CTP 499
Computer Graphics for CT

Texture Mapping

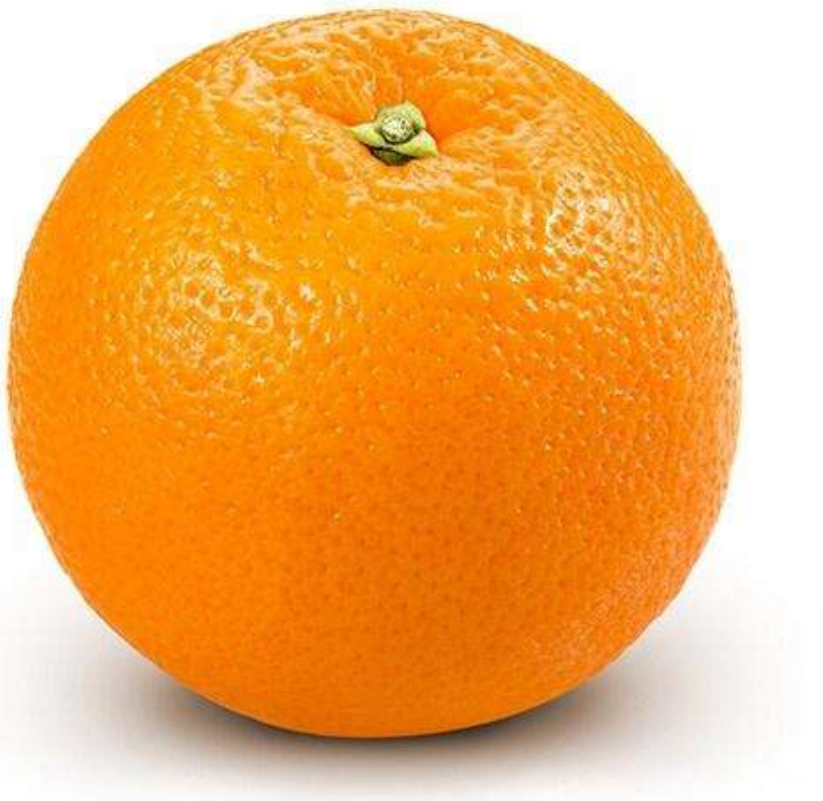
Visual Media Lab

KAIST

November 5, 2025

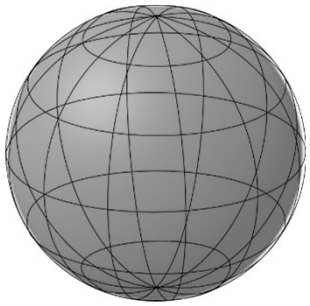
Professor Junyong Noh

Modeling an Orange



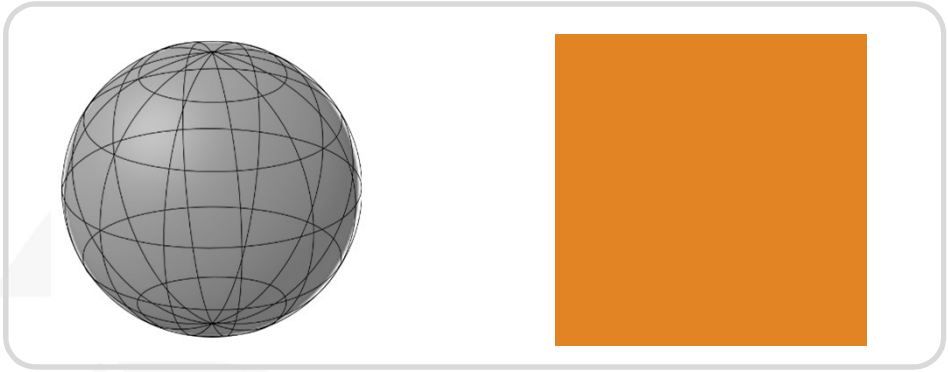
Modeling an Orange

- Geometric orange shape - sphere



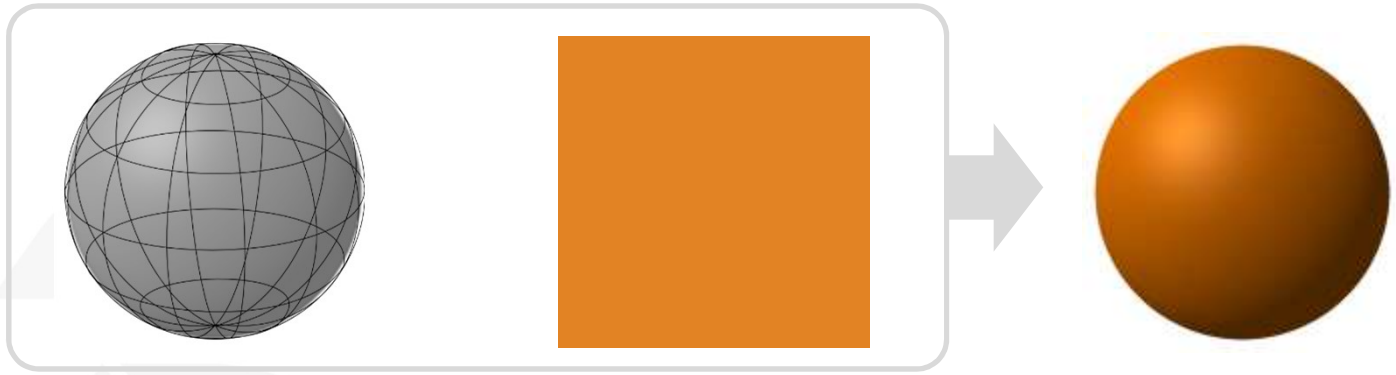
Modeling an Orange

- Orange-colored sphere



Modeling an Orange

- **Orange-colored sphere**
  - Too simple
  - Surface with uniform reflectance
  - Okay for walls or solid balls, but not for most objects



Modeling an Orange

- **Orange-colored sphere**
  - Too simple
  - Surface with uniform reflectance
  - Okay for walls or solid balls, but not most objects
- **Replace sphere with a more complex shape**
  - Takes too many polygons to model all the dimples

Modeling an Orange

- **Texture mapping**
  - Take a picture of a real orange, scan it, and “paste” onto simple geometric model

Example

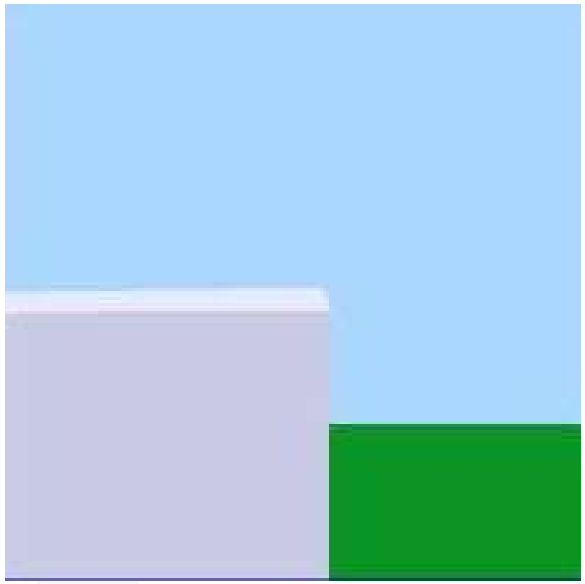

WIBC ABC
PLASTIC COATED
APPROVED
Brunswick
max
Okino
Computer Graphics

Texture Mapping

- Goal: adding visual detail without increasing geometric detail

Texture Mapping

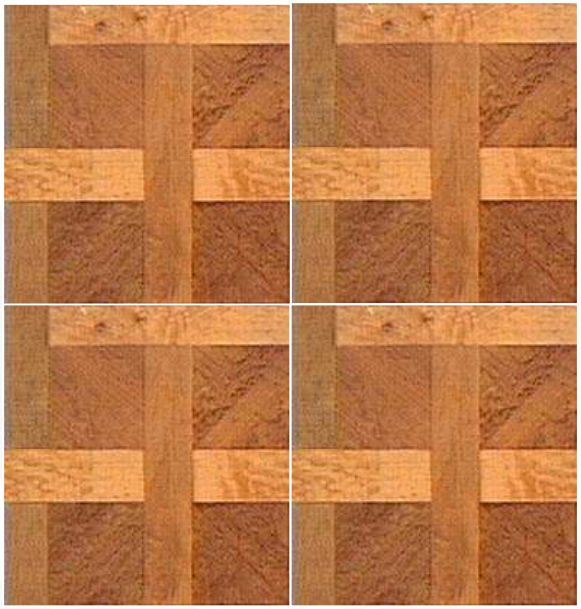
- Goal: adding visual detail without adding geometric detail

