Announcements

- **Assignment 2**
  - Theory **due Wednesday**
  - Programming **due next Friday**
    - OpenGL support on mathlab machines has been fixed

- **Office hours**
  - **Today 12-1 pm**

Last week …

- Lighting
  - Phong model components: Ambient, Diffuse, Specular

Diffuse | Specular | Ambient

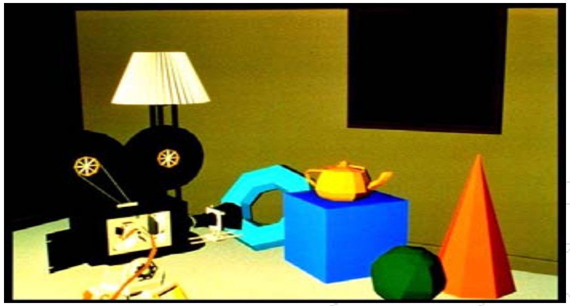
$$\mathbf{L}(\bar{\mathbf{p}},\vec{\mathbf{c}},\vec{\mathbf{s}}) = \mathbf{r_d}\mathbf{I_d}\max(0,\vec{\mathbf{s}}\cdot\vec{\mathbf{n}}) + \mathbf{r_s}\mathbf{I_s}\max(0,\vec{\mathbf{r}}\cdot\vec{\mathbf{c}})^{\alpha} + \mathbf{r_a}\mathbf{I_a}$$

Last week …

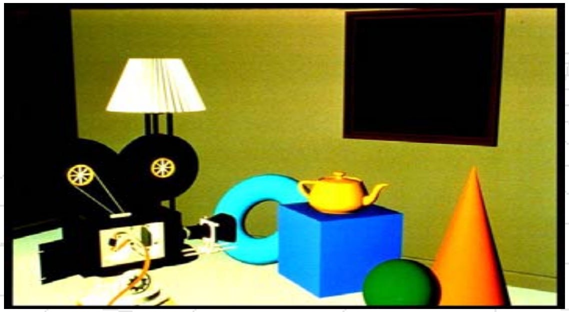
- Lighting
  - Phong model components: Ambient, Diffuse, Specular

- Shading

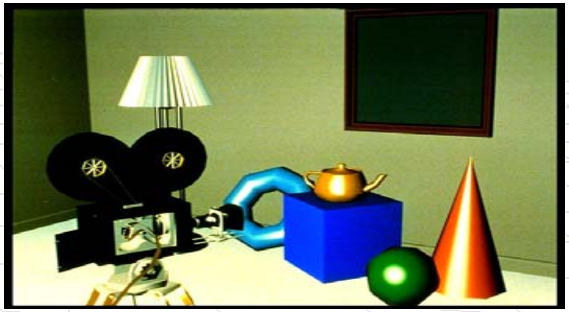
Flat



Gouraud



Phong



Foley, van Dam, Feiner, Hughes

- Scan conversion with shading

Texture Mapping

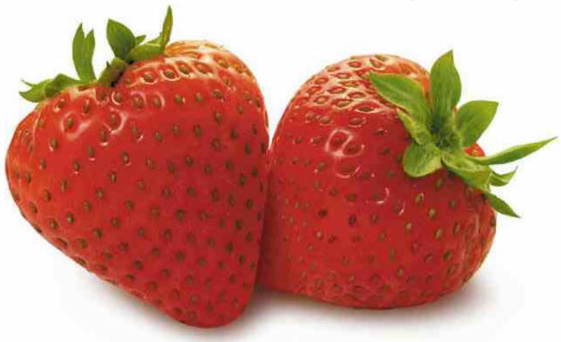
Computer Graphics, CSCD18

Fall 2008

Instructor: Leonid Sigal

Texture Mapping

- So far we only considered objects that have consistent color (that is modulated by light)
- To get more realistic variations in reflectance (that conveys texture) we need to model them
- There are two natural sources of textures
  - **Surface markings** – variations in the total light reflected
  - **Surface relief** – variations in 3D shape which introduce local variability in shading



Why do we need textures?

- An alternative would be to have much more complex models
  - This is **expensive** computationally
  - The tools for building such high fidelity models are not readily available

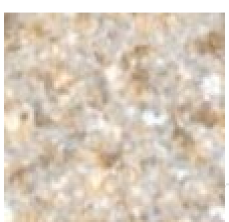
- Textures
  - **Cheaper to render** (especially on current graphics hardware)
  - **Reusable**
    - Once we have the texture (e.g. wood) we can use it for many different objects

Texture Mapping Examples

From http://www.cs.ualberta.ca/~yang/Projects/texture_analysis_and_synthesis.htm

Sky

Parchment



Marble

Texture Mapping

- Texture mapping is also a great way to create artificial objects

Questions we must address

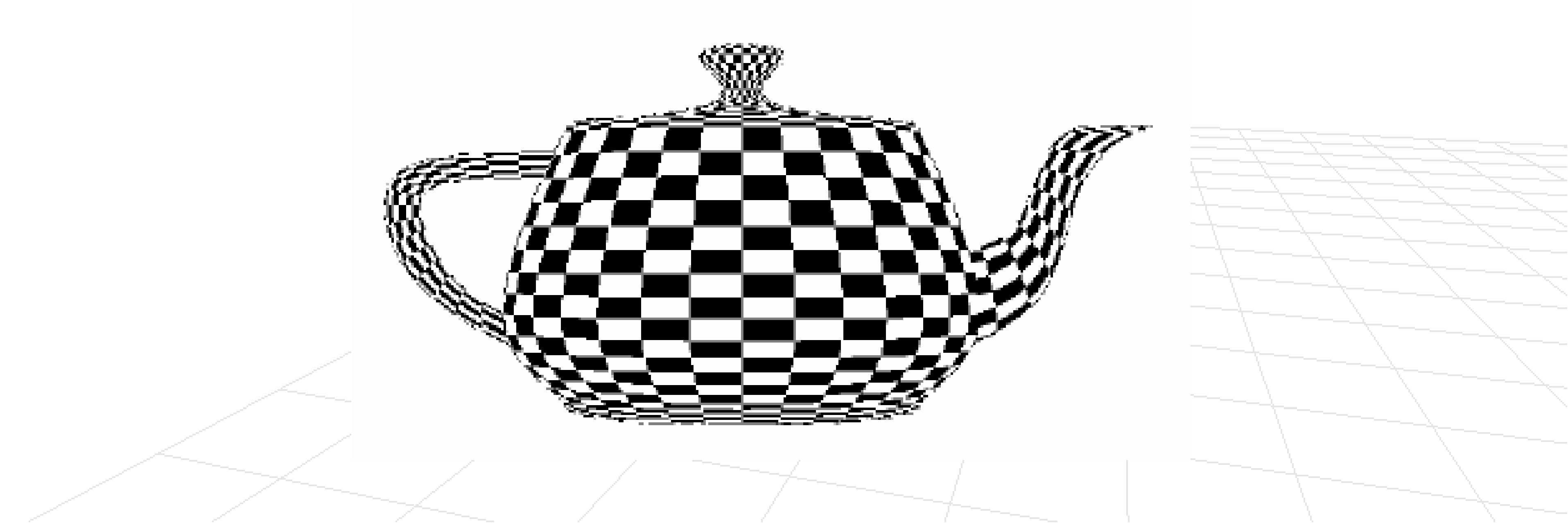
- Where do textures come from?
- How do we map texture onto a surface?
- How does texture change reflectance properties and shading of the surface?
- Scan conversion (how do we actually render texture mapped surface?)

