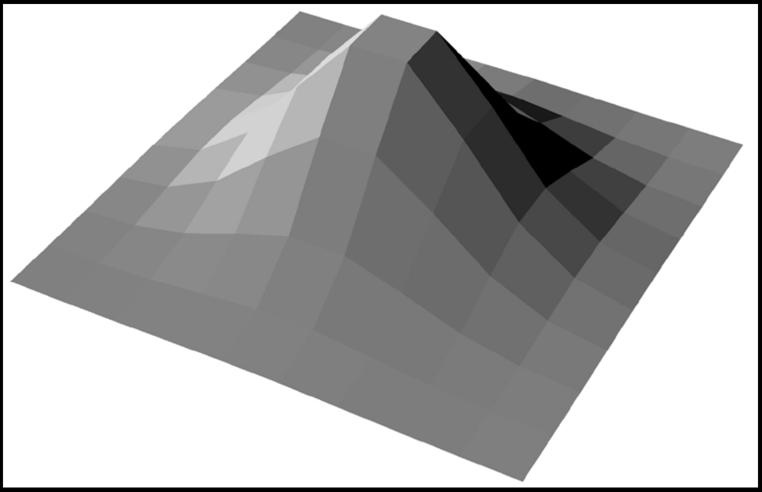


Figure 6.29 Perceived and actual intensities at an edge.



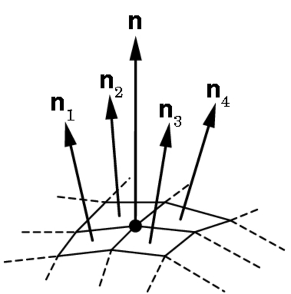


Gouraud Shading

- Interpolative shading
- Compute vertex normal by averaging all adjacent face normals

$$\mathbf{n} = \frac{\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4}{|\mathbf{n}_1 + \mathbf{n}_2 + \mathbf{n}_3 + \mathbf{n}_4|}$$

- Requires knowledge about which faces share a vertex—adjacency info



Defining and Maintaining Normals

- Define unit normal before each vertex

```
glNormal3f(nx, ny, nz);
glVertex3f(x, y, z);
```

- Length changes under some transformations
- Ask OpenGL to re-normalize (always works)

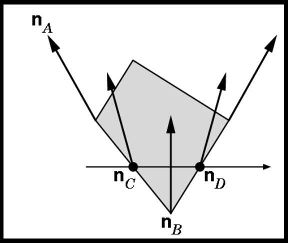
```
glEnable(GL_NORMALIZE);
```

- Ask OpenGL to re-scale normal (works for **uniform** scaling, rotate, translate)

```
glEnable(GL_RESCALE_NORMAL);
```

Phong Shading

- Interpolate normals rather than colors
- Significantly more expensive
- Mostly done off-line (not supported in OpenGL)