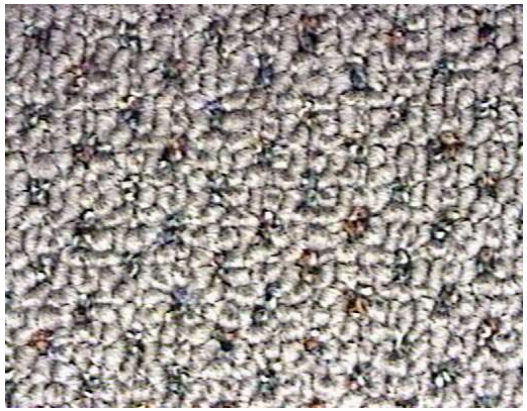


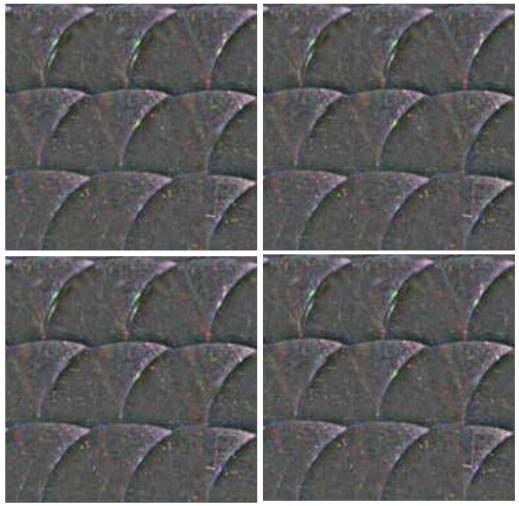
8 polygons

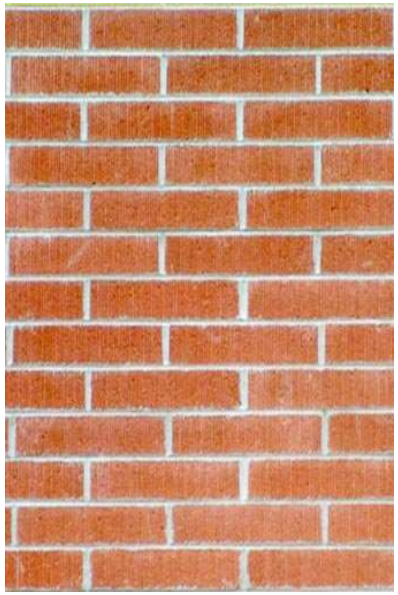
8 polygons

Typical Textures



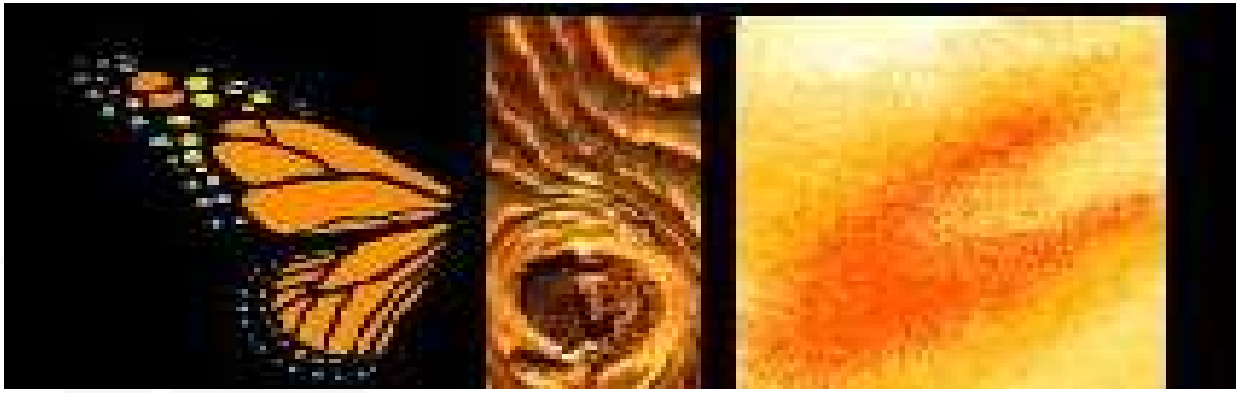






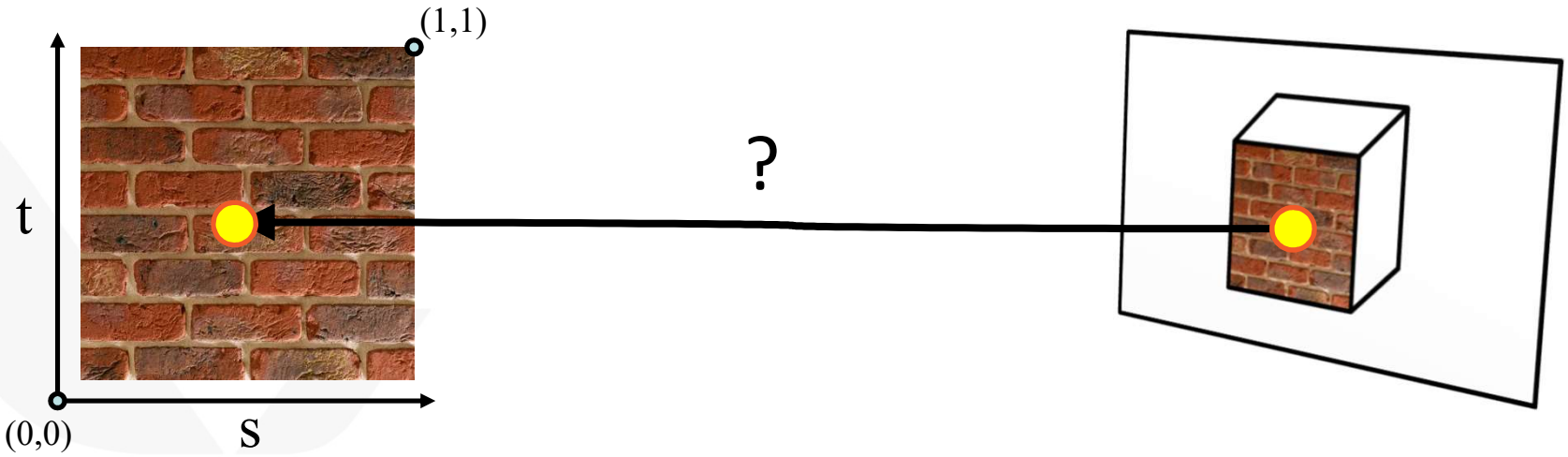
Textures

- Any image can be used as a texture map



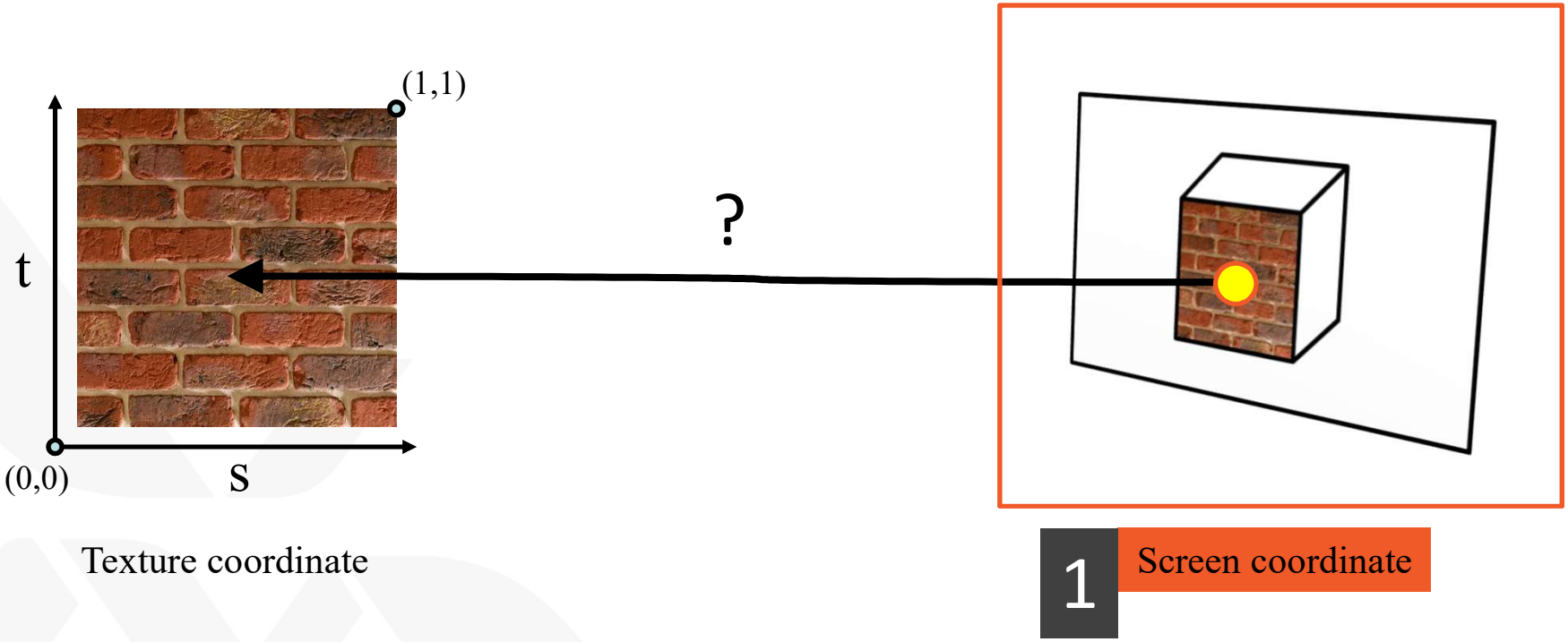
Texture Mapping Process

- **Mapping Pixels to Texels**
  - Basic renderer request: “given pixel (x,y), what is its color?”