Where do we get a texture?

- Textures can be defined procedurally
  - **Input:** point on the surface
  - **Output:** surface **albedo** at that point

albedo of an object is the extent to which it diffusely reflects light

- Example of procedural texture

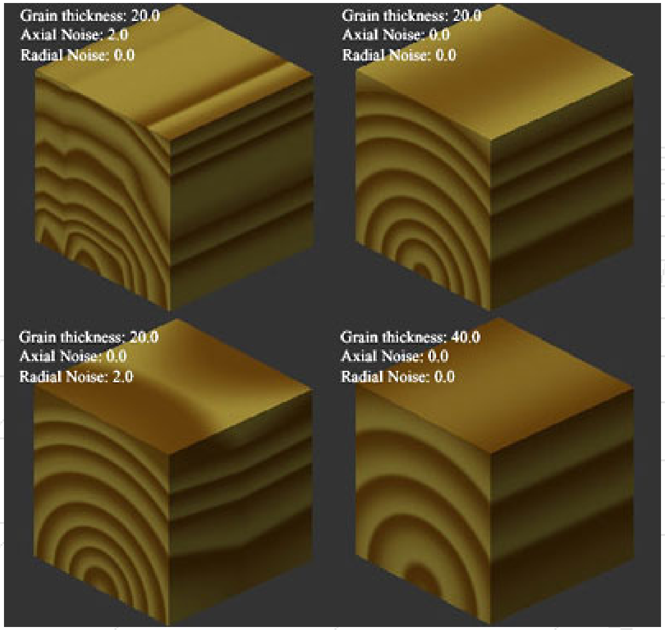


Where do we get a texture?

- Textures can be defined procedurally
  - **Input:** point on the surface
  - **Output:** surface **albedo** at that point

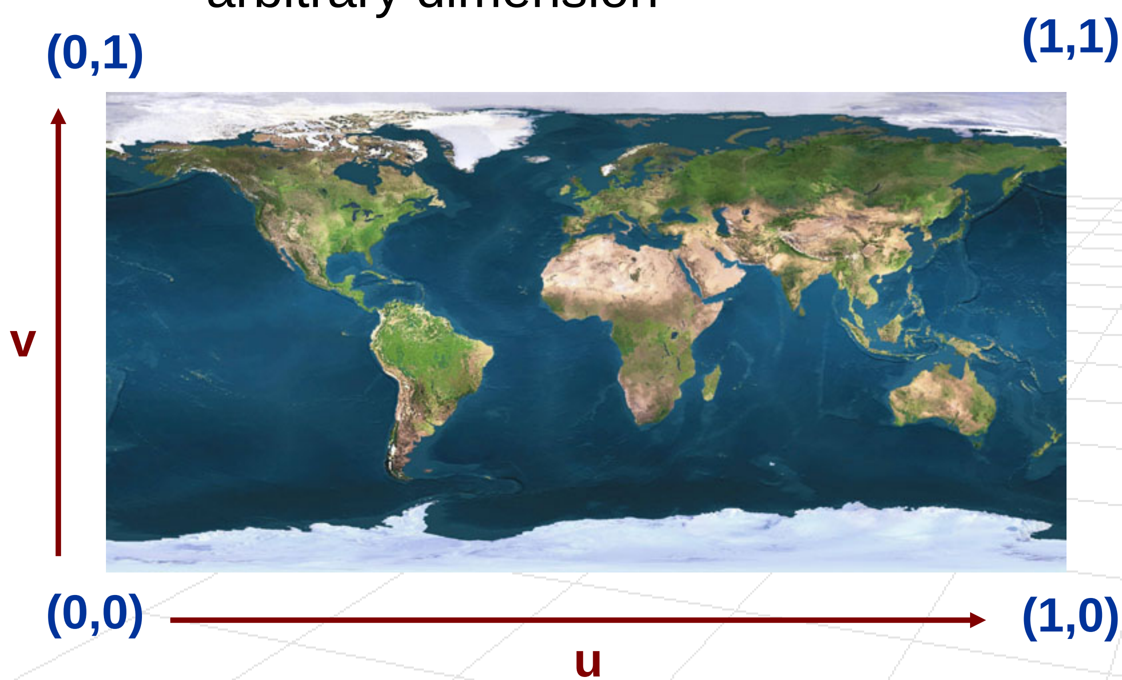
albedo of an object is the extent to which it diffusely reflects light

- Example of procedural texture (in 3D)



Where do we get a texture?

