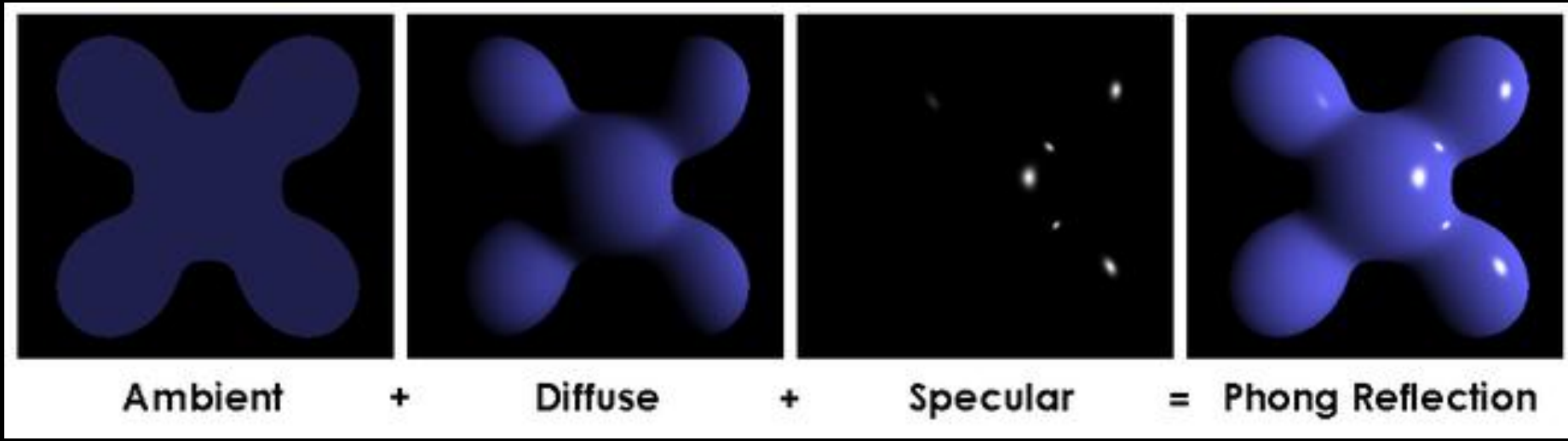
# Review: Phong Illumination Model




$$I_p = k_a i_a + \sum_{\text{m} \in \text{lights}} (k_d (L_m \cdot N) i_d + k_s (R_m \cdot V)^{\alpha} i_s).$$

chalkboard

# Phong vs. Gouraud Shading

Key: What is interpolated when shading a triangle?

chalkboard

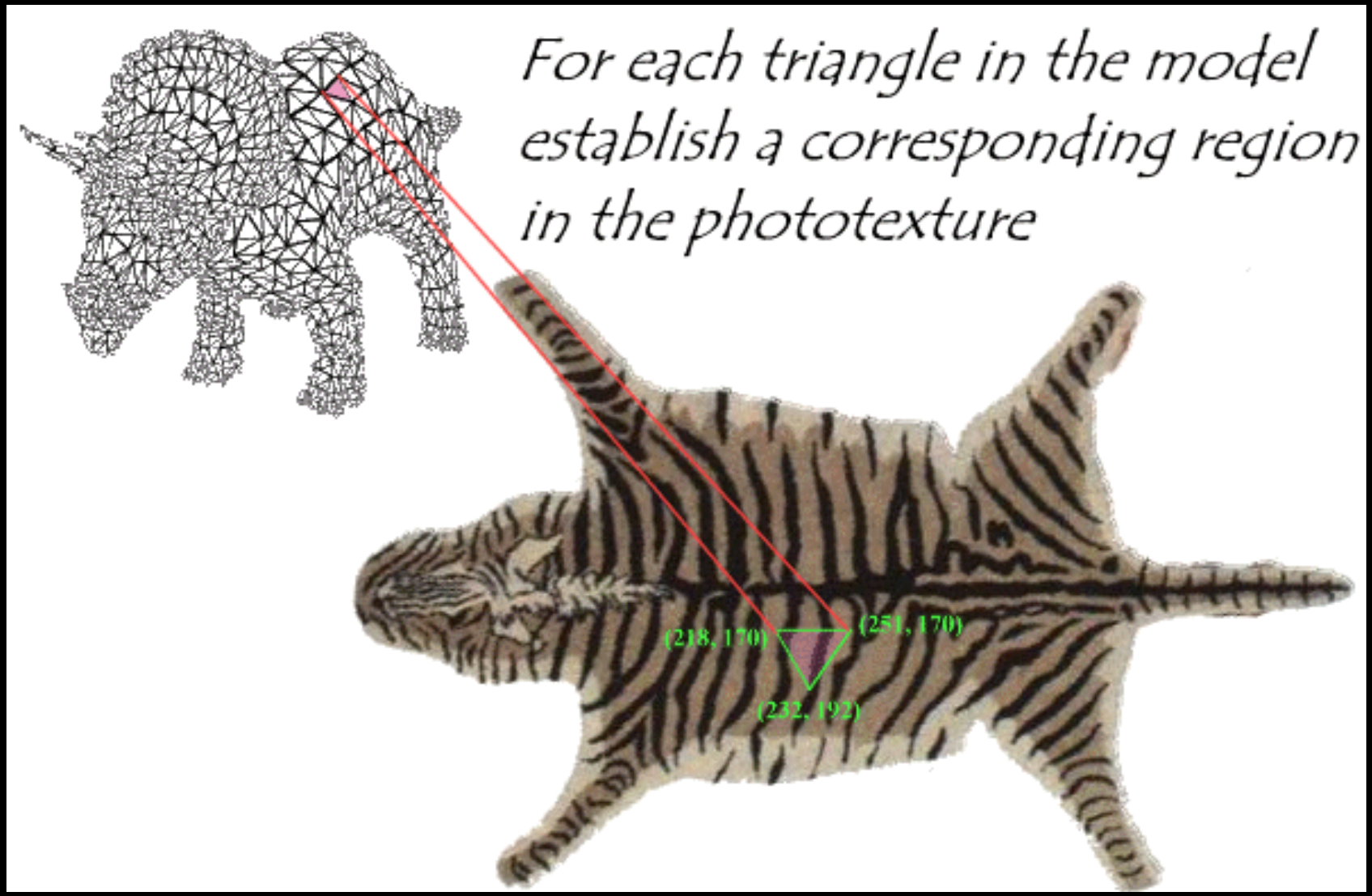
See Shirley, Ch 10 and
http://en.wikipedia.org/wiki/Phong_shading