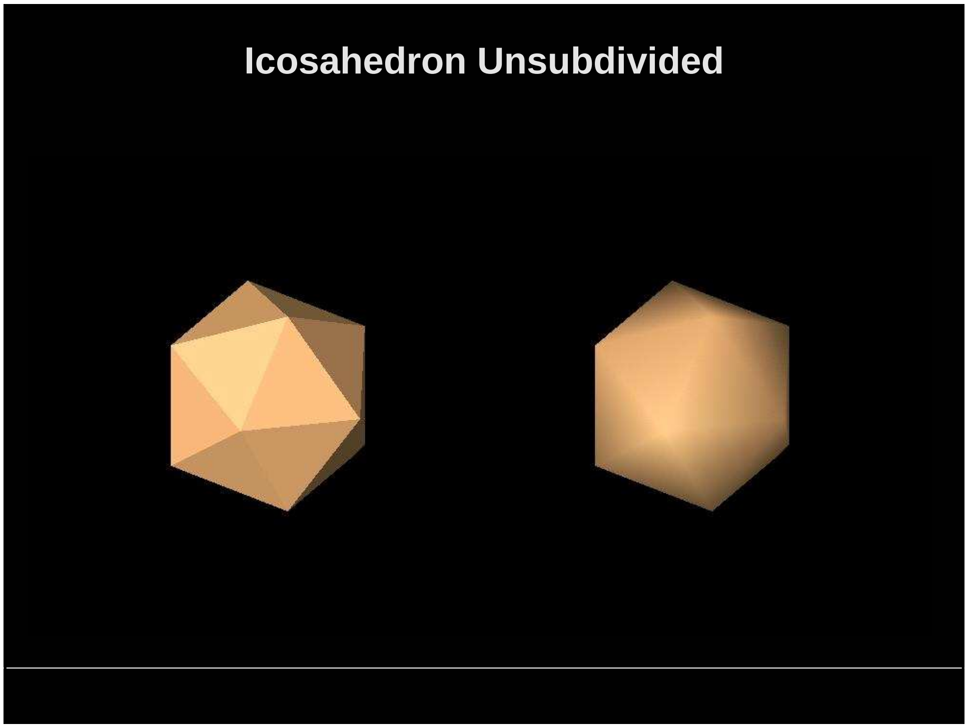


Phong Shading Results

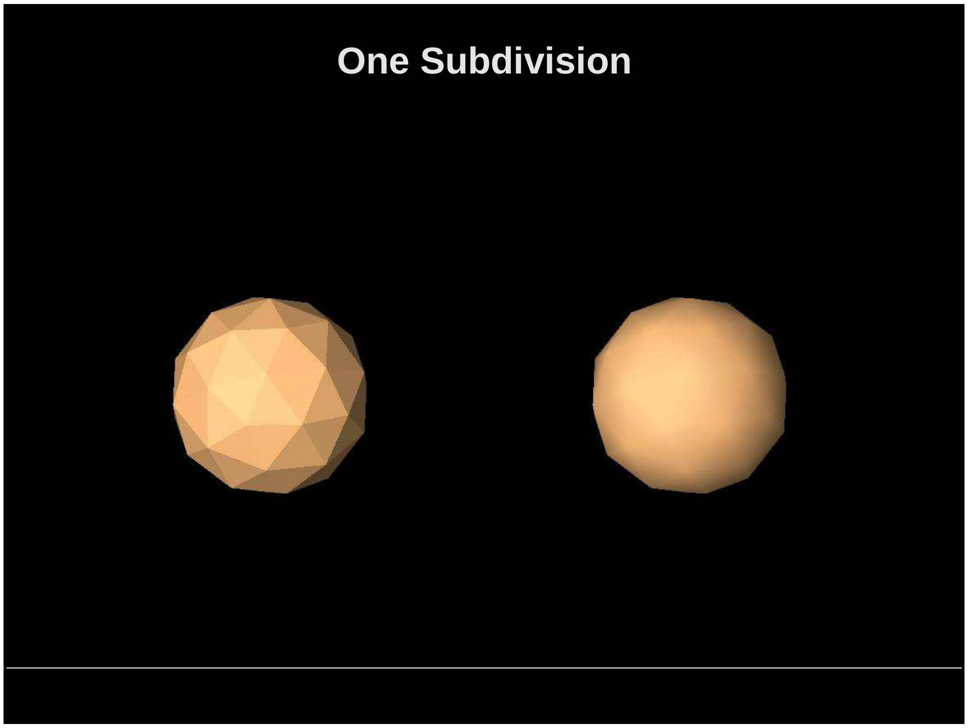
Michael Gold, Nvidia

Phong Lighting
Gouraud Shading

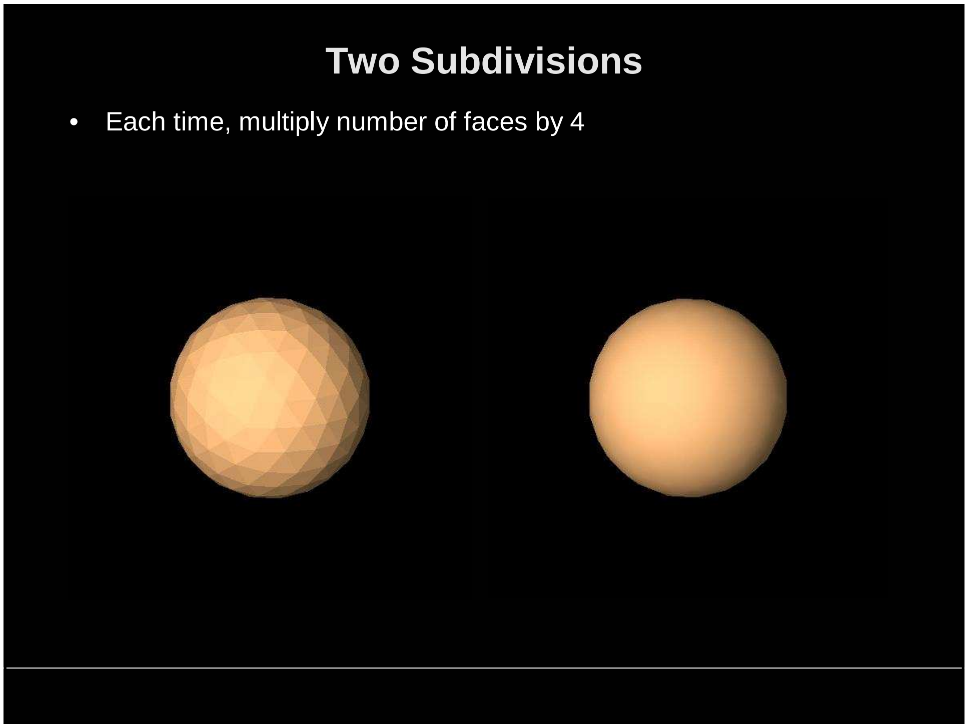
Phong Lighting,
Phong Shading



Icosahedron Unsubdivided



One Subdivision

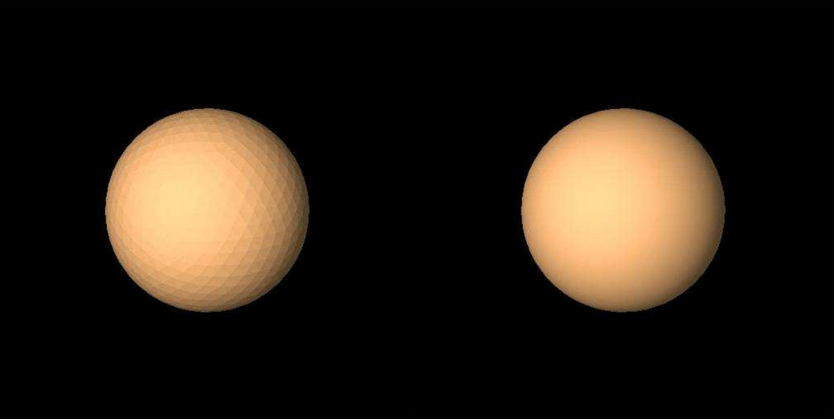


Two Subdivisions

- Each time, multiply number of faces by 4

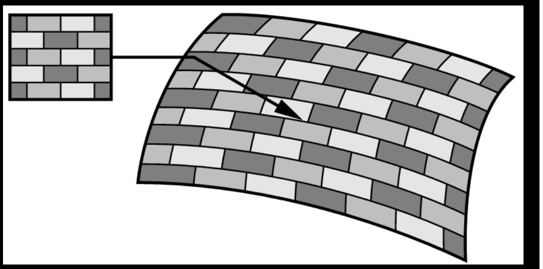
Three Subdivisions

- Reasonable approximation to sphere

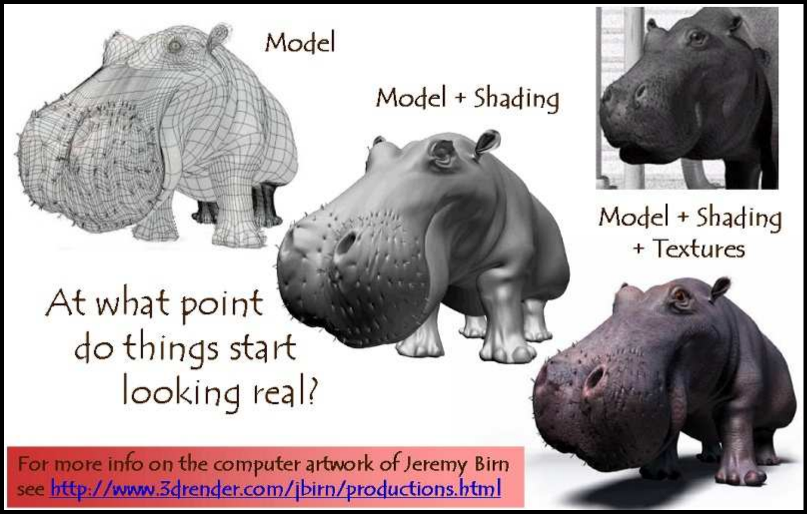


Textures

- Real objects have small surface features
- One option: use a huge number of polygons with appropriate surface coloring and reflectance characteristics
- Another option: use a mapping algorithm to modify the shading algorithm
  - Texture mapping
  - Bump mapping
  - Displacement mapping
  - Environmental mapping



The Quest for Visual Realism



2D Texture Mapping

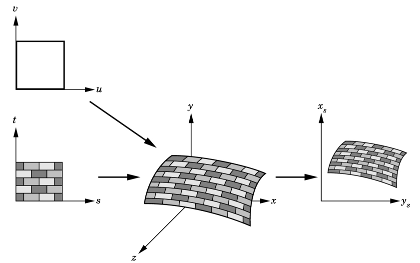
Texture images to make our surfaces more life-like