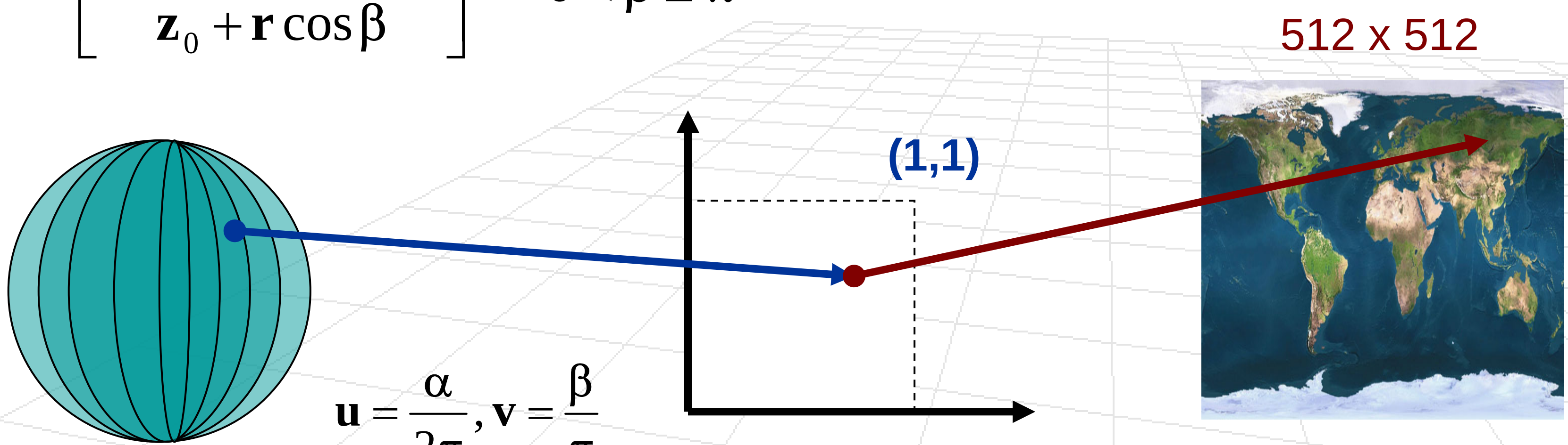
- We can also use digital images as textures
  - Imagine gluing a 2D picture over a 3D surface
- How do we do this?
  - map a point on the arbitrary geometry to a point on an abstract unit square (we call this texture space)
  - map a point on abstract unit square to a point on the image of arbitrary dimension



Texture Mapping Details

- Simplest approaches to texture mapping
  - For each face of the mesh, specify a point $(\mathbf{u_i}, \mathbf{v_i})$ for each vertex point $\mathbf{p_i}$
  - Continuous mapping from parametric form of the surface onto texture, for example for sphere

$$\mathbf{s}(\alpha,\beta) = \begin{bmatrix} \mathbf{x}_0 + \mathbf{r}\cos\alpha\sin\beta \\ \mathbf{y}_0 + \mathbf{r}\sin\alpha\sin\beta \\ \mathbf{z}_0 + \mathbf{r}\cos\beta \end{bmatrix}, \quad \begin{matrix} 0 \le \alpha \le 2\pi \\ 0 < \beta \le \pi \end{matrix}$$



$$\mathbf{u} = \frac{\alpha}{2\pi}, \mathbf{v} = \frac{\beta}{\pi}$$

Questions we must address

- Where do textures come from?
- How do we map texture onto a surface?
- How does texture change reflectance properties and shading of the surface?
- Scan conversion (how do we actually render texture mapped surface?)

What about color texture map?

- Assuming that the texture values are $0 \leq \tau \leq 1$ (we can achieve this by normalizing intensities of the texture map image), we can simply scale the reflection coefficients of ambient and diffuse components of the **Phong model** accordingly

$$\tilde{\mathbf{r}}_{\mathbf{d}} = \tau \, \mathbf{r}_{\mathbf{d}}$$

$$\tilde{\mathbf{r}}_{\mathbf{a}} = \tau \, \mathbf{r}_{\mathbf{a}}$$

- We could also similarly modulate the secular reflectance coefficient as well

What about color texture map?

- Assuming that the texture values are $0 \leq \tau \leq 1$ (we can achieve this by normalizing intensities of the texture map image), we can simply scale the reflection coefficients of ambient and diffuse components of the **Phong model** accordingly

$$\tilde{\mathbf{r}}_{\mathbf{d,R}} = \tau_{\mathbf{R}} \mathbf{r}_{\mathbf{d,R}} \qquad \tilde{\mathbf{r}}_{\mathbf{d,G}} = \tau_{\mathbf{G}} \mathbf{r}_{\mathbf{d,G}} \qquad \tilde{\mathbf{r}}_{\mathbf{d,B}} = \tau_{\mathbf{B}} \mathbf{r}_{\mathbf{d,B}}$$

$$\tilde{\mathbf{r}}_{\mathbf{a,R}} = \tau_{\mathbf{R}} \mathbf{r}_{\mathbf{a,R}} \qquad \tilde{\mathbf{r}}_{\mathbf{a,G}} = \tau_{\mathbf{G}} \mathbf{r}_{\mathbf{a,G}} \qquad \tilde{\mathbf{r}}_{\mathbf{a,B}} = \tau_{\mathbf{B}} \mathbf{r}_{\mathbf{a,B}}$$