http://en.wikipedia.org/wiki/Gouraud_shading

# Texture and other Mappings

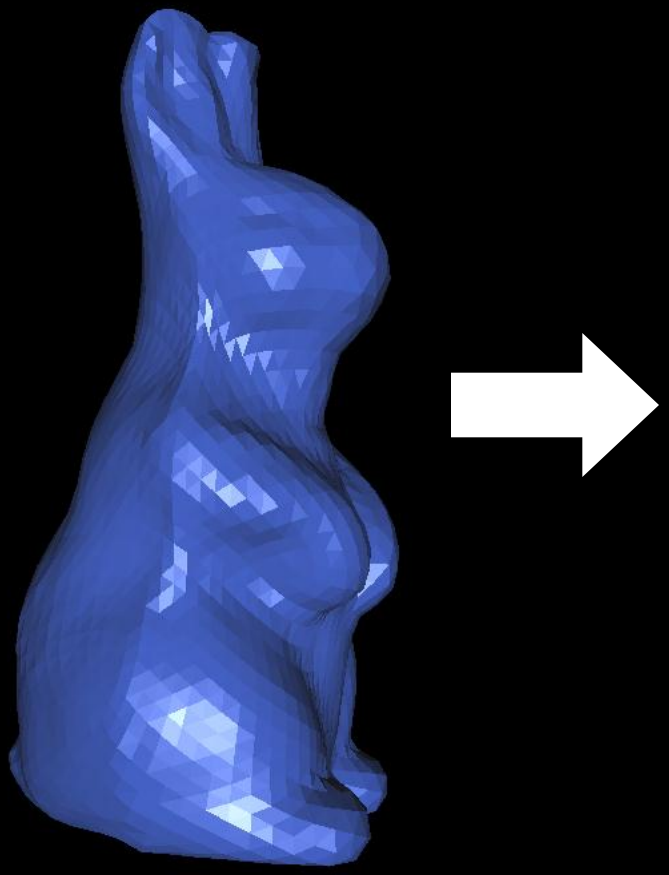
Texture Mapping
Bump Mapping
Displacement Mapping
Environment Mapping

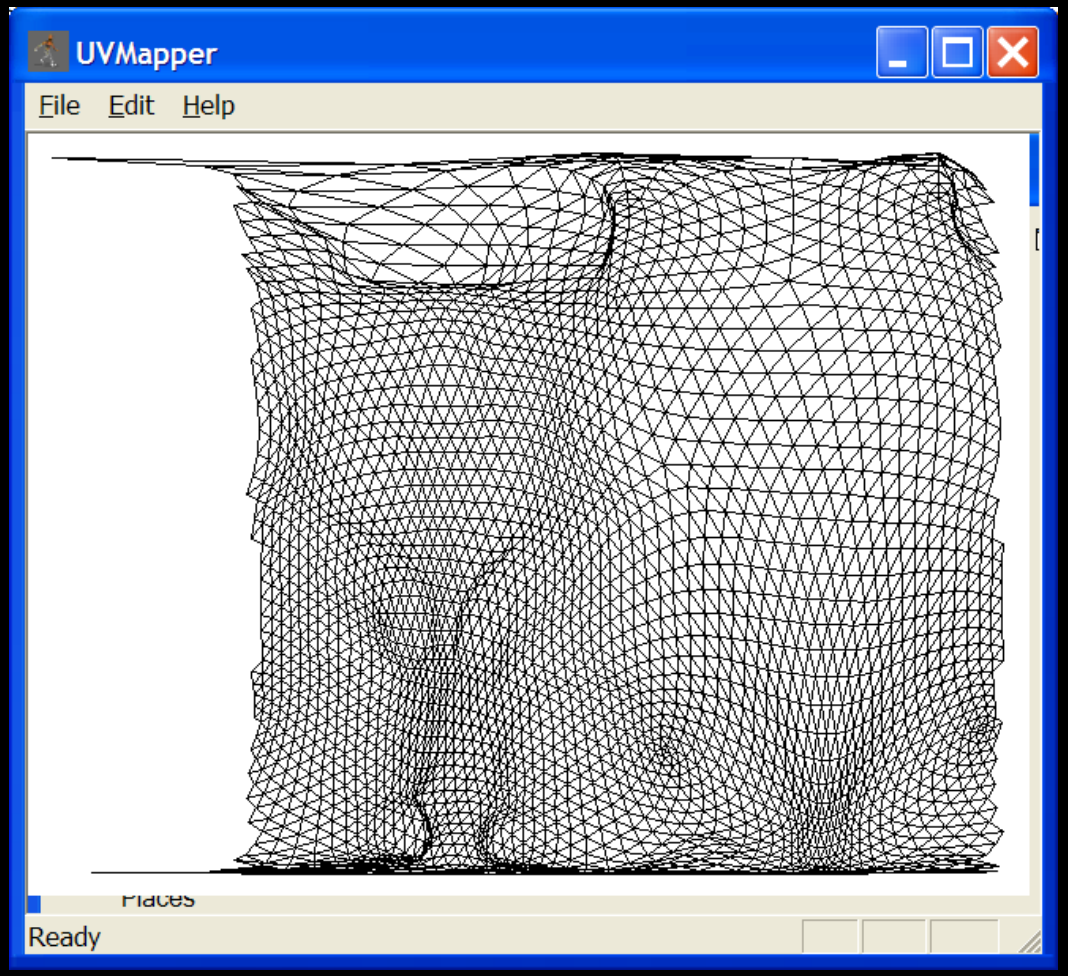
# We *could* specify all texture coordinates by hand…