    - “given pixel (x,y), what is corresponding texel value?”



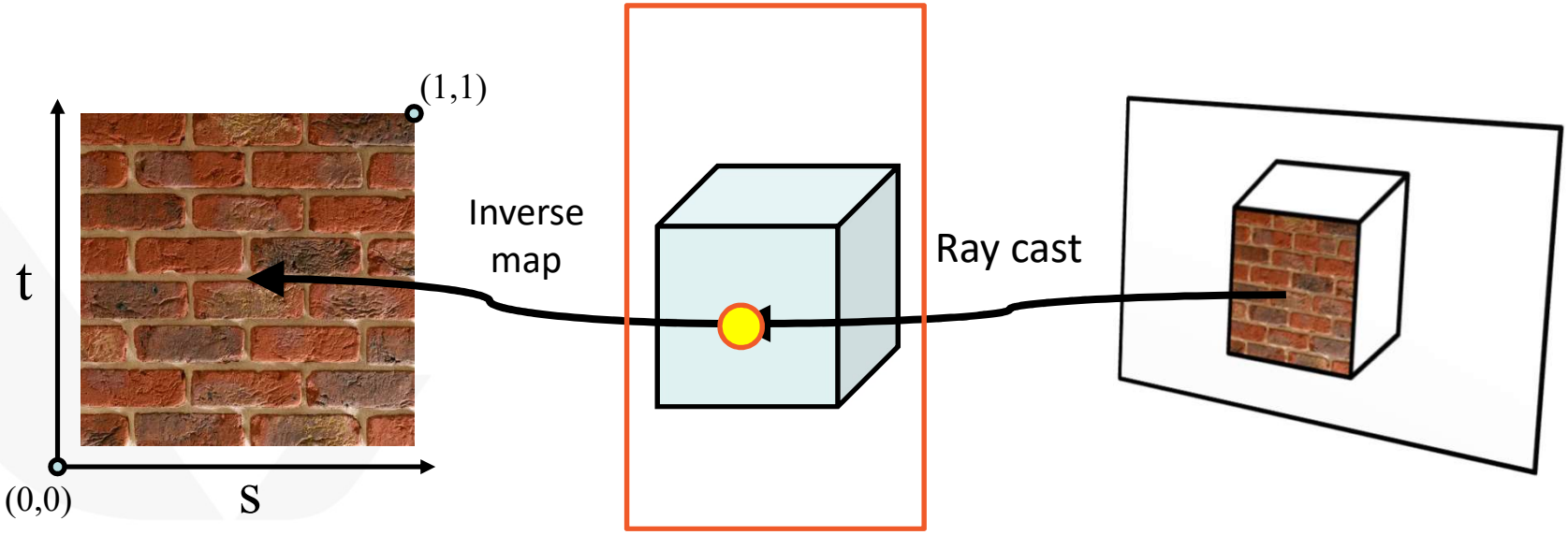
Texture Mapping Process

- **Mapping Pixels to Texels**
  - Basic renderer request: “given pixel (x,y), what is its color?”
    - “given pixel (x,y), what is corresponding texel value?”



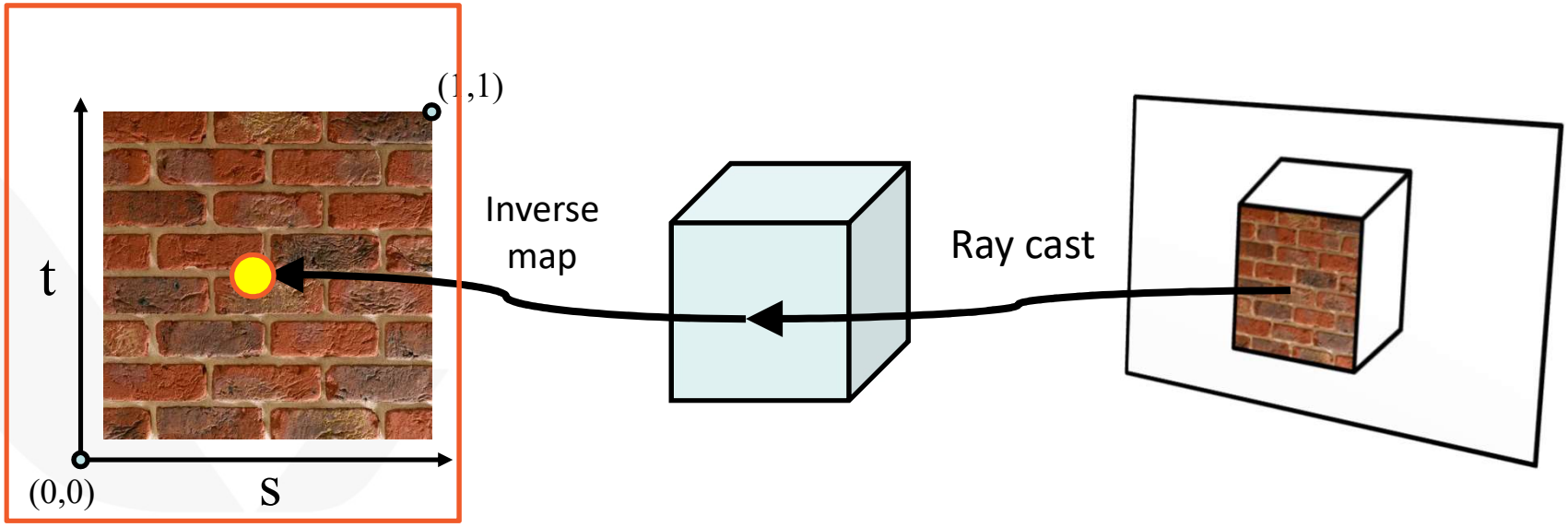
Texture Mapping Process

- **Mapping Pixels to Texels**
  - Basic renderer request: “given pixel (x,y), what is its color?”
    - “given pixel (x,y), what is corresponding texel value?”