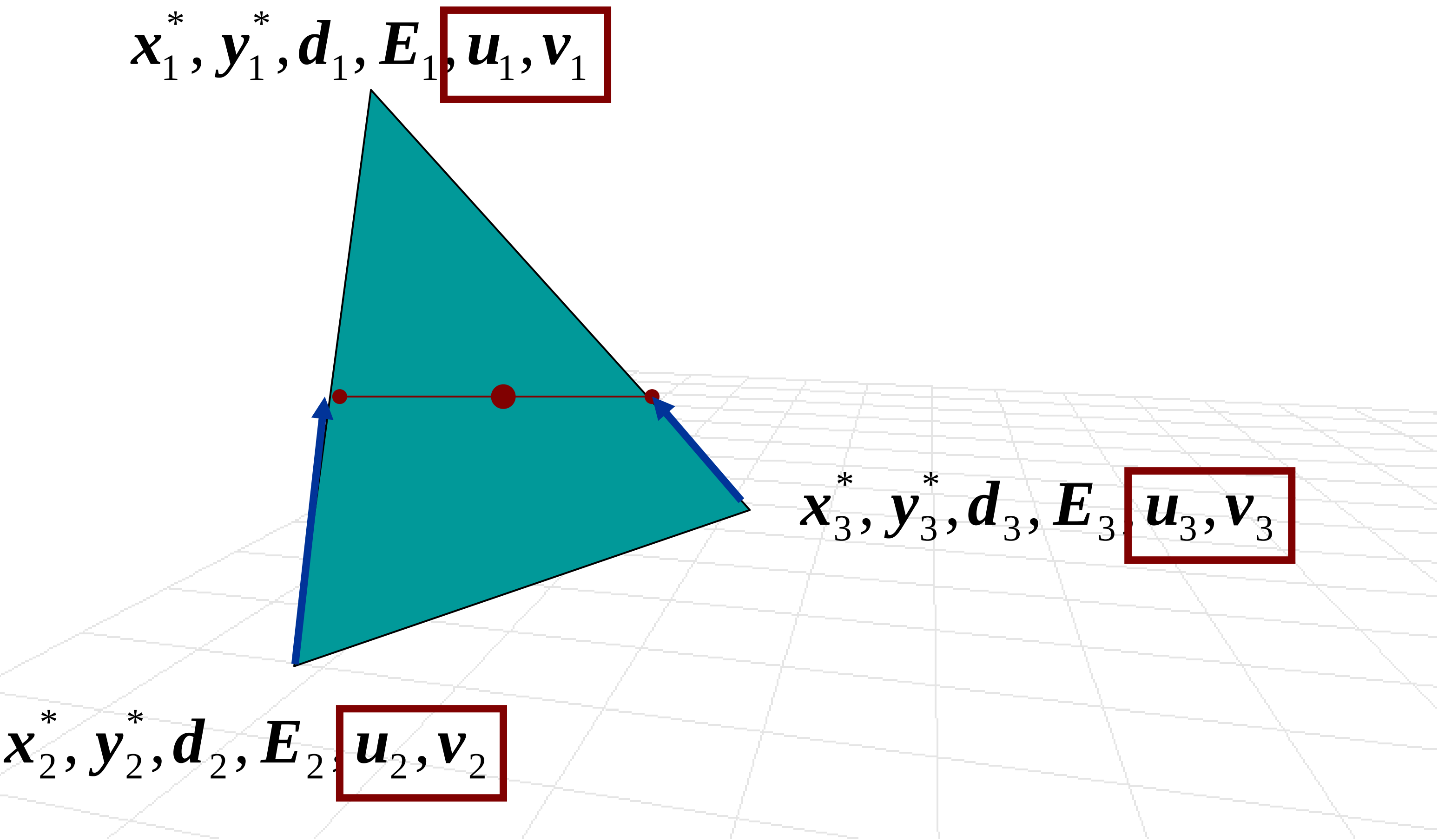
- We could also similarly modulate the secular reflectance coefficient as well

Questions we must address

- Where do textures come from?
- How do we map texture onto a surface?
- How does texture change reflectance properties and shading of the surface?
- Scan conversion (how do we actually render texture mapped surface?)

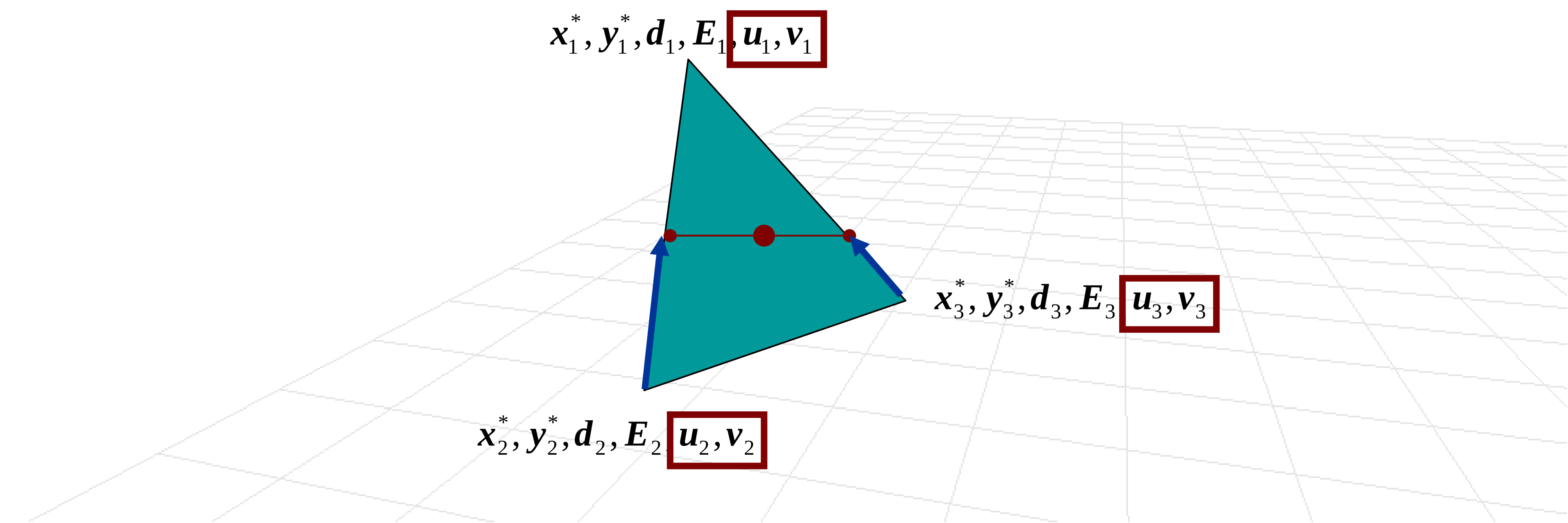
Scan Conversion with Texture Mapping

- Let's try extending the scan conversion algorithm from last class



Scan Conversion with Texture Mapping

- Let's try extending the scan conversion algorithm from last class
  - Linearly interpolate **u**, **v** along with radiance and pseudodepth
  - Scale radiance according to the texture map values