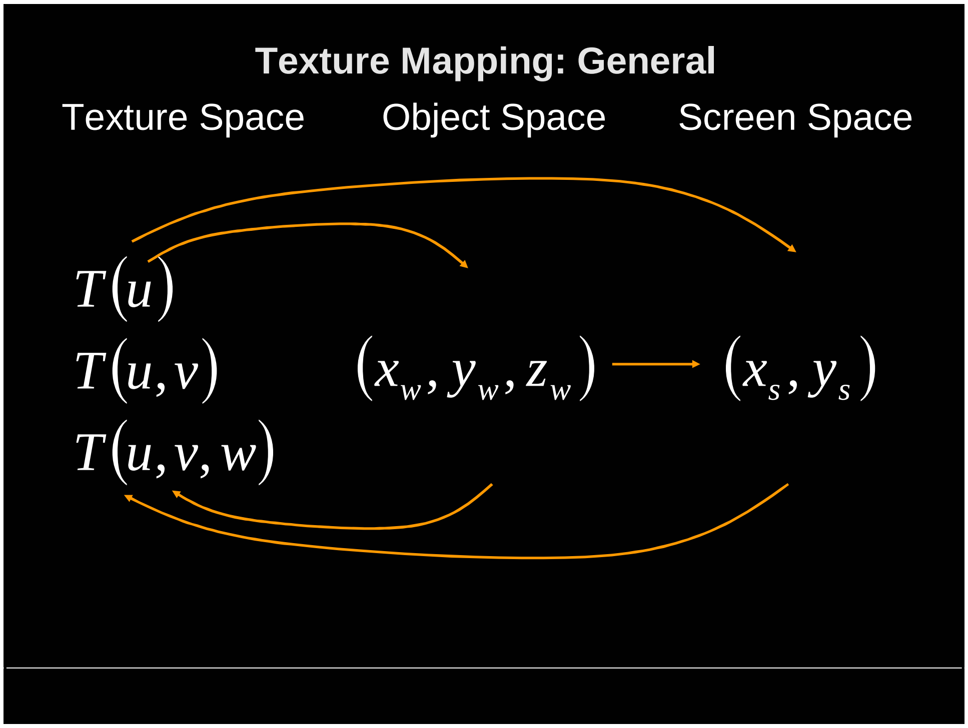
- Scan textures from the world (clouds, wood grain) or paint them yourself
- Store the texture in a 2D image
- Map the image onto the surface by a function which maps *(u,v)* coordinates of our parametric surface onto *(x,y)* image coordinates
- When shading a surface point, we look up the appropriate pixel from the 2D image, and use that to affect the final color

Voila! Your favorite picture painted onto a donut.

This technique is called *parametric texture mapping*

But how to map from texture coordinates to object coordinates? Easy for a parametric surface, less obvious for other models.





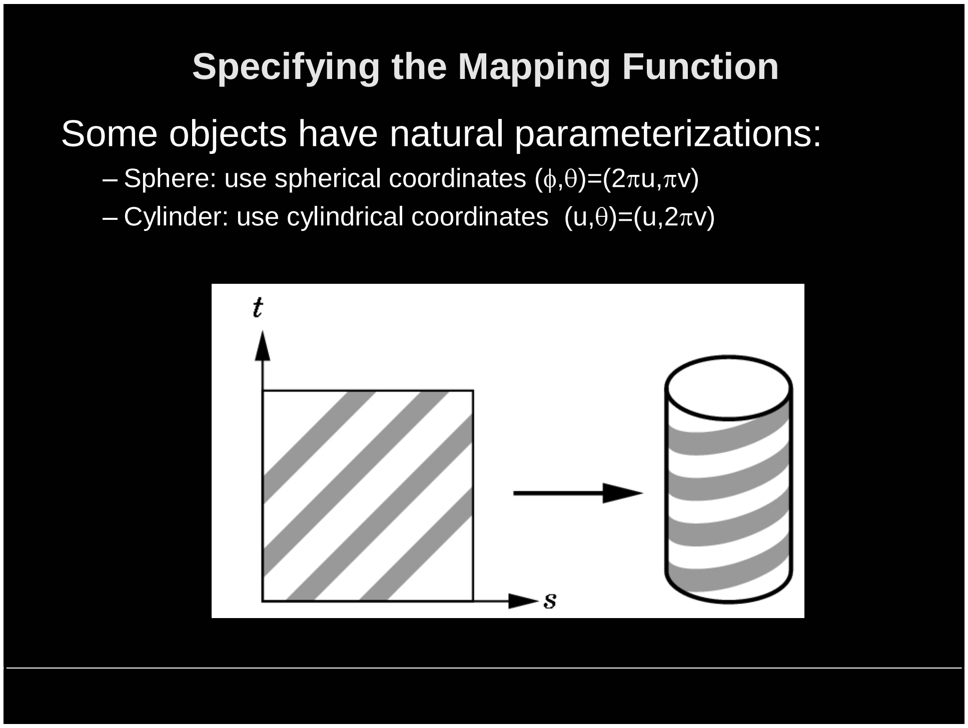
Texture Mapping: General

Texture Space Object Space Screen Space

$T(u)$

$T(u,v)$ $(x_w, y_w, z_w) \longrightarrow (x_s, y_s)$

$T(u,v,w)$



Specifying the Mapping Function

Some objects have natural parameterizations:

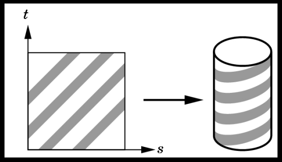
- Sphere: use spherical coordinates $(\phi,\theta)=(2\pi u,\pi v)$
- Cylinder: use cylindrical coordinates $(u,\theta)=(u,2\pi v)$

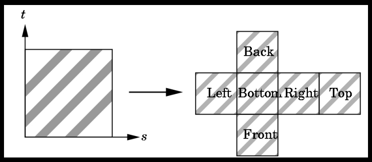
Specifying the Mapping Function

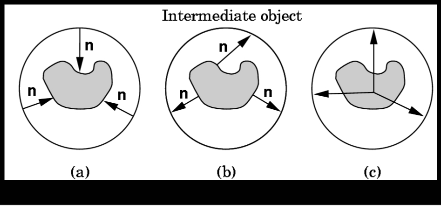
What about arbitrary polygonal objects?

Two step mapping:

- To a canonical shape first
- Then project normals from object

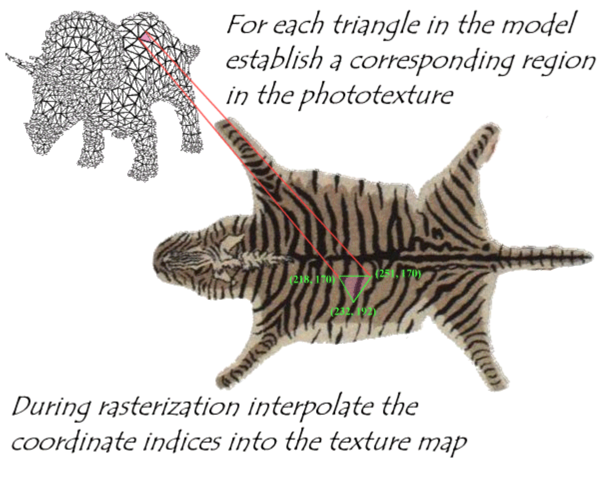






a) From texture value to object
b) Use normals to find texture
c) Line from center to point to texture value