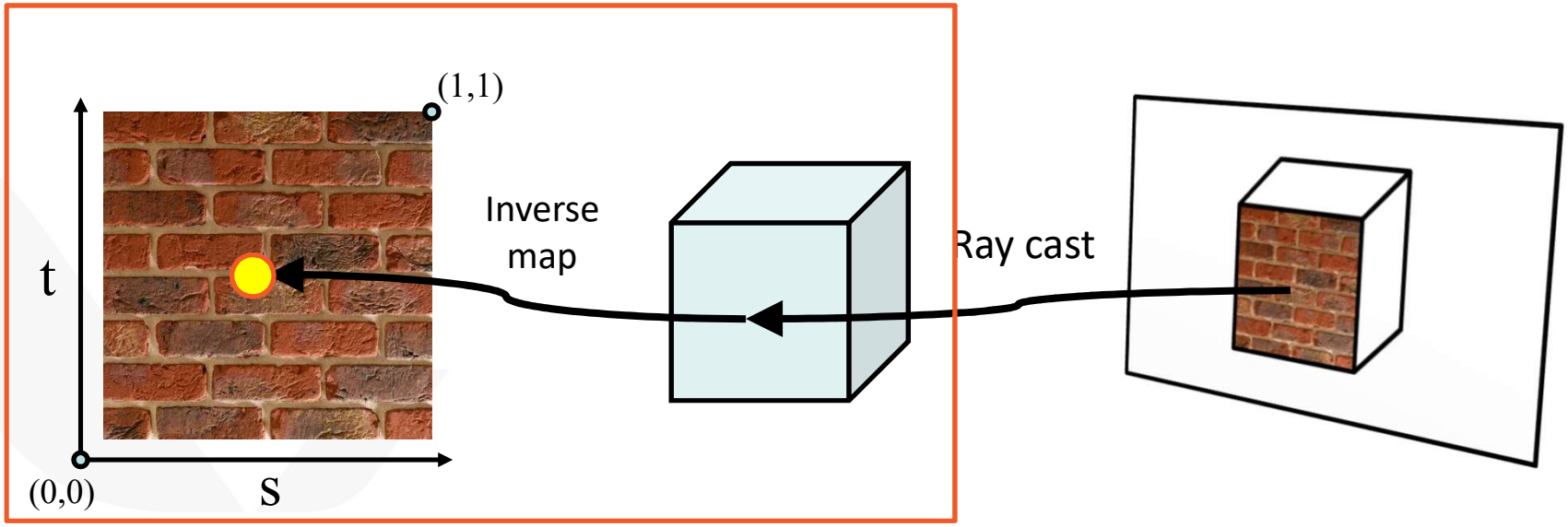
Texture Mapping Process

- **Mapping Pixels to Texels**
  - Basic renderer request: “given pixel (x,y), what is its color?”
    - “given pixel (x,y), what is corresponding texel value?”



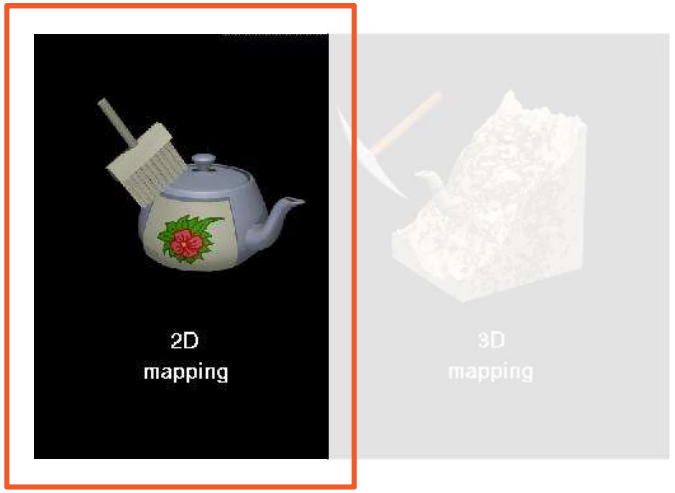
Texture Mapping Process

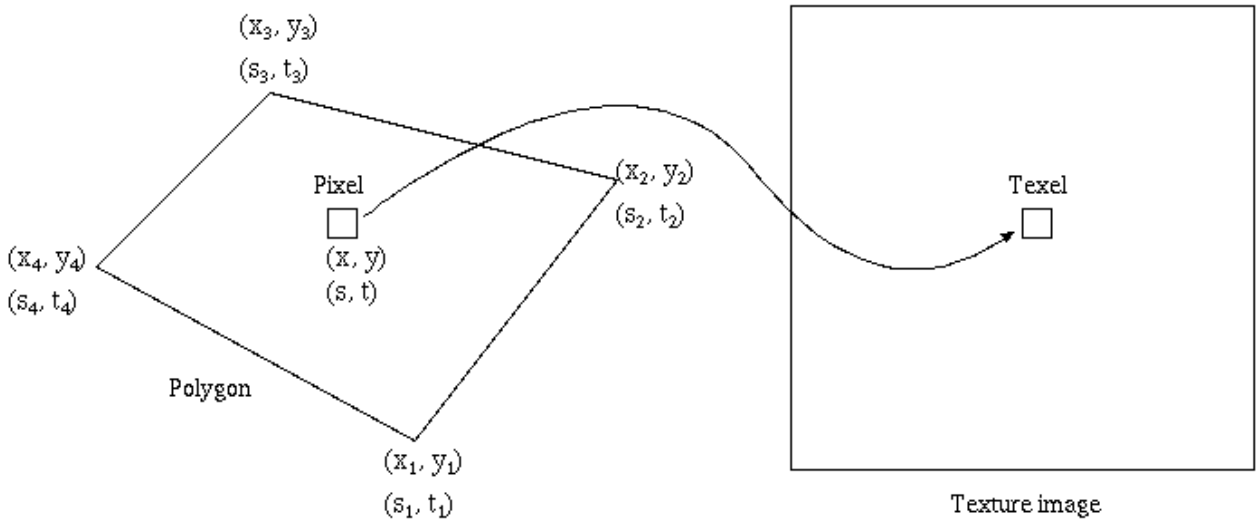
- **Mapping Pixels to Texels**
  - Basic renderer request: “given pixel (x,y), what is its color?”
    - “given pixel (x,y), what is corresponding texel value?”



Texture Mapping

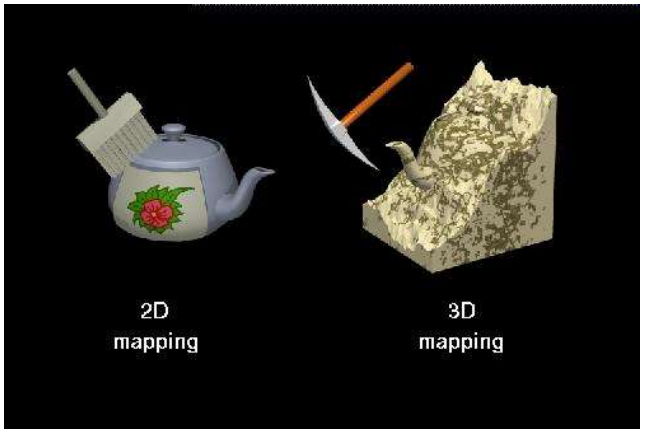
- **2D Texture**
  - ‘Wallpaper’ 2D image onto an object