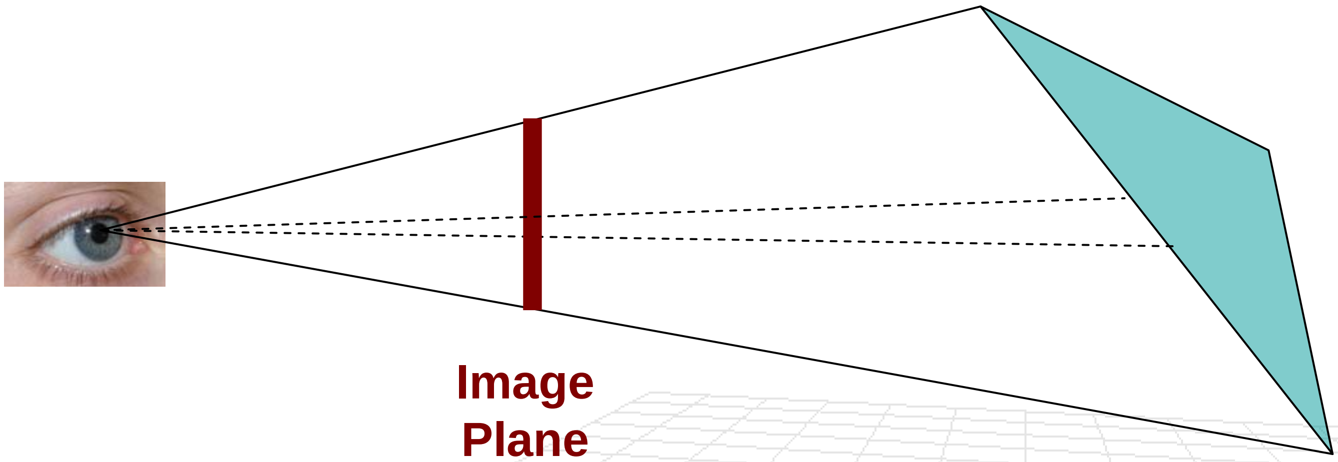
Simple Scan Conversion with Textures

- Let's try extending the scan conversion algorithm from last class
  - Linearly interpolate **u**, **v** along with radiance and pseudodepth
  - Scale radiance according to the texture map values

```
if ( d < z-buffer(x, y) )
        use current u,v to index into the texture map
        to get texture value τ
        scale the radiance value Ẽ = τ E
        putpixel(x, y, Ẽ)
        z-buffer(x, y) = d
end
```

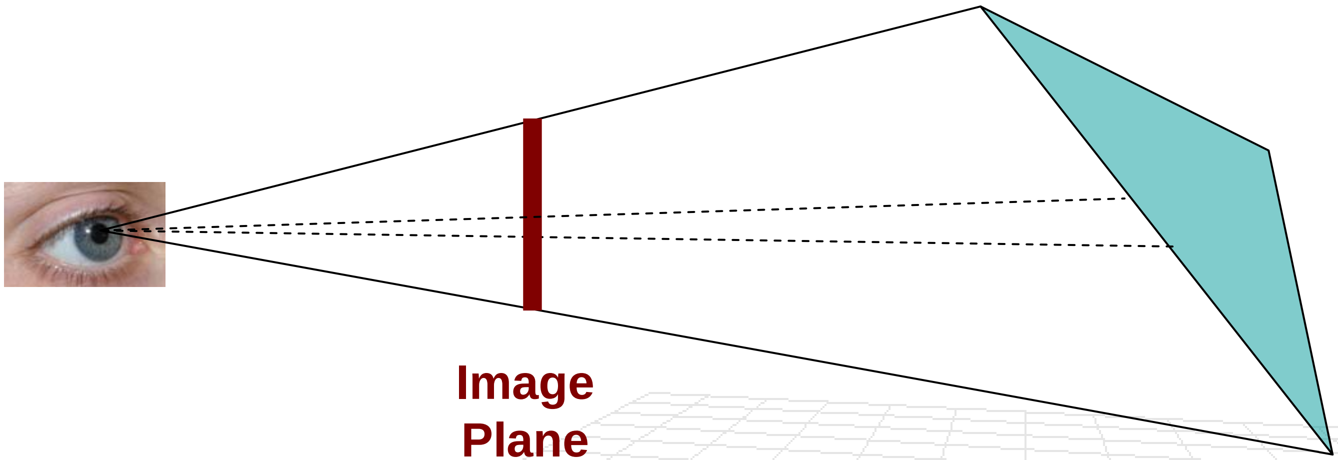
Problems with Simple Scan Conversion

- Perspective projection is non-linear
  - Lines map to lines
  - But, mid-point is not necessarily maps to mid-point



Problems with Simple Scan Conversion

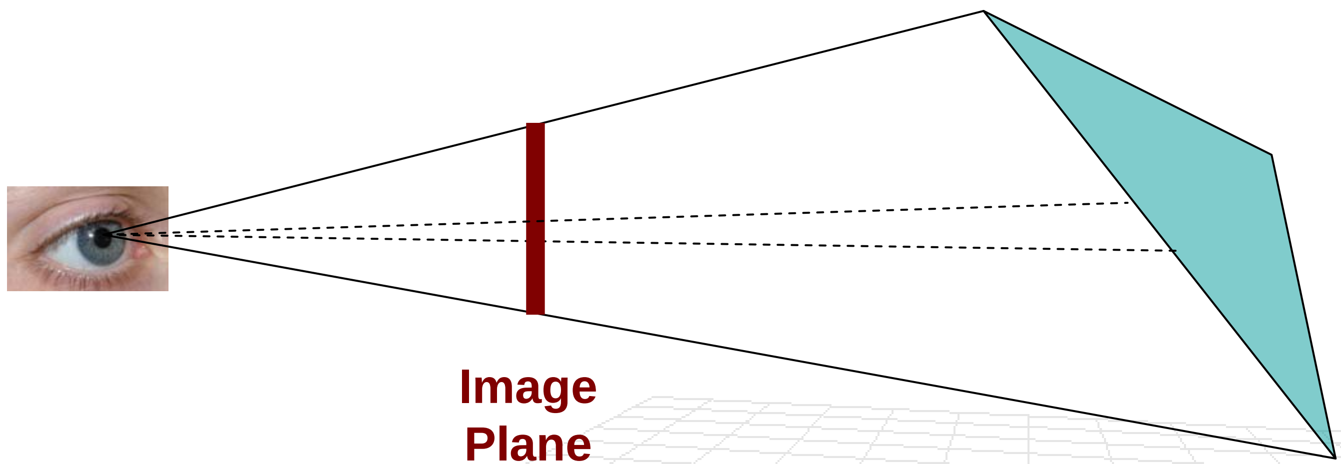
- Perspective projection is non-linear
  - Lines map to lines
  - But, mid-point is not necessarily maps to mid-point



- So, distortion depends on the slope of the surface with respect to line of sight
- **Why did we not care about this before?**

Problems with Simple Scan Conversion

- Perspective projection is non-linear
  - Lines map to lines
  - But, mid-point is not necessarily maps to mid-point