# Tools help us unroll an object to “paint” it

- www.uvmapper.com

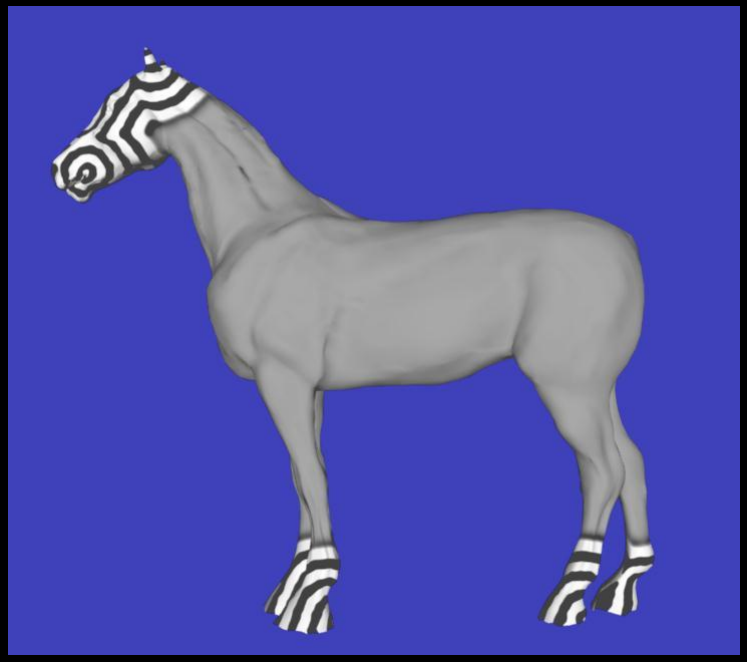


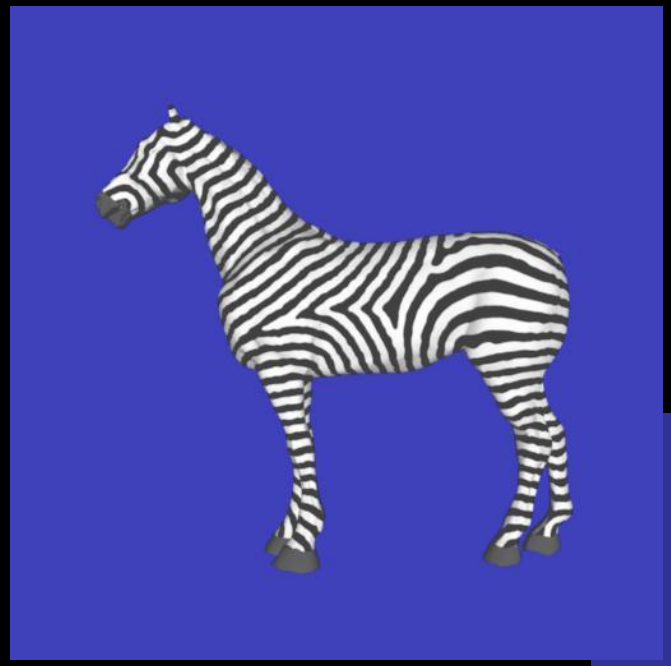


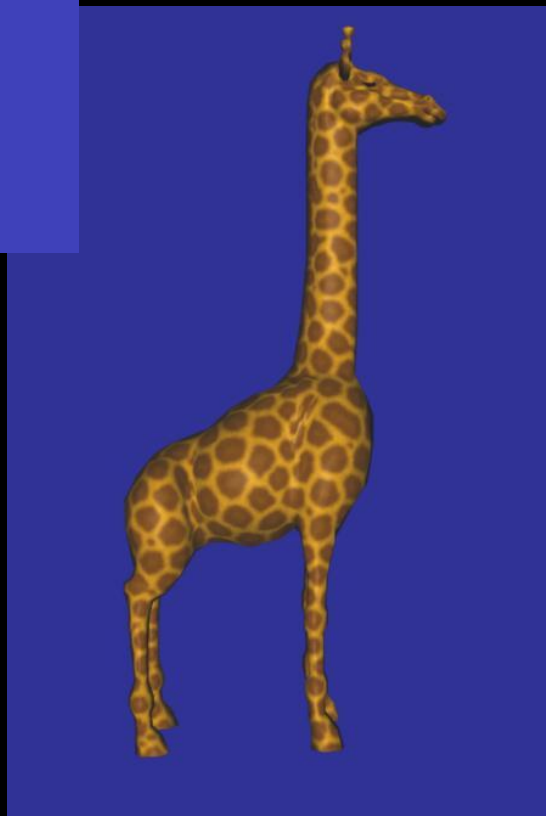
# Texture Generation

Photographs
Drawings
Procedural methods (2D or 3D)
(2D: stripe, wave, and noise patterns
3D: sculpting in marble and granite)