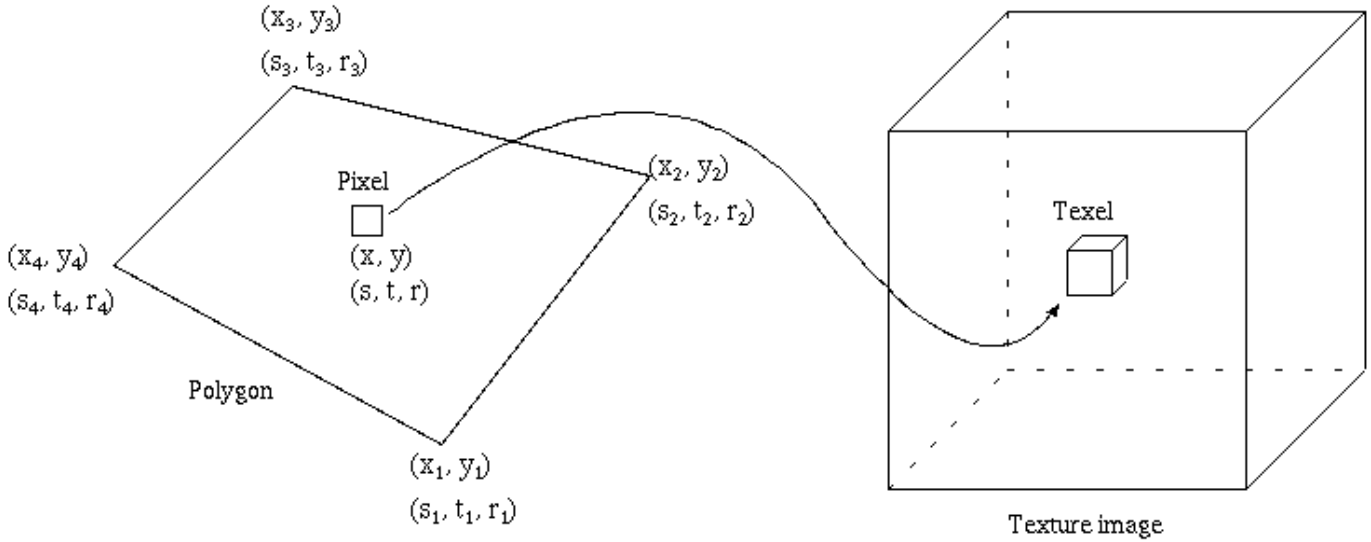




Texture Mapping

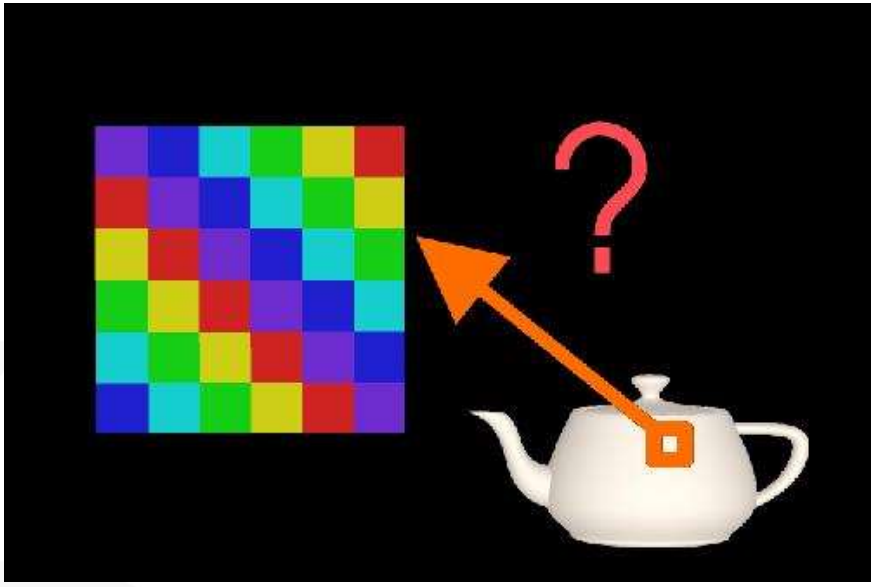
- **2D Texture**
  - ‘Wallpaper’ 2D image onto an object
- **3D Texture**
  - ‘Carve’ 3D object out of a block





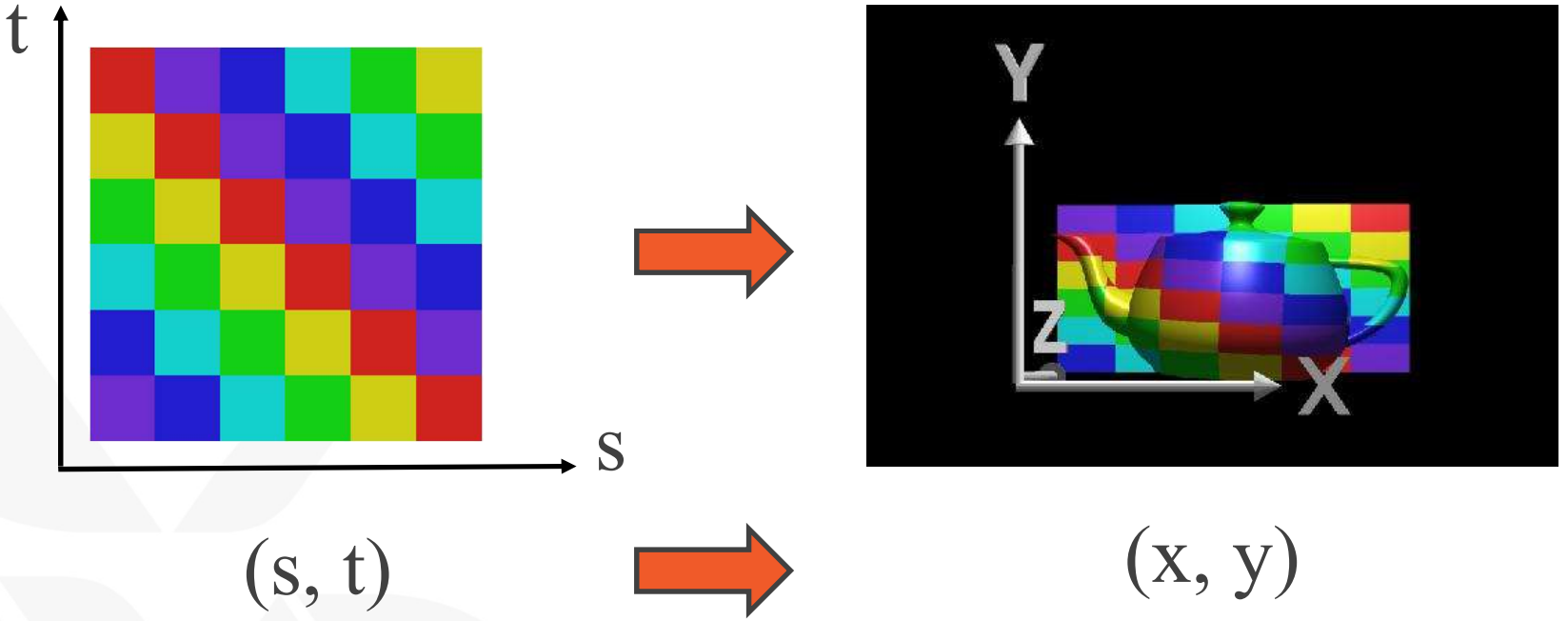
2D Texture Mapping

- **Given a 2D texture (image) and a 3D object, map the texture onto the object**
  - Where does each point on the object map into the texture?



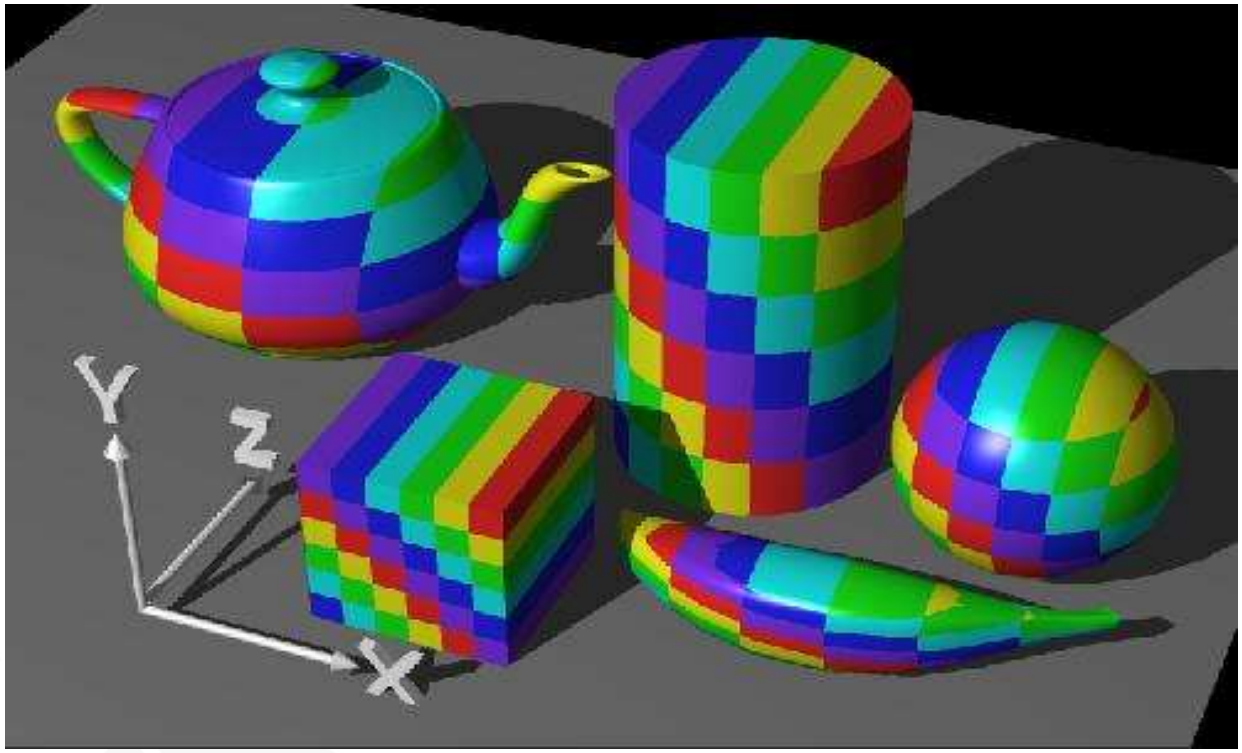
Texture Map Shapes

- **Planar Map**
  - Simply remove one of the object's coordinates (project onto that coordinate plane)