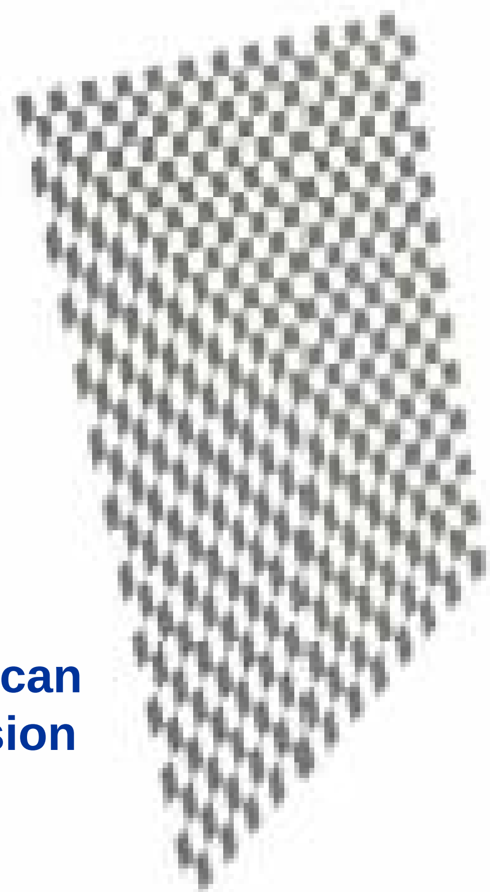


- So, distortion depends on the slope of the surface with respect to line of sight

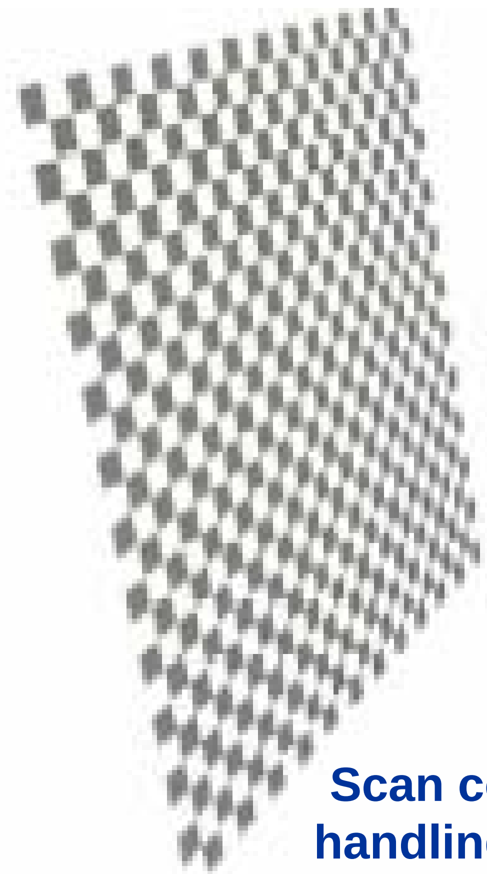
Only evident in animation or for textures with straight lines

Problems with Simple Scan Conversion

- Does this really happen in practice?



Simple scan conversion

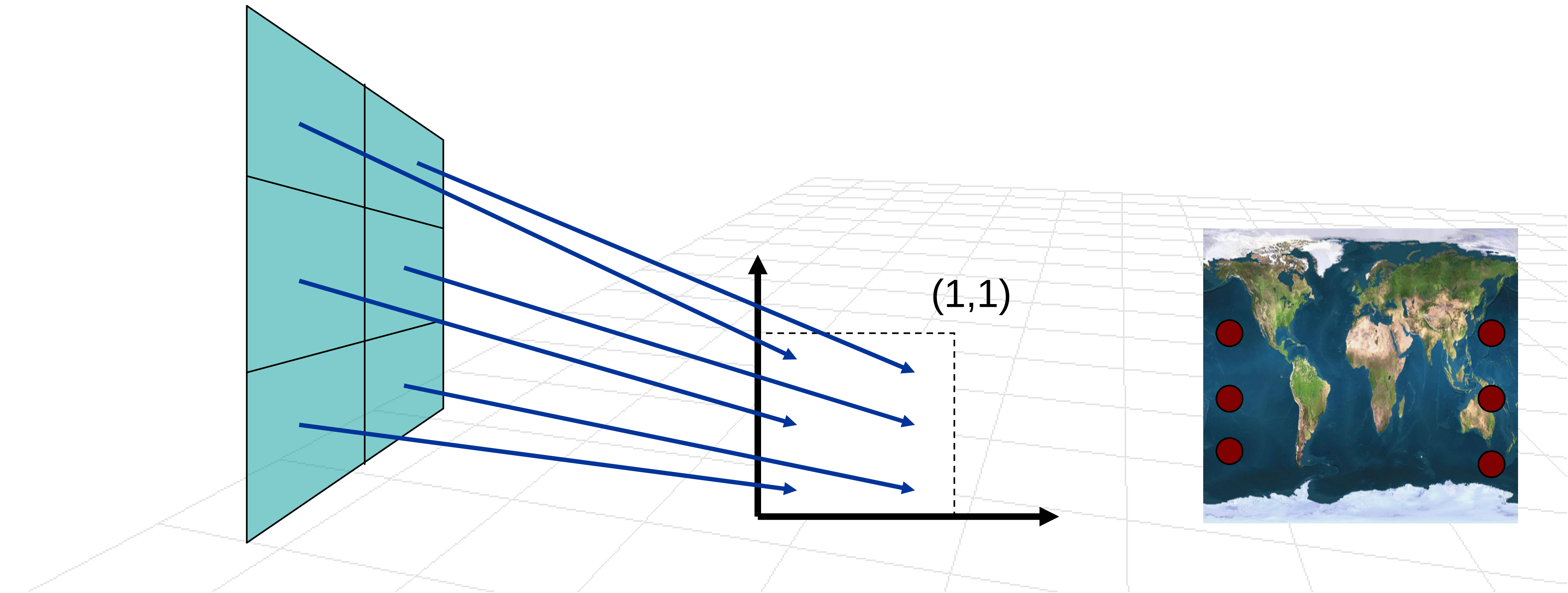


Scan conversion with handling of perspective

- We need to handle perspective effects during scan conversion (more complicated)