# Procedural Methods







Reaction-Diffusion
Greg Turk, Siggraph ‘91

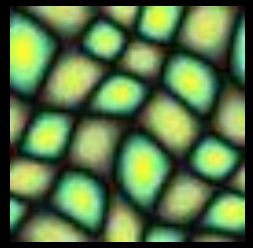
# Solid Textures

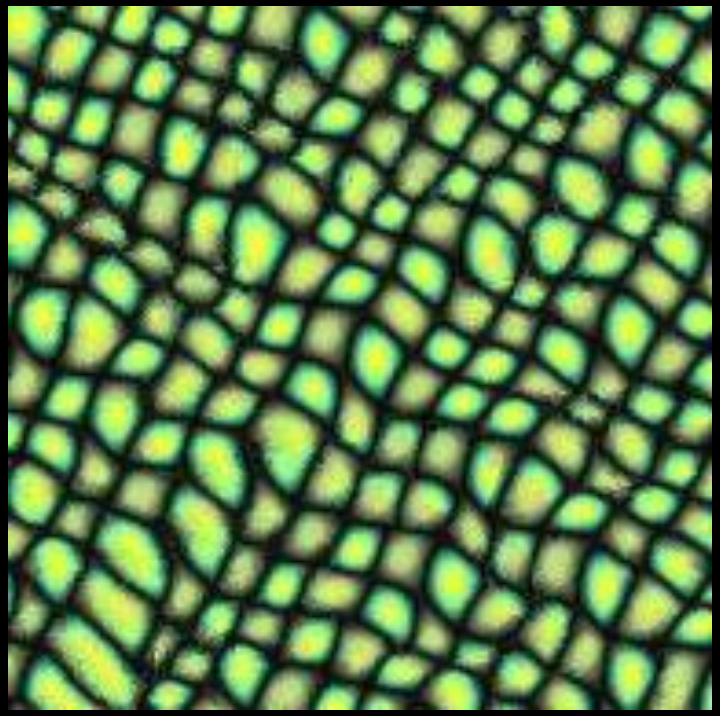
- Have a 3-D array of texture values (e.g., a block of marble)
- In practice the map is often defined procedurally
  - No need to store an entire 3D array of colors
  - Just define a function to generate a color for each 3D point
- The most interesting solid textures are random ones
- Evaluate the texture coordinates in object coordinates - otherwise moving the object changes its texture!
- Ken Perlin's talk "Making Noise"

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

# Data Driven Approaches

- Made popular by Efros and Leung, 1999





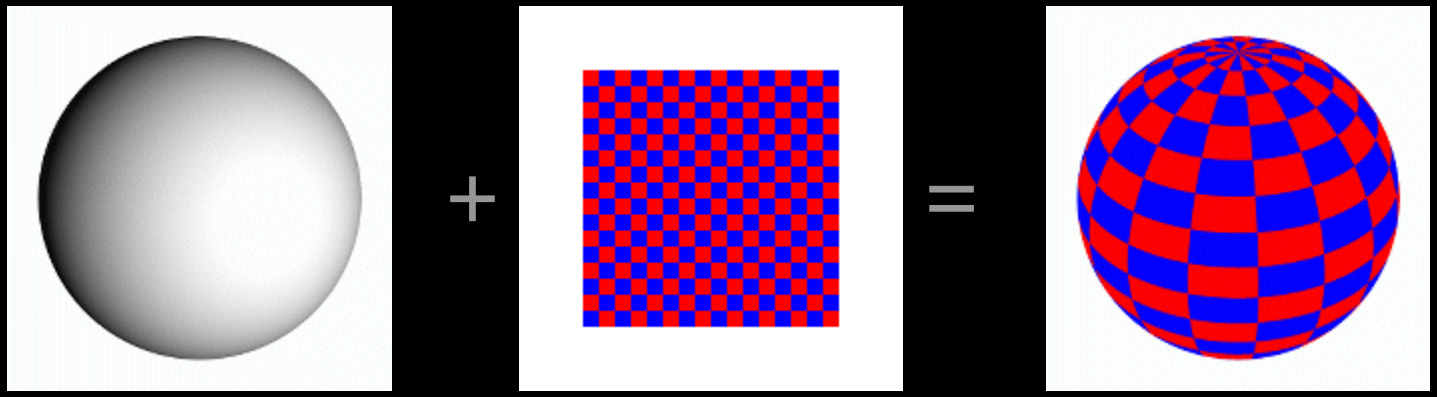
http://graphics.cs.cmu.edu/people/efros/research/EfrosLeung.html

# Uses for Texture Mapping

Use texture to affect a variety of parameters

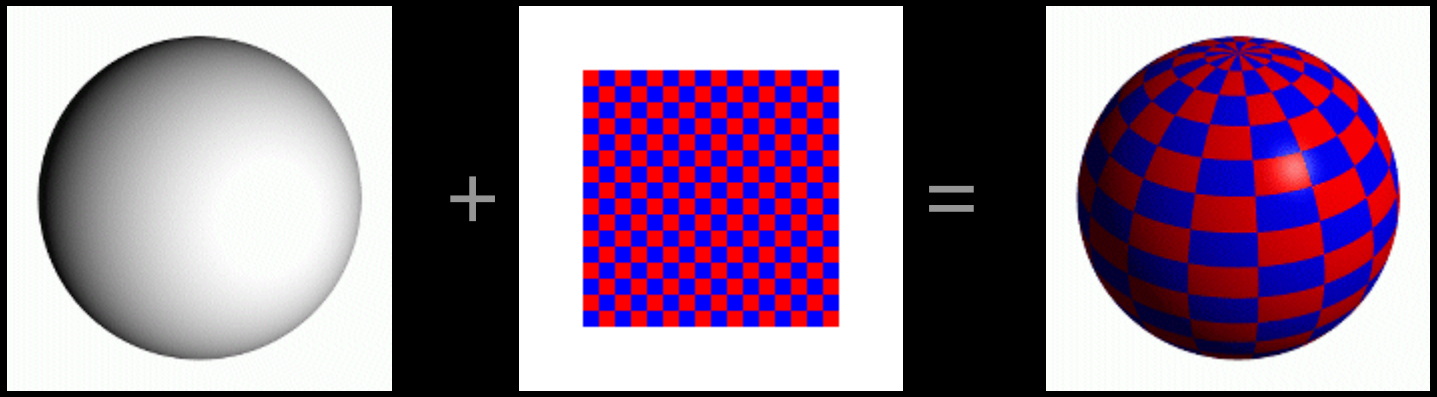
- surface color - radiance of each point on surface (Catmull 1974)
- surface reflectance - reflectance coefficients $k_d$, $k_s$, or $n_{shiny}$
- normal vector - bump mapping (Blinn 1978)
- geometry - displacement mapping
- transparency - transparency mapping (clouds) (Gardener 1985)
- light source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