Or design the mapping by hand

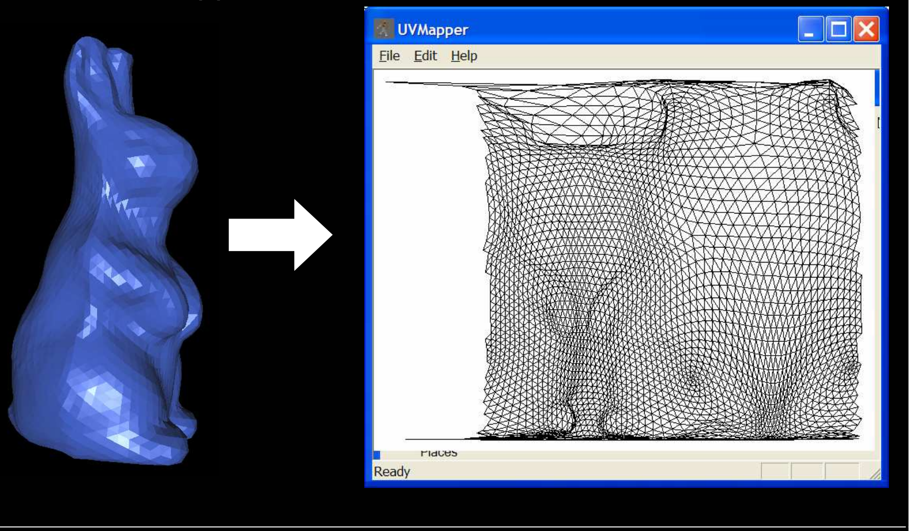


For each triangle in the model establish a corresponding region in the phototexture

During rasterization interpolate the coordinate indices into the texture map

Demo: “uvMapper”

- www.uvmapper.com

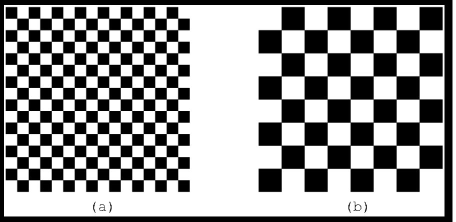


Texture Mapping in OpenGL

```
Glubyte my_texels[512][512];
glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0,
             GL_RGB,GL_UNSIGNED_BYTE, my_texels);
/* level, components, w, h, border, format, type, tarray */

glEnable(GL_TEXTURE_2D);

/* assign texture coordinates */
glBegin(GL_QUAD);
       glTexCoord2f(0.0, 0.0);
       glVertex2f(x1,y1,z1);
       glTexCoord2f(1.0, 0.0);
       glVertex2f(x2,y2,z2);
       glTexCoord2f(1.0,1.0);
       glVertex2f(x3,y3,z3);
       glTexCoord2f(0.0,1.0);
       glVertex2f(x4,y4,z4);
glEnd();
```



Grungy details we’ve ignored

- Specify s or t out of range? Use `GL_TEXTURE_WRAP` in `glTexParameter` because many textures are carefully designed to repeat

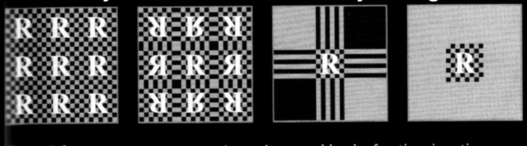
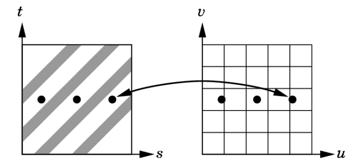


Figure 5.6. Image texture repeat, mirror, clamp, and border functions in action.

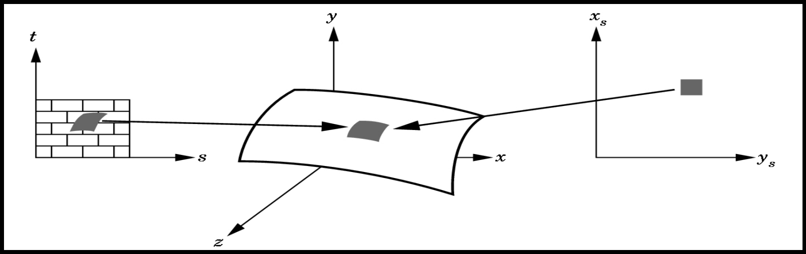
- Aliasing? Mapping doesn’t send you to the center of a texel. Can average nearest 2x2 texels using `GL_LINEAR`



- Mipmapping: use textures of varying resolutions. 64x64 becomes 32x32,16x16,8x8,4x4,2x2 and 1x1 arrays with `gluBuild2Dmipmaps`

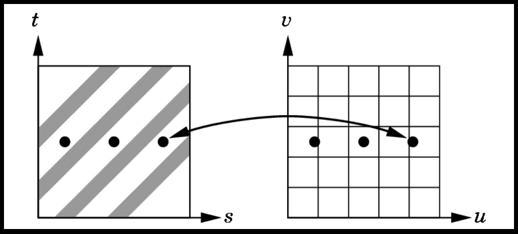
What is aliasing?

- Sampling error when mapping texture images to screen



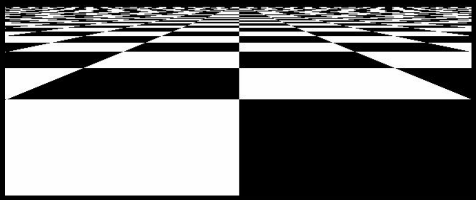
What is aliasing?

- An on-screen pixel may not map neatly to a texel.



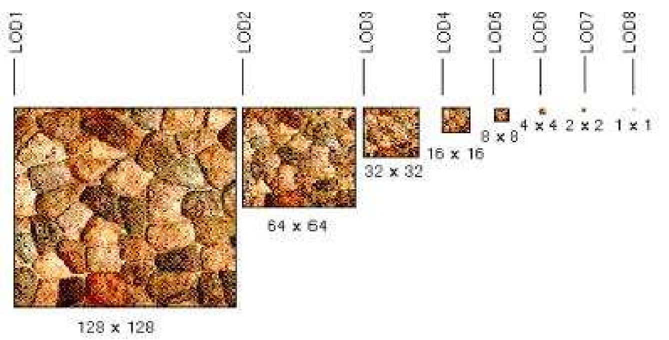
Example: Checkerboard

- Particularly severe problems in regular textures



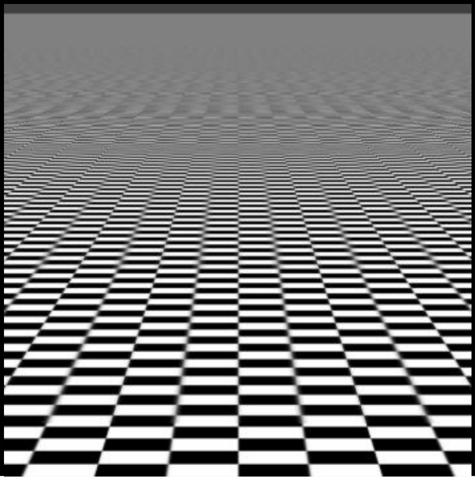
The Beginnings of a Solution: Mipmapping

- Pre-calculate how the texture should look at various distances, then use the appropriate texture at each distance. This is called *mipmapping*.
- “Mip” → “multum in parvo” or “many things in a small place”
- Each mipmap (each image below) represents a level of resolution.
- Powers of 2 make things much easier.