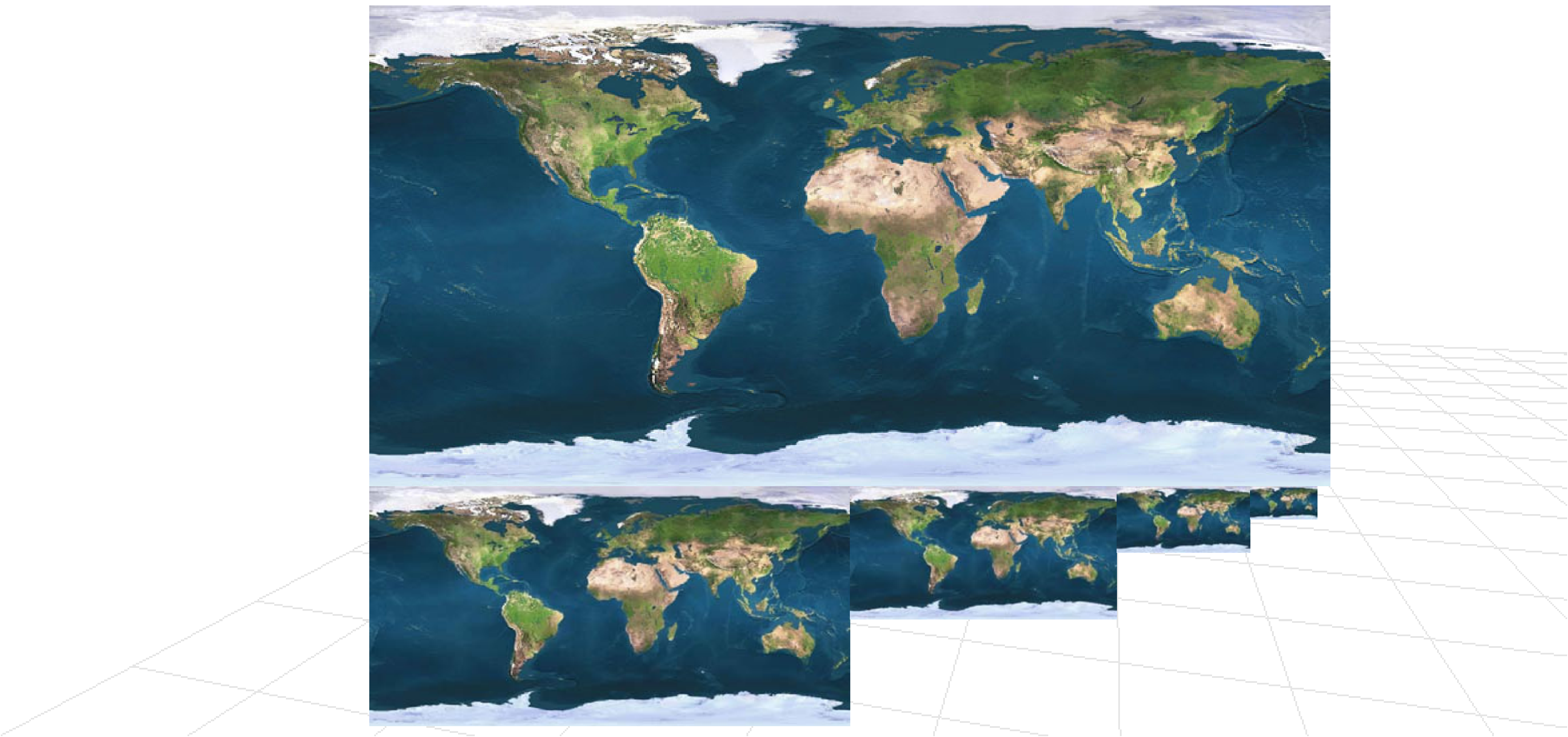
Aliasing

- **Another Problem:** When adjacent pixels in the image plane are rendered, the corresponding coordinates in the texture can be far apart (if the object is far away and we have high resolution texture) and sampling artifacts can be seen.



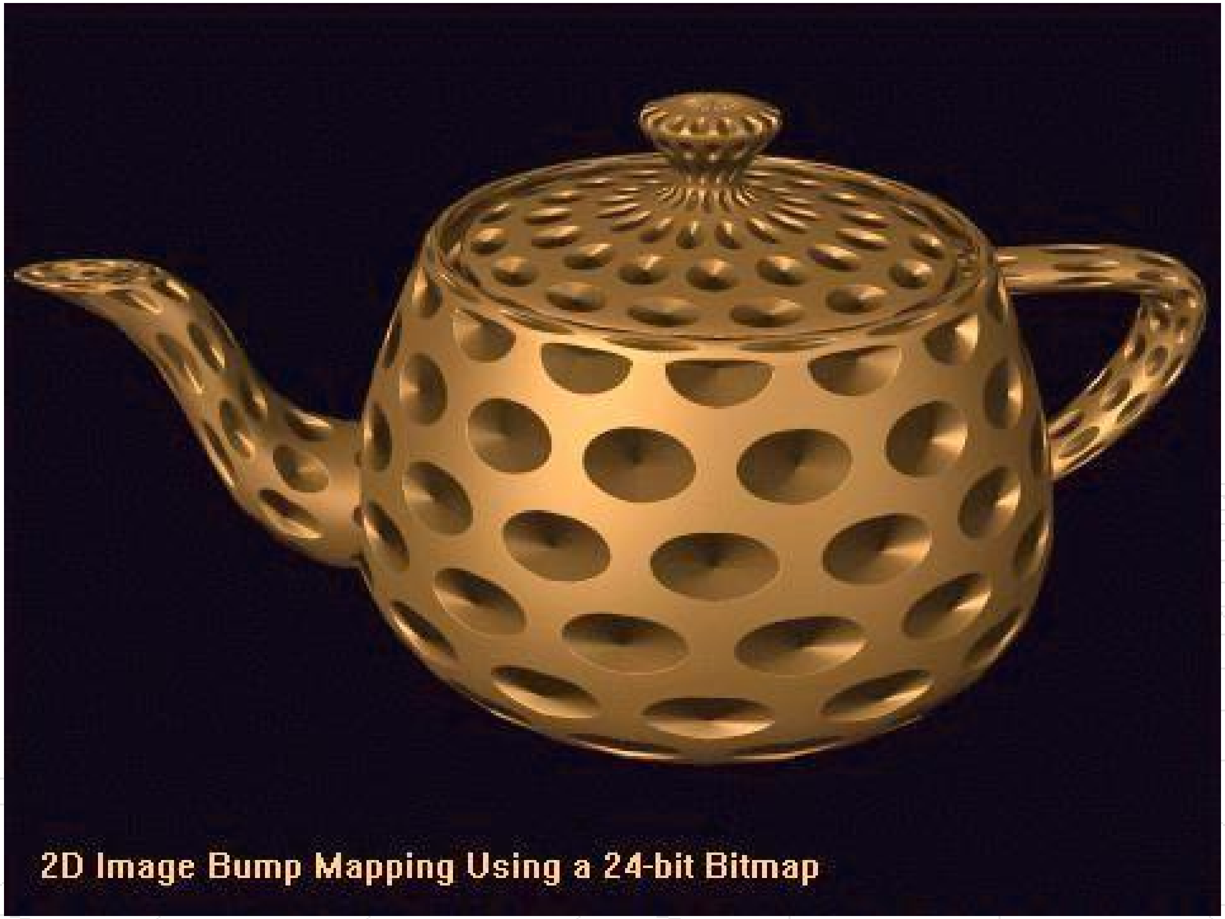
Mipmapping

- **Solution:** use high resolution texture for rendering objects that are close, and low-resolution texture when the object is far away