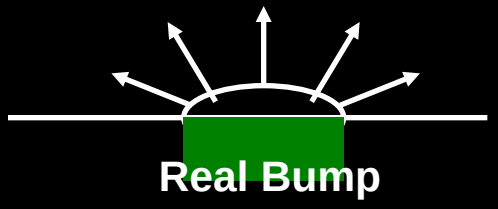
Sphere w/ Uniform Diffuse coefficient  Reflectance ($k_d$) Map  Sphere w/ Reflectance Map

**Texture specifies diffuse color ($k_d$ coefficients) for each point on surface**

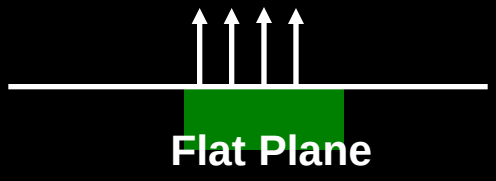
**- three coefficients, one each for R, G, and B radiance channels**

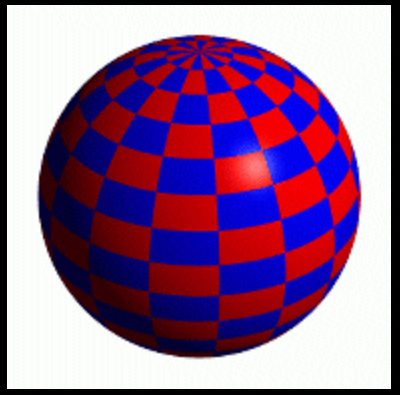
# Bump Mapping

- 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







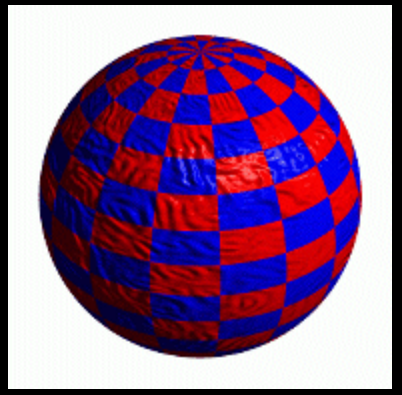



Sphere w/Diffuse Texture Map

+

Bump Map

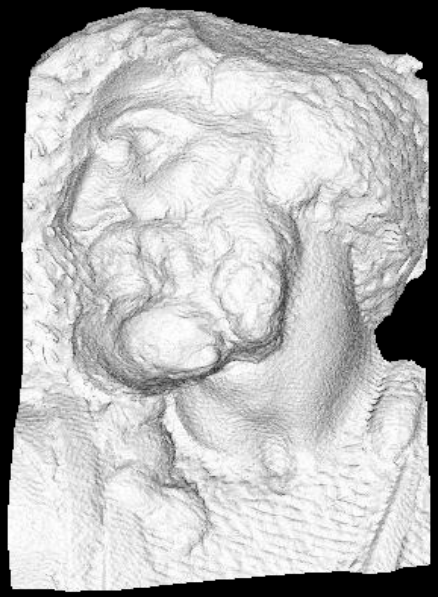
=



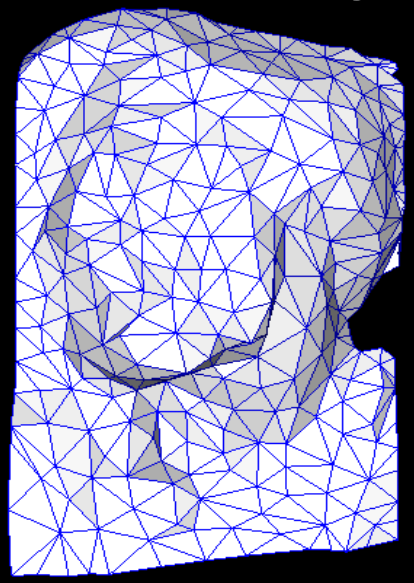
Sphere w/Diffuse Texture + Bump Map

# Bump Mapping

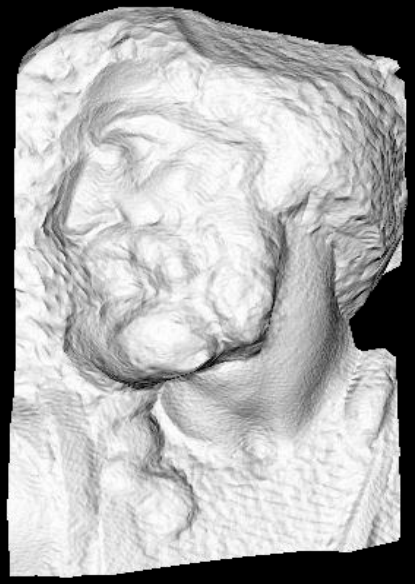
- We can perturb the normal vector without having to make any actual change to the shape.
- This illusion can be seen through—how?



Original model (5M)



Simplified (500)