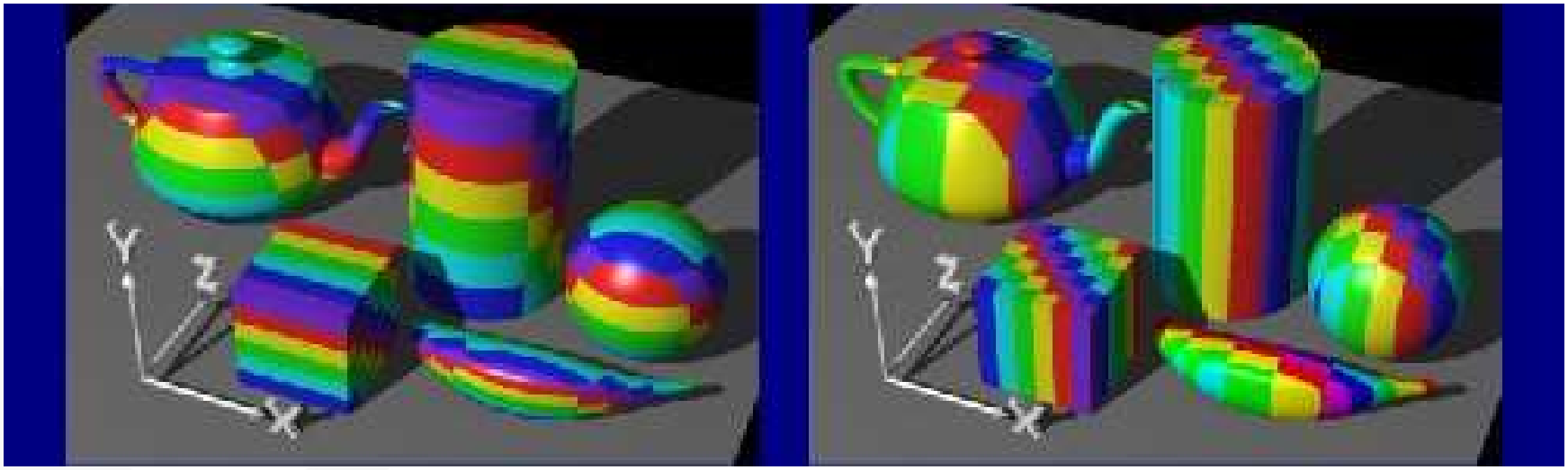
Planar Map

- **The texture is constant in one direction.**
  - Often not what you want



Planar Map

- Similarly, using other directions:



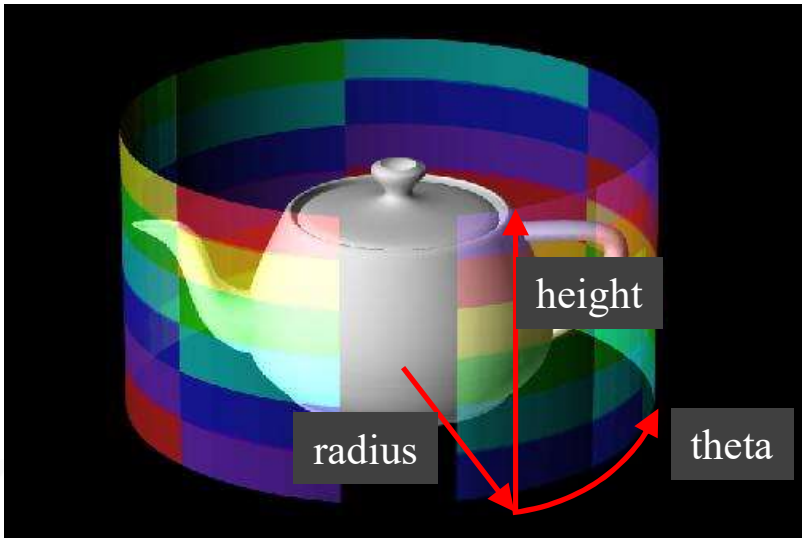
X axis removed

Y axis removed

Cylindrical Map

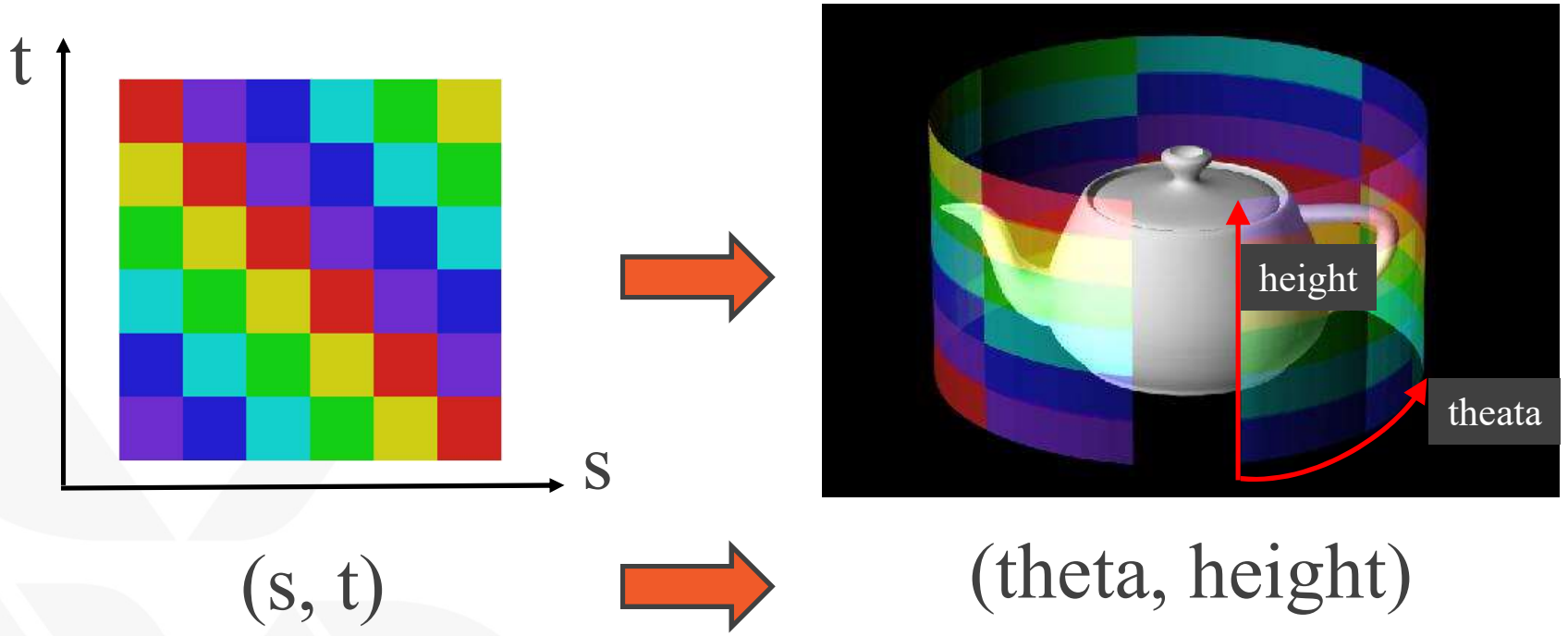
Cylindrical Map

- (x,y,z) is converted to (r, theta, height).