The Beginnings of a Solution

- Problem: Clear divisions between different depth levels
- Mipmapping alone is unsatisfactory.



Another Component: Filtering

- Take the average of multiple texels to obtain the final RGB value
- Typically used along with mipmapping
- *Bilinear filtering*
  - Average the four surrounding texels
  - Cheap, and eliminates some aliasing, but does not help with visible LOD divisions

Another Component: Filtering

- *Trilinear filtering*
  - Interpolate between two LODs
  - Final RGB value is between the result of a bilinear filter at one LOD and a second bilinear filter at the next LOD
  - Eliminates “seams” between LODs
  - At least twice as expensive as bilinear filtering

Another Component: Filtering

- *Anisotropic filtering*
  - Basic filtering methods assume that a pixel on-screen maps to a square (isotropic) region of the texture
  - For surfaces tilted away from the viewer, this is not the case!

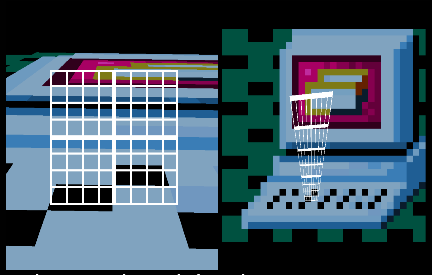
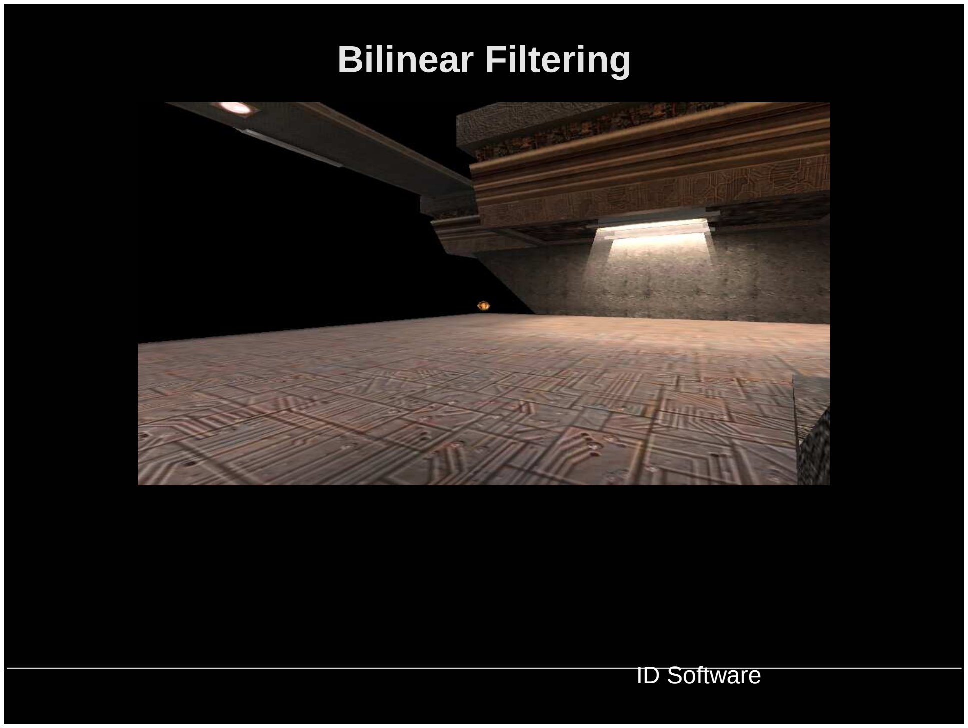


Figure 5. Anisotropic footprints are very common.

Image courtesy of nVidia

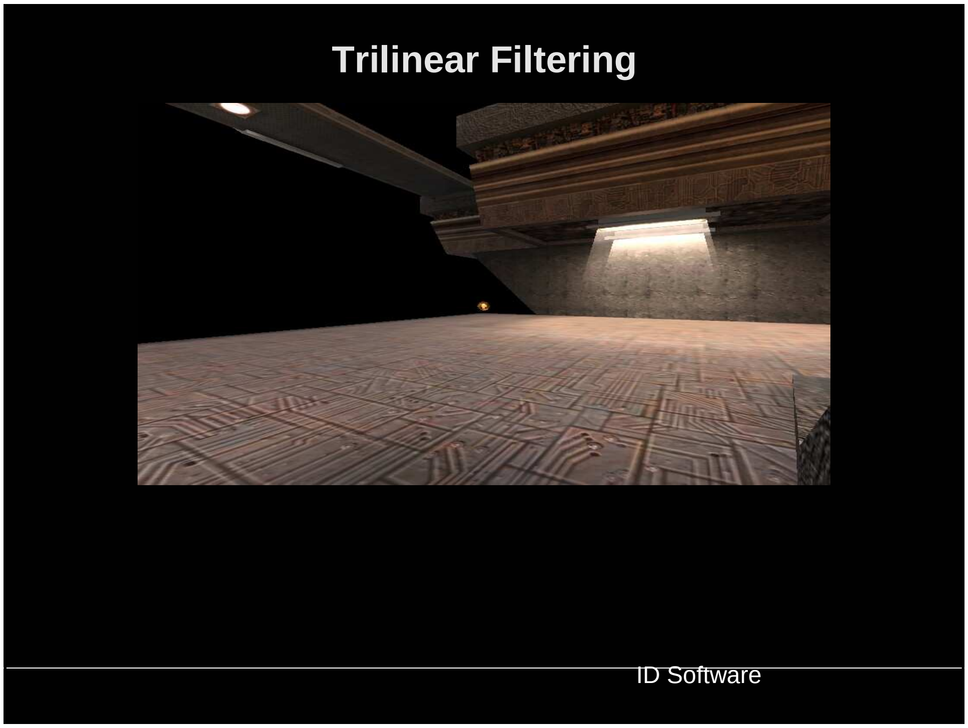
Another Component: Filtering

- *Anisotropic filtering*
  - A pixel may map to a rectangular or trapezoidal section of texels—shape filters accordingly and use either bilinear or trilinear filtering
  - Complicated, but produces very nice results