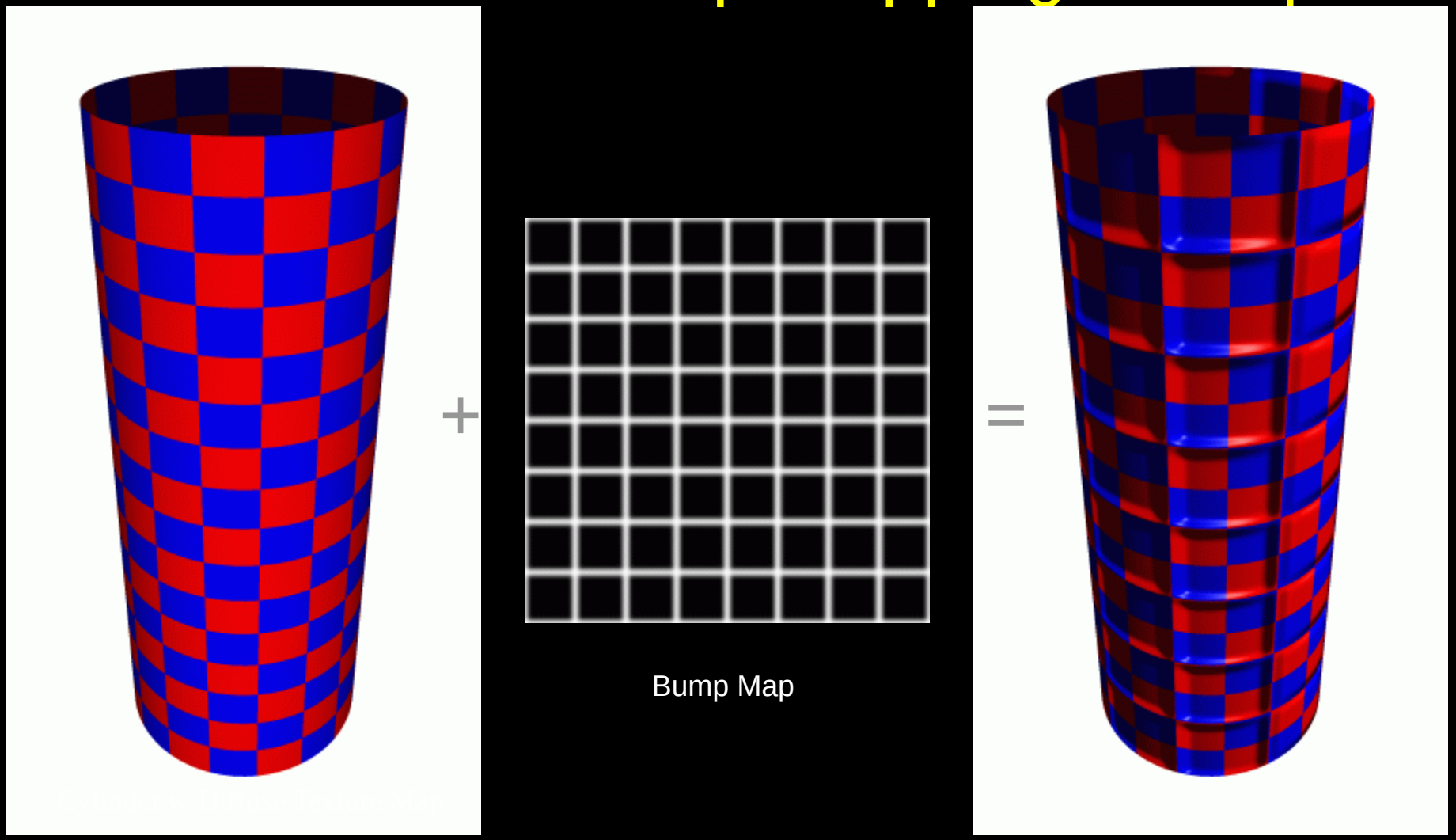


Simple model with bump map

# Bump Mapping

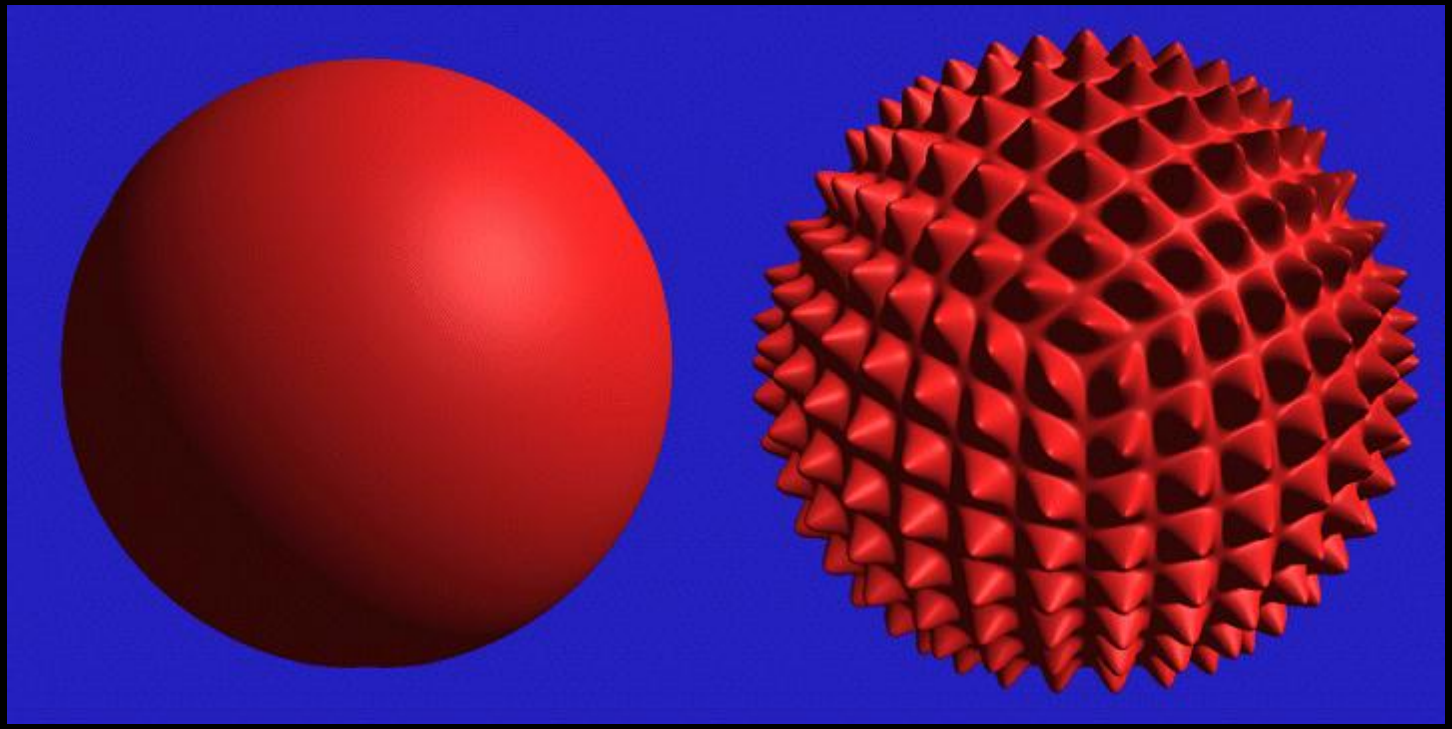
Greg Turk

# Another Bump Mapping Example



# Displacement Mapping

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



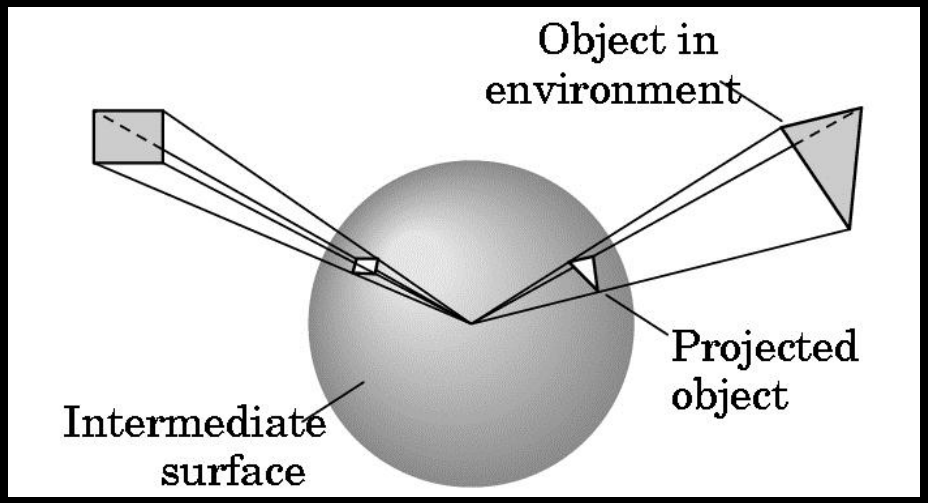
- How is this different from bump mapping?