Bump Mapping

- **Idea:** Instead of perturbing reflectance properties, why don't we perturb the normals? **What's the difference?**



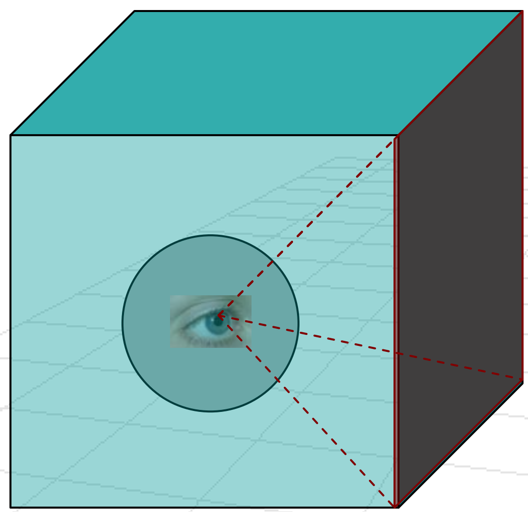
Bump Mapping

Texture Mapping **Bump Mapping**

Environment Mapping

- **Idea:** Use a texture map that corresponds to the view of the environment to achieve the effect of reflectance in a given object.

Let's assume we want to render a silver sphere

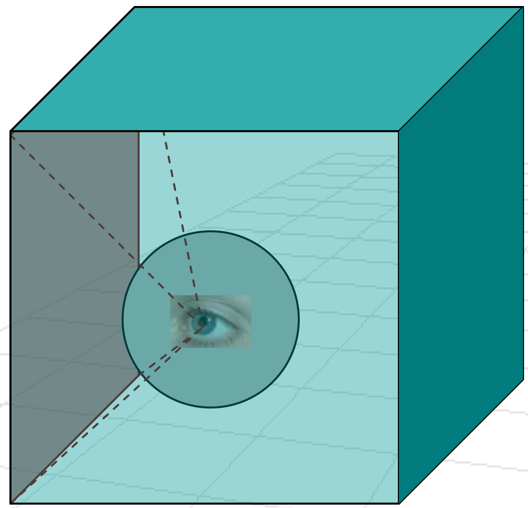


Cube environment mapping

Environment Mapping

- **Idea:** Use a texture map that corresponds to the view of the environment to achieve the effect of reflectance in a given object.

Let’s assume we want to render a silver sphere

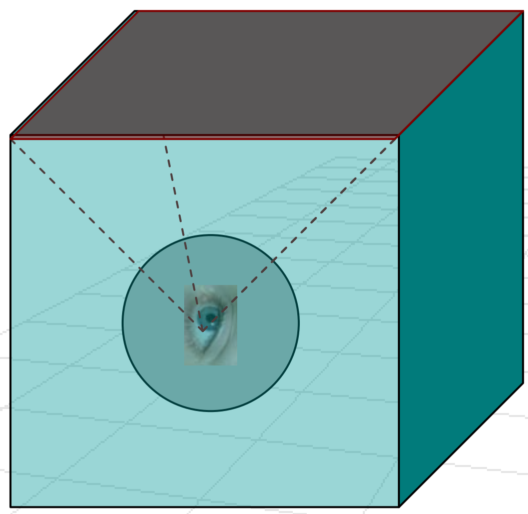


Cube environment mapping

Environment Mapping

- **Idea:** Use a texture map that corresponds to the view of the environment to achieve the effect of reflectance in a given object.

Let's assume we want to render a silver sphere



Cube environment mapping

Environment Mapping

- **Idea:** Use a texture map that corresponds to the view of the environment to achieve the effect of reflectance in a given object.

Image from slides by Aditi Majumder

Environment Mapping

- **Idea:** Use a texture map that corresponds to the view of the environment to achieve the effect of reflectance in a given object.

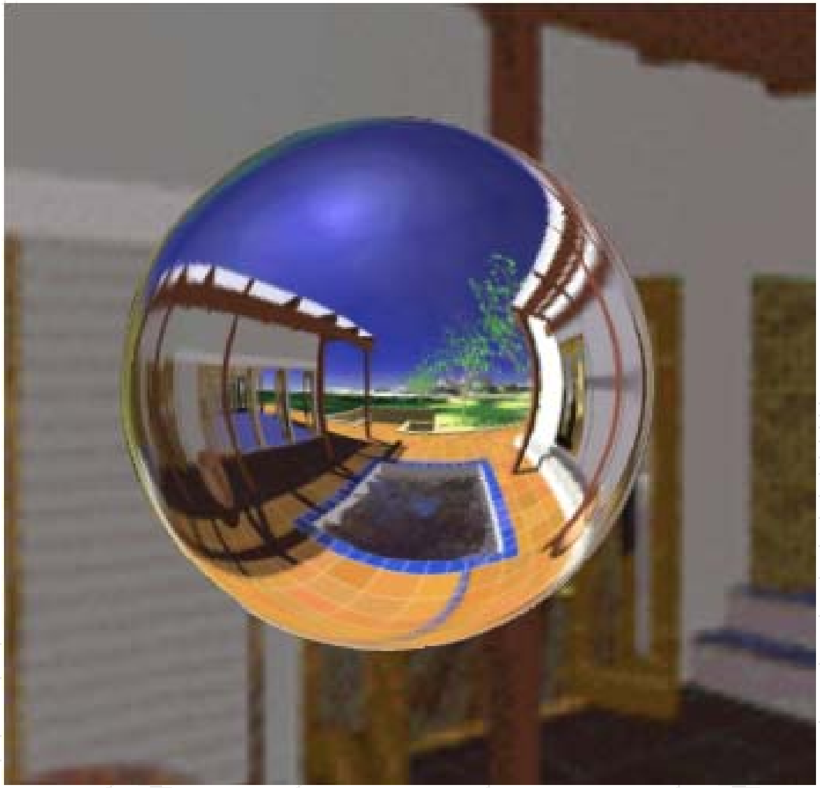


Image from slides by Aditi Majumder