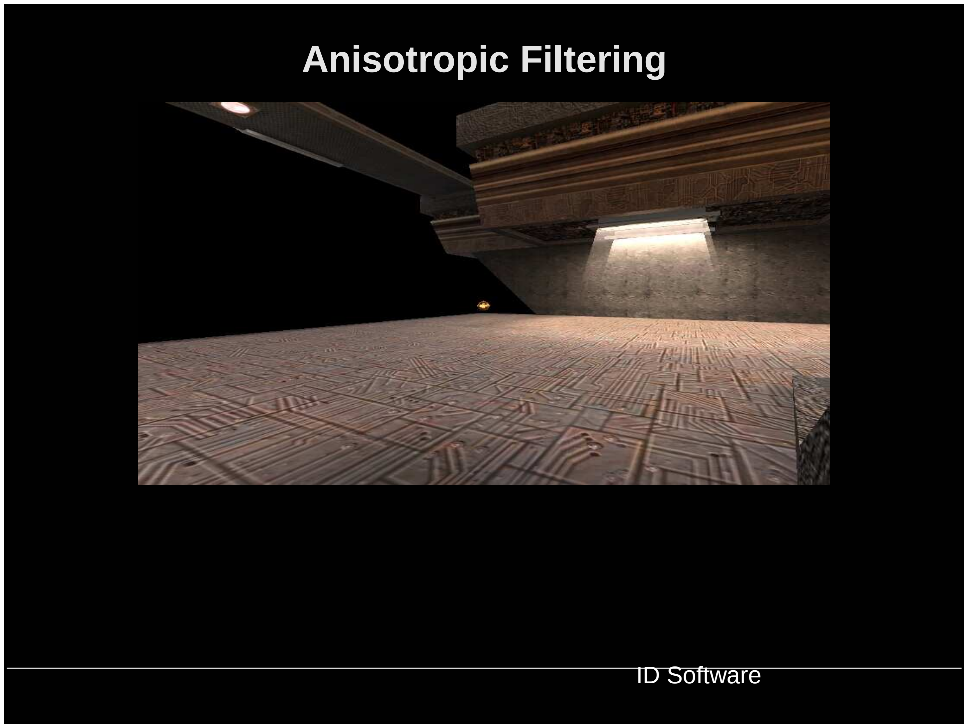
Bilinear Filtering

ID Software



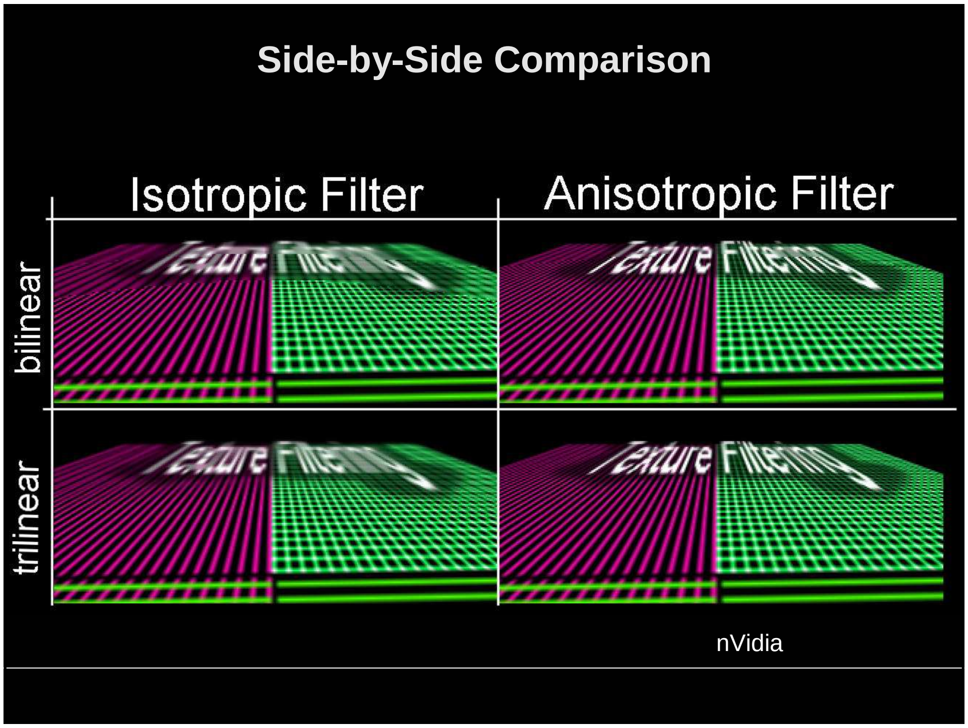
Trilinear Filtering

ID Software



Anisotropic Filtering

ID Software



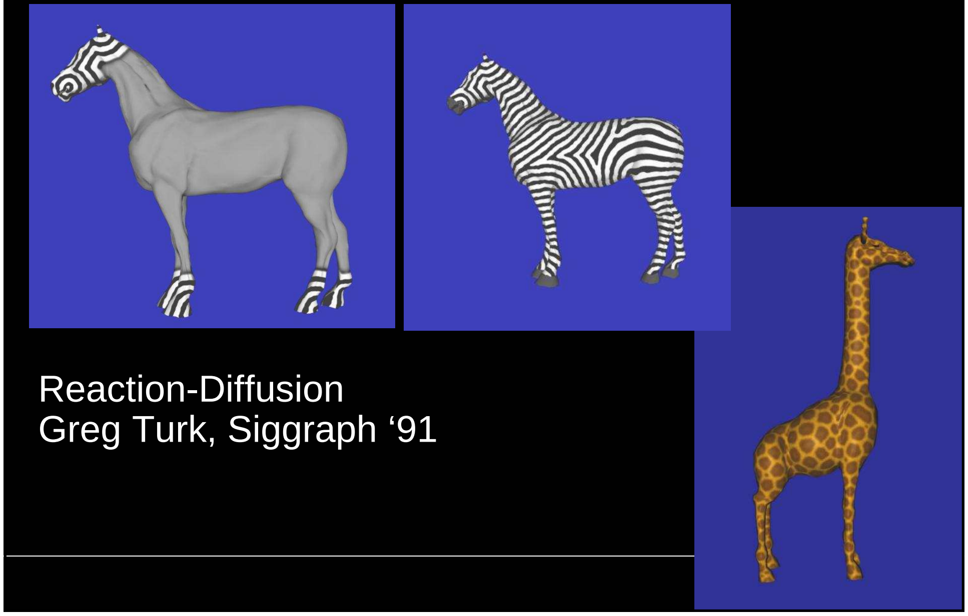
Side-by-Side Comparison

nVidia

Texture Generation

Photographs
Drawings
Procedural methods (2D or 3D)
- Associate each x,y,z value directly with an s,t,r value in the texture block (sculpting in marble and granite)

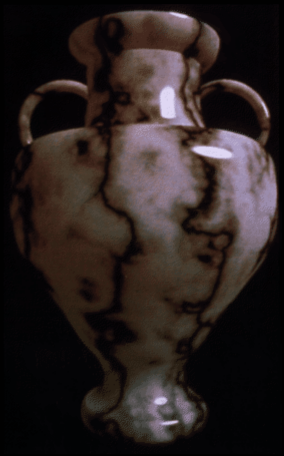
Procedural Methods



Reaction-Diffusion
Greg Turk, Siggraph ‘91

Solid Textures

- Have a 3-D array of texture values (e.g., a block of marble)
  - Use a function [xyz] -> [RGB] to map colors to points in space
- Such a 3D map is called a *solid texture map*
- In practice the map is often defined procedurally
  - No need to store an entire 3D array of colors
  - Just define a function to generate a color for each 3D point
- The most interesting solid textures are random ones
  - a great marble algorithm has now become cliché
- Evaluate the texture coordinates in object coordinates - otherwise moving the object changes its texture!