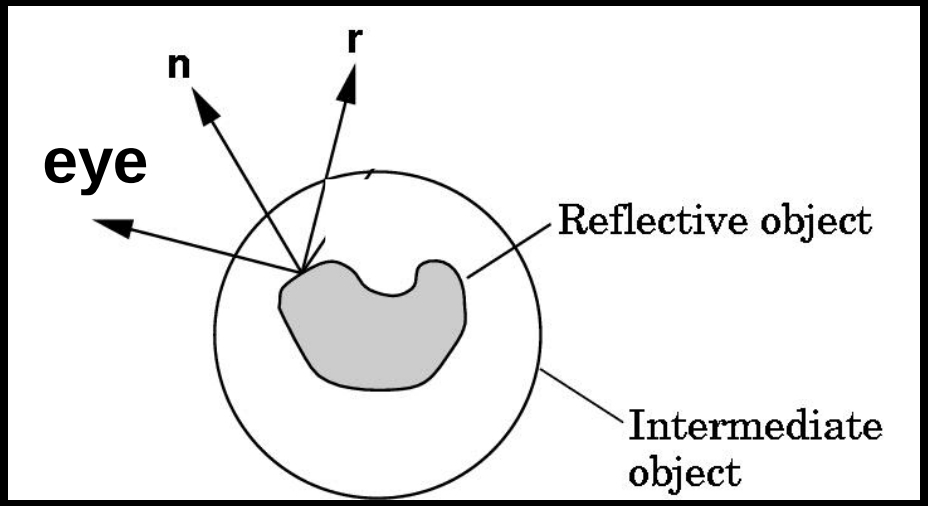
# Environment Mapping

Specular reflections that mirror the environment

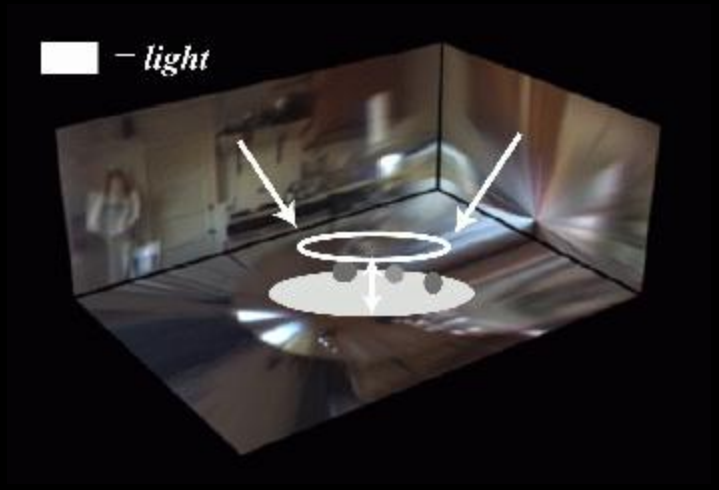
# Environment Mapping

Specular reflections that mirror the environment

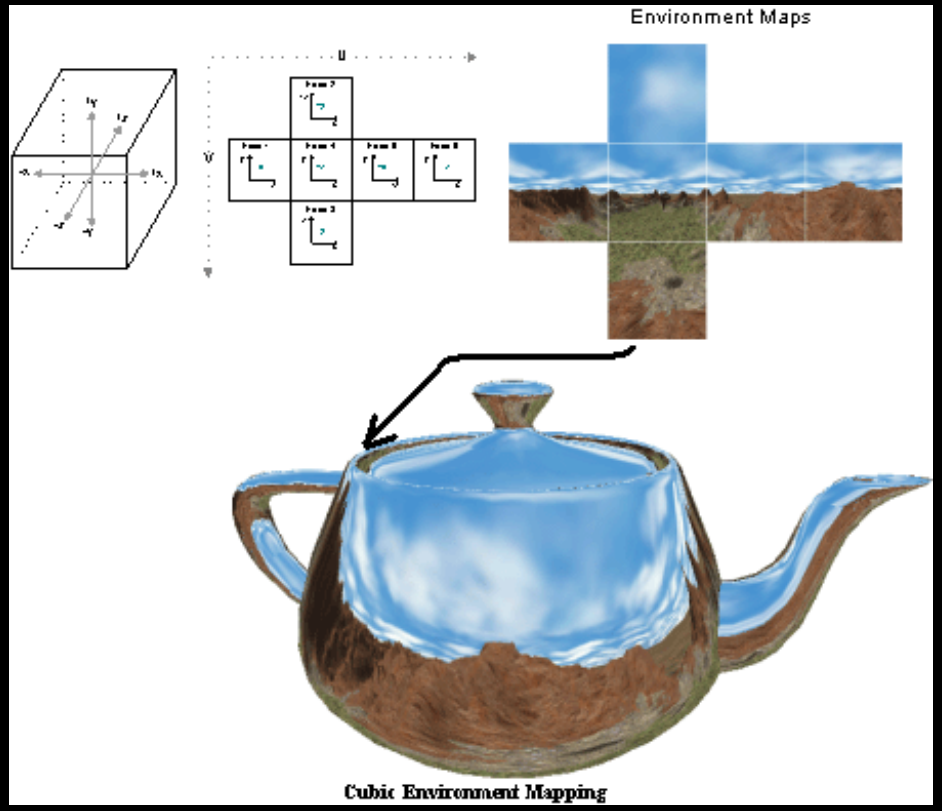



Cube is a natural intermediate object for a room



# Environment Mapping: Cube Maps



# Paul Debevec:  Image-Based Lighting

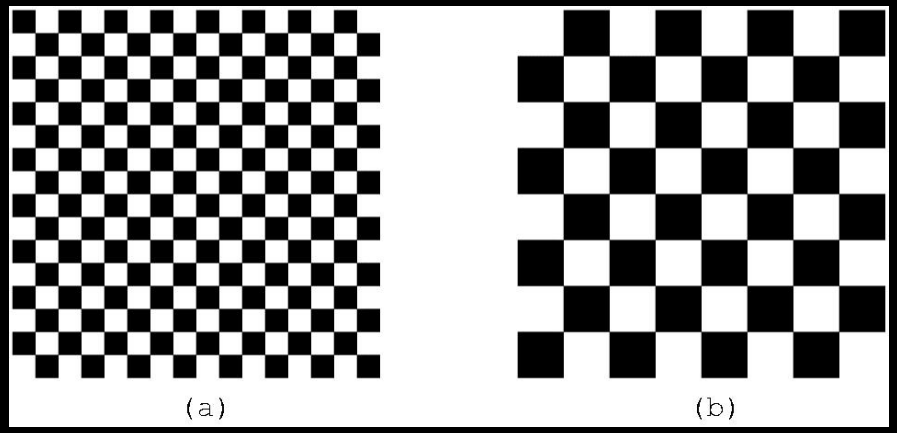

http://www.pauldebevec.com/IBL2003/

# Basics of Texture Mapping in OpenGL

```
Glubyte my_texels[512][512][3];
Gluint texID;

glGenTextures(1, &texID);
glBindTexture(GL_TEXTURE_2D, texID);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 512, 512, 0,
             GL_RGB,GL_UNSIGNED_BYTE, my_texels);
/* level, components, w, h, border, format, type, tarray */

/* assign texture coordinates */
glEnable(GL_TEXTURE_2D);
glBegin(GL_QUAD);
       glTexCoord2f(0.0, 0.0);
       glVertex3f(x1,y1,z1);
       glTexCoord2f(1.0, 0.0);
       glVertex3f(x2,y2,z2);
       glTexCoord2f(1.0,1.0);
       glVertex3f(x3,y3,z3);