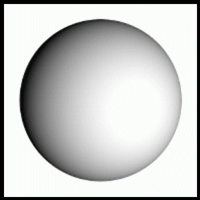
From: *An Image Synthesizer*
by Ken Perlin, SIGGRAPH '85

Uses for Texture Mapping

Use texture to affect a variety of parameters

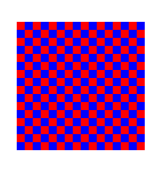
• surface color (Catmull 1974)	- color (radiance) of each point on surface
• surface reflectance	- reflectance coefficients k_d, k_s, or n_{shiny}
• normal vector	- bump mapping (Blinn 1978)
• geometry	- displacement mapping
• transparency	- transparency mapping (clouds) (Gardener 1985)
• light source radiance	- environment mapping (Blinn 1978)

Radiance vs. Reflectance Mapping



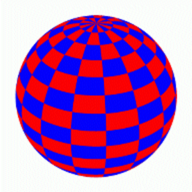
Sphere w/ Uniform Diffuse coefficient

+



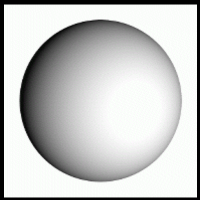
Radiance Map

=



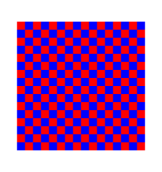
Sphere w/ Radiance Map

Texture specifies (isotropic) radiance for each point on surface



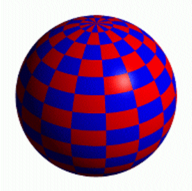
Sphere w/ Uniform Diffuse coefficient

+



Reflectance (k_d) Map

=



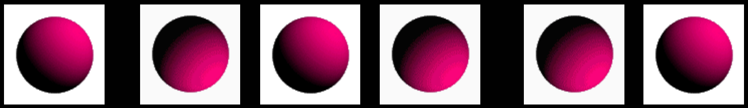
Sphere w/ Reflectance Map

Texture specifies diffuse color (k_d coefficients) for each point on surface
- three coefficients, one each for R, G, and B radiance channels

---

Bump Mapping: A Dirty Trick

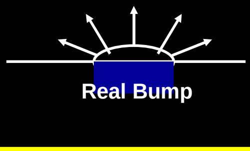
- Which spots bulge out, and which are indented?

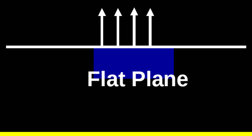


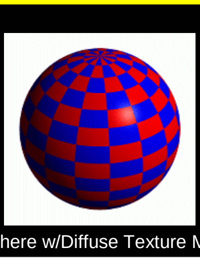
- Answer: None! This is a flat image.
- The human visual system is hard-coded to expect light from above
- In CG, we can perturb the normal vector without having to make any actual change to the shape.

Bump Mapping

- Basic texture mapping paints on to a smooth surface
- How do you make a surface look *rough*?
  - – Option 1: model the surface with many small polygons
  - – Option 2: perturb the normal vectors before the shading calculation



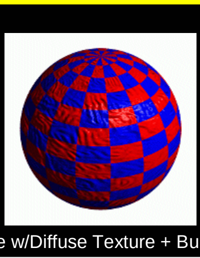



Sphere w/Diffuse Texture Map

\+


Bump Map

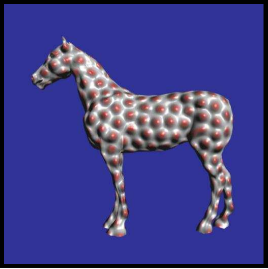
=

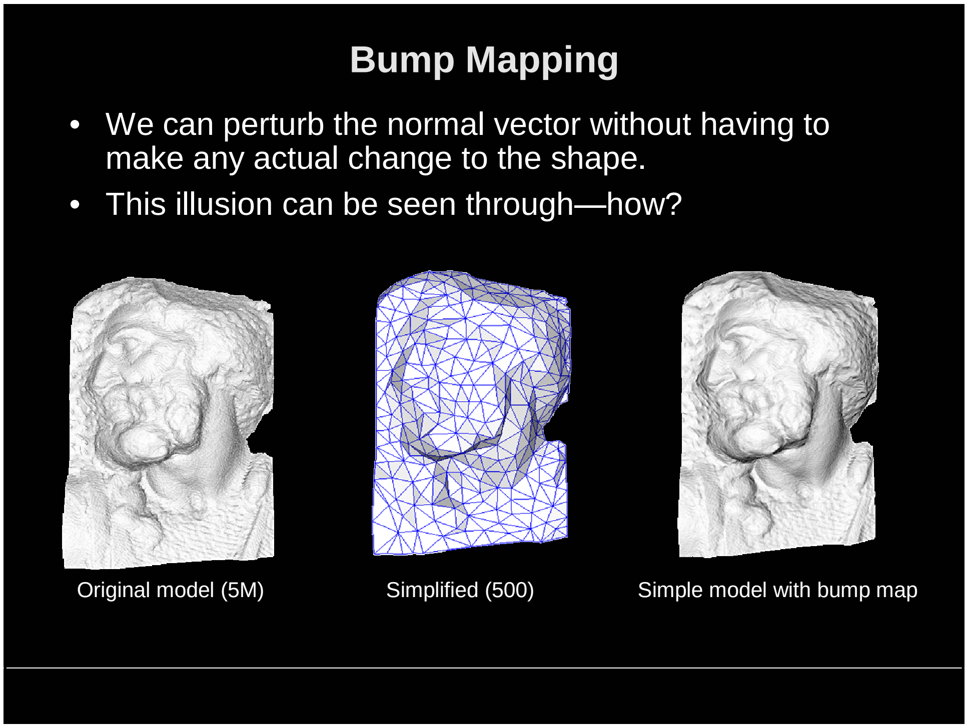

Sphere w/Diffuse Texture + Bump Map

Bump Mapping

- Basic texture mapping paints on to a smooth surface
- How do you make a surface look *rough*?
  - – Option 1: model the surface with many small polygons
  - – Option 2: perturb the normal vectors before the shading calculation
    - » the surface doesn’t actually change, but shading makes it look that way
    - » bump map fakes small displacements above or below the true surface
    - » can use texture-mapping for this
      - – texture image gives amount to perturb surface normal

What kind of anomaly will this produce?


Greg Turk



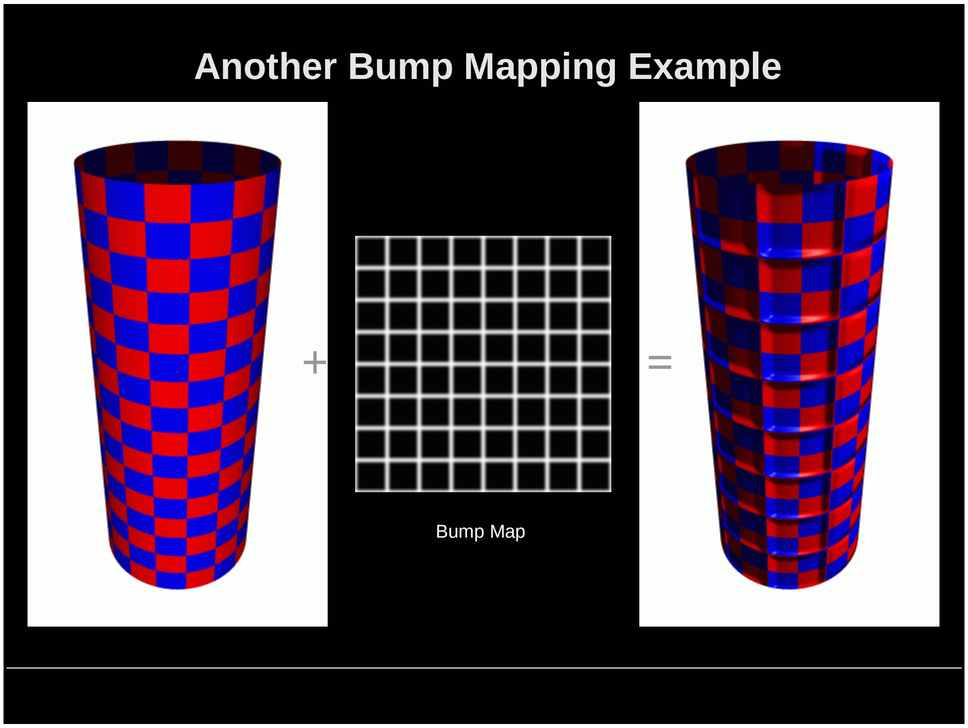
Bump Mapping

- We can perturb the normal vector without having to make any actual change to the shape.
- This illusion can be seen through—how?

Original model (5M)

Simplified (500)

Simple model with bump map



Another Bump Mapping Example

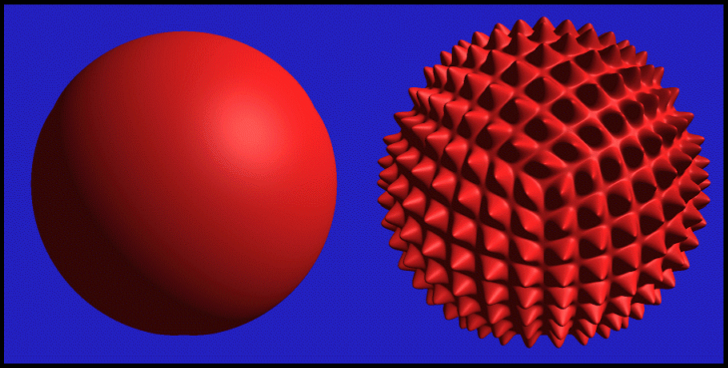
+

Bump Map

=

Displacement Mapping

- Use texture map to displace each point on the surface
  - Texture value gives amount to move in direction normal to surface



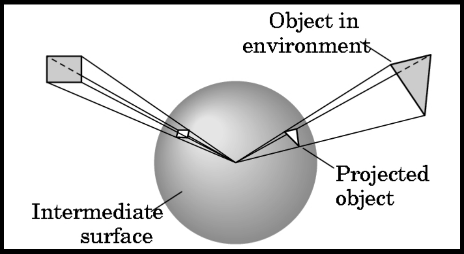
- How is this different from bump mapping?

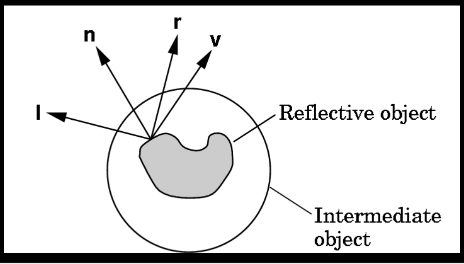
Environment Mapping

Specular reflections that mirror the environment

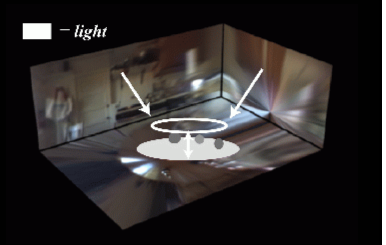
Environment Mapping

Specular reflections that mirror the environment

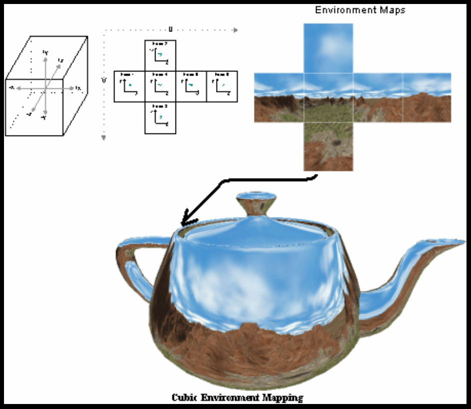




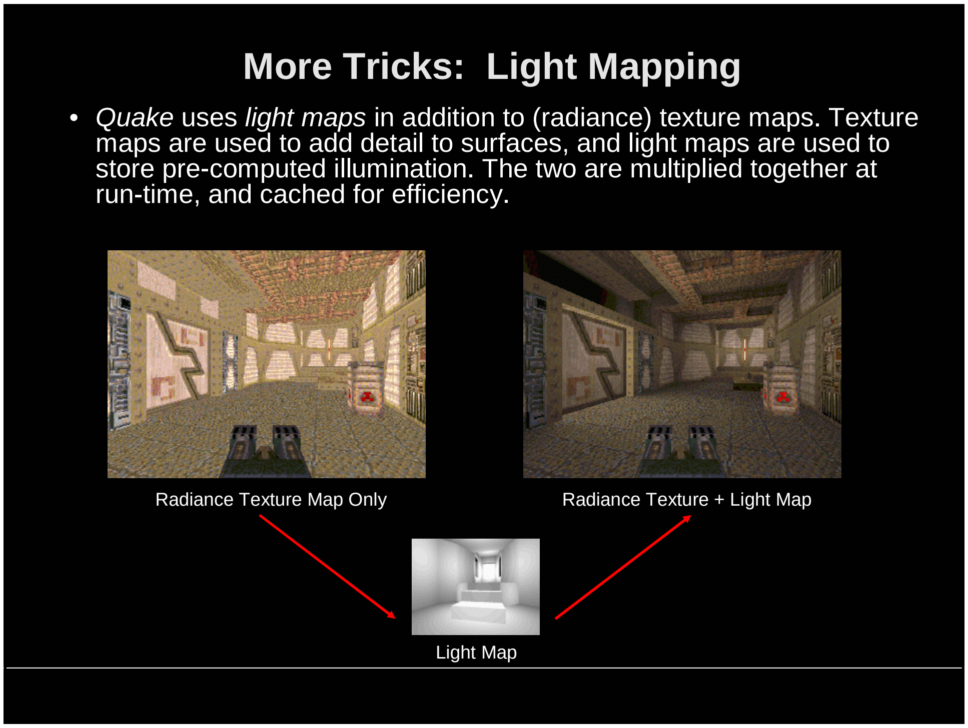
Cube is a natural intermediate object for a room



Environment Mapping: Cube Maps



Cubic Environment Mapping



More Tricks: Light Mapping

- *Quake* uses *light maps* in addition to (radiance) texture maps. Texture maps are used to add detail to surfaces, and light maps are used to store pre-computed illumination. The two are multiplied together at run-time, and cached for efficiency.

Radiance Texture Map Only

Radiance Texture + Light Map

Light Map