       glTexCoord2f(0.0,1.0);
       glVertex3f(x4,y4,z4);
glEnd();
glDisable(GL_TEXTURE_2D);
```



# 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

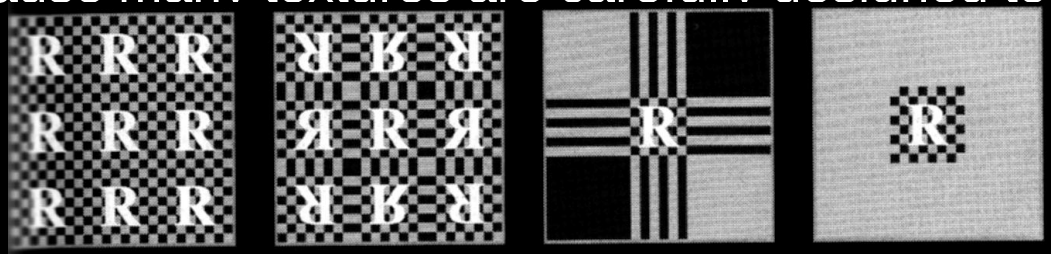
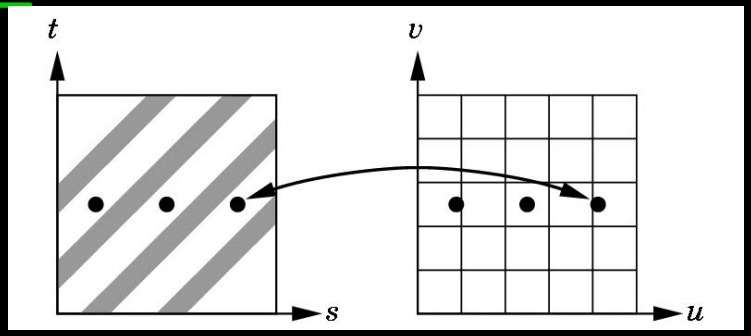


Figure 5.6. Image texture repeat, mirror, clamp, and border functions in action.

- 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**

# Practice Problems

How might you create an environment map to capture lighting effects from a real-world outdoor scene? (Think of Paul Debevec's "Rendering with Natural Light" scene shown in class.) Assume only tools that you may have in your home (e.g., a camera).

# Practice Problems

When would you choose a displacement map over a bump map? Which is likely to be more difficult to implement and why? (Think carefully about the flow of operations in the standard computer graphics pipeline.)

Why do you not get a realistic appearance when using an environment map to render the appearance of a large flat mirror? What visual artifacts would you expect to see if you tried to do this?