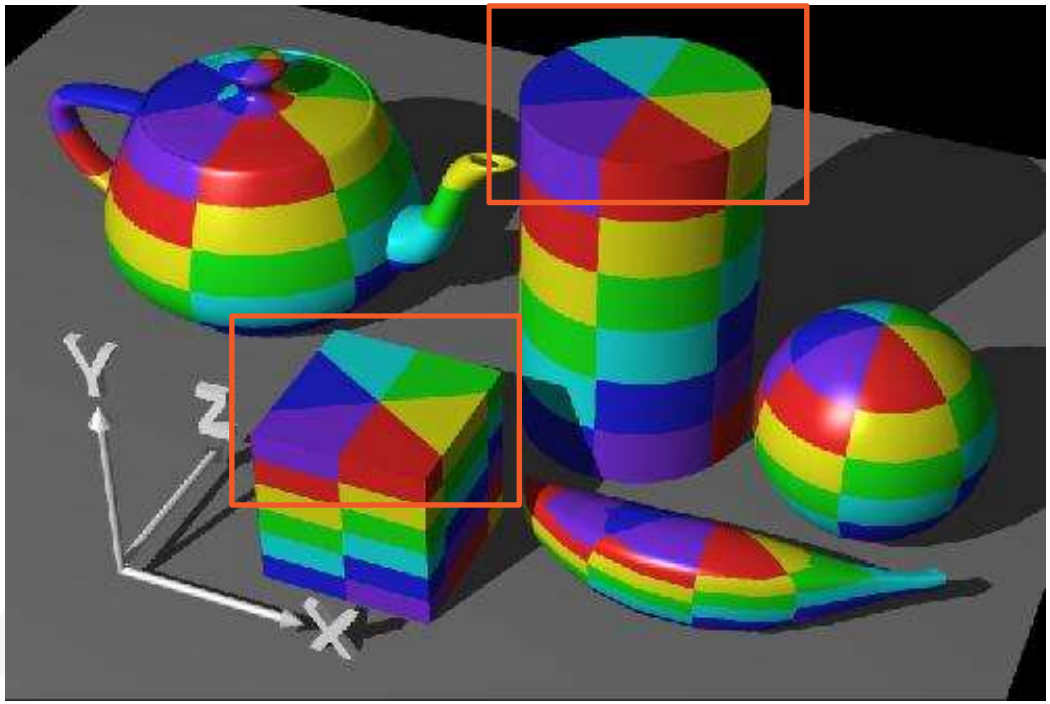
Cylindrical Map

- (x,y,z) is converted to (r, theta, height).
  - For texture mapping, theta is converted into a s-coordinate and height is converted into a t-coordinate.



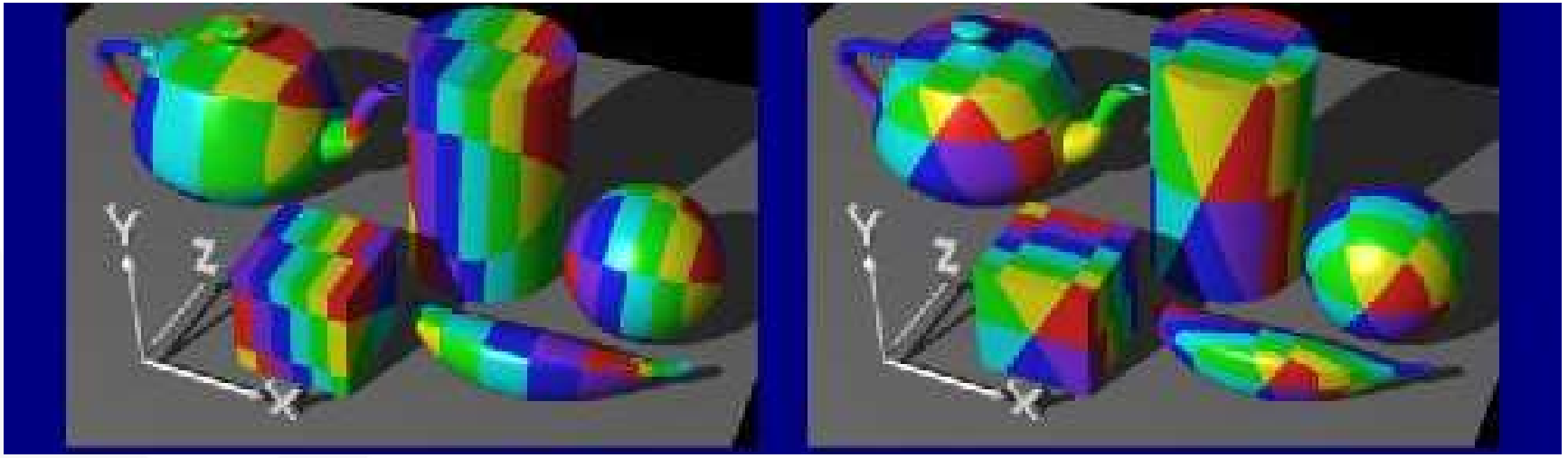
Cylindrical Map

- At minimum and maximum extents of the cylinder, the texture is pinched together.



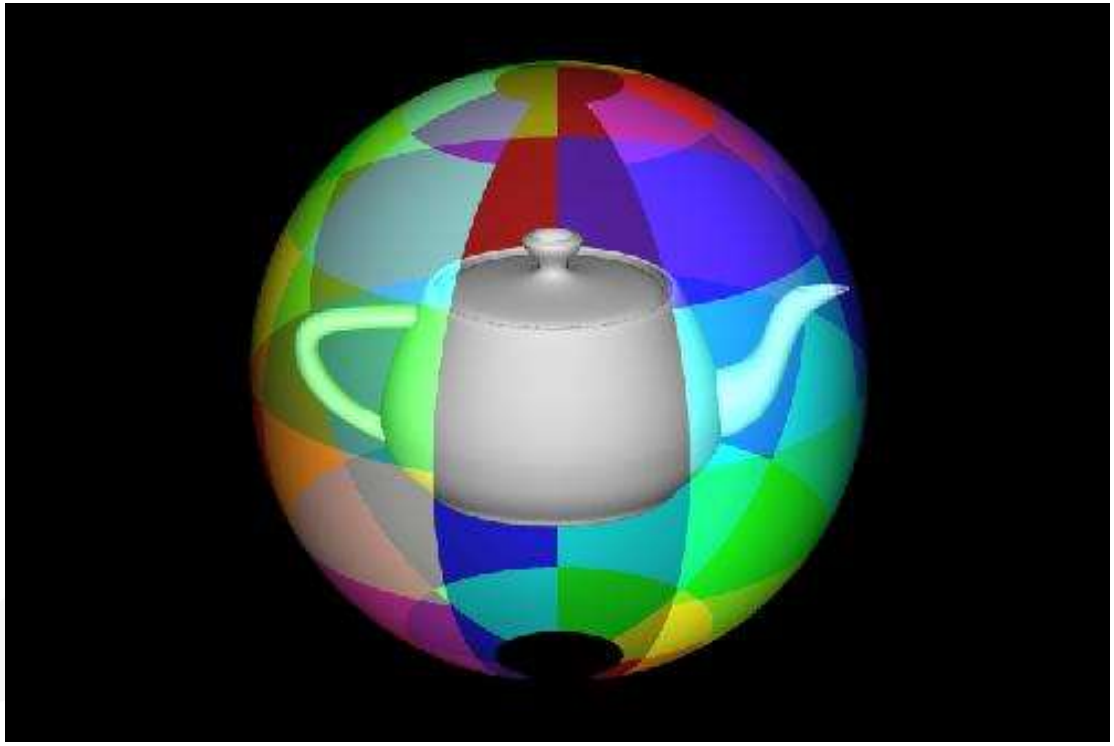
Cylindrical Map

- Similarly, using other directions:



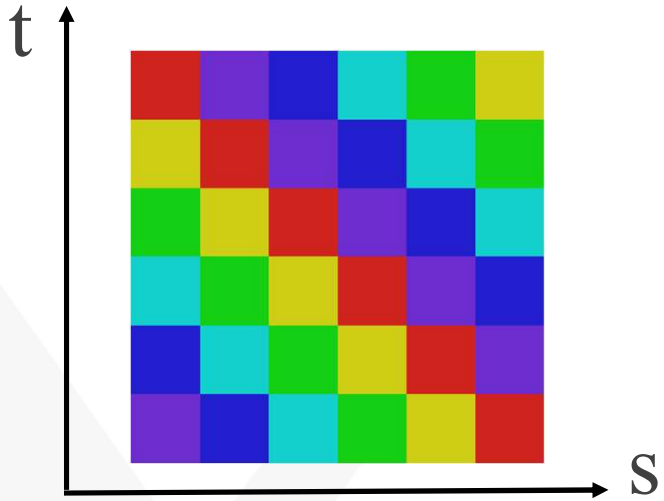
Spherical Map

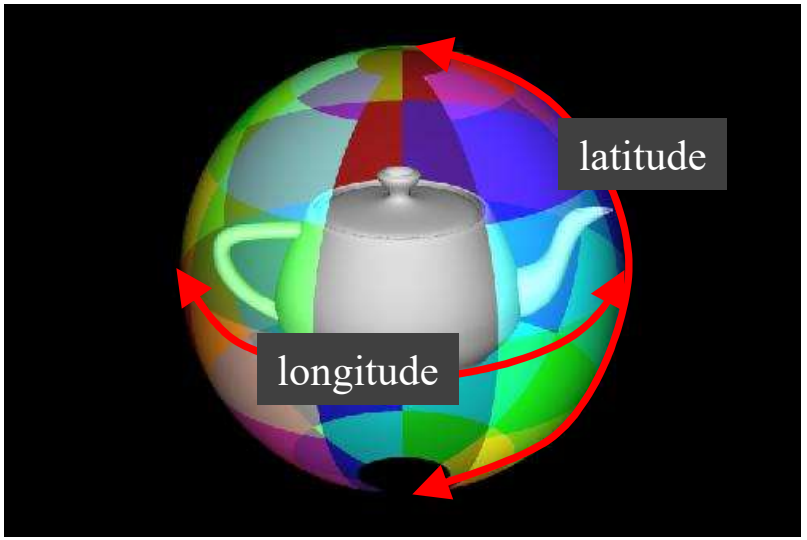
- Convert from (x, y, z) to spherical coordinates.



Spherical Map

- **Convert from (x, y, z) to spherical coordinates.**
  - Longitude is converted to a s-coordinate, latitude is converted to a t-coordinate.



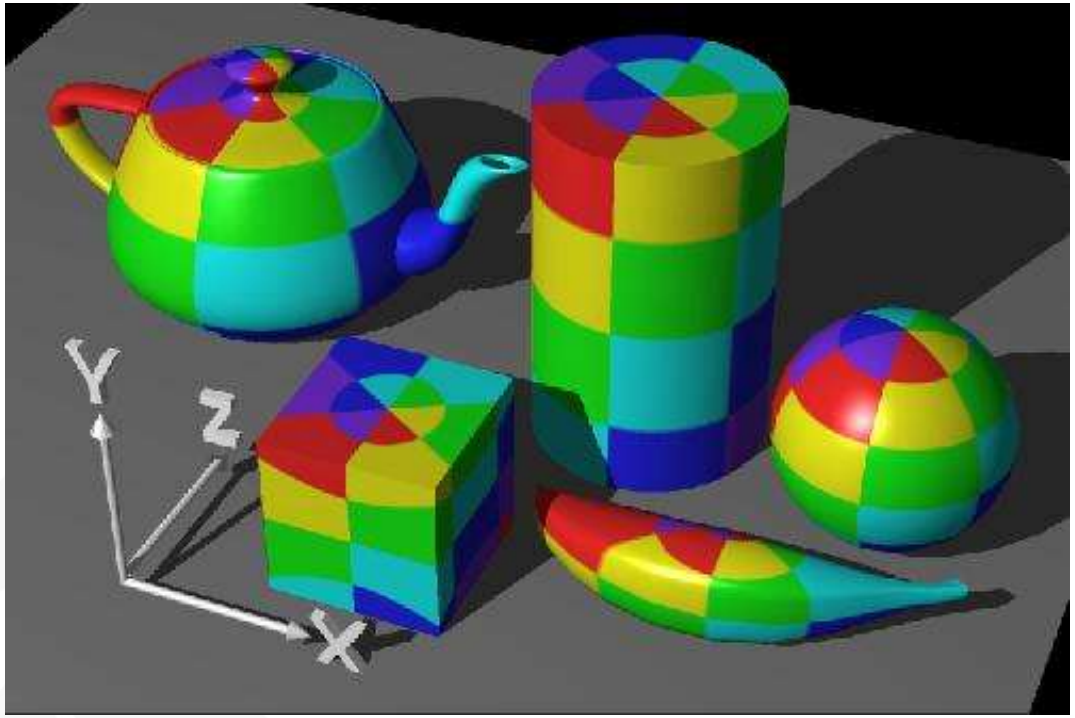


(s, t)

(longitude, latitude)

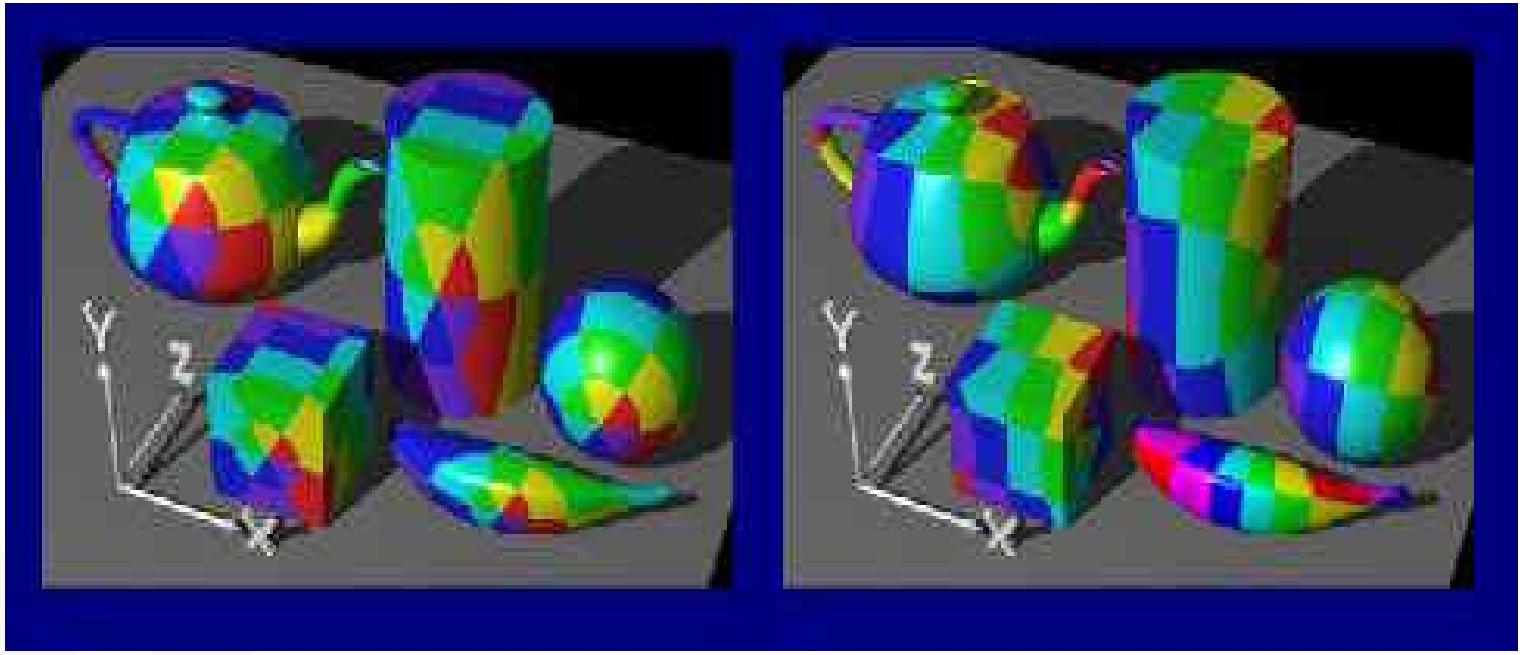
Spherical Map

- This still pinches the texture at the poles, but it's different from using a cylindrical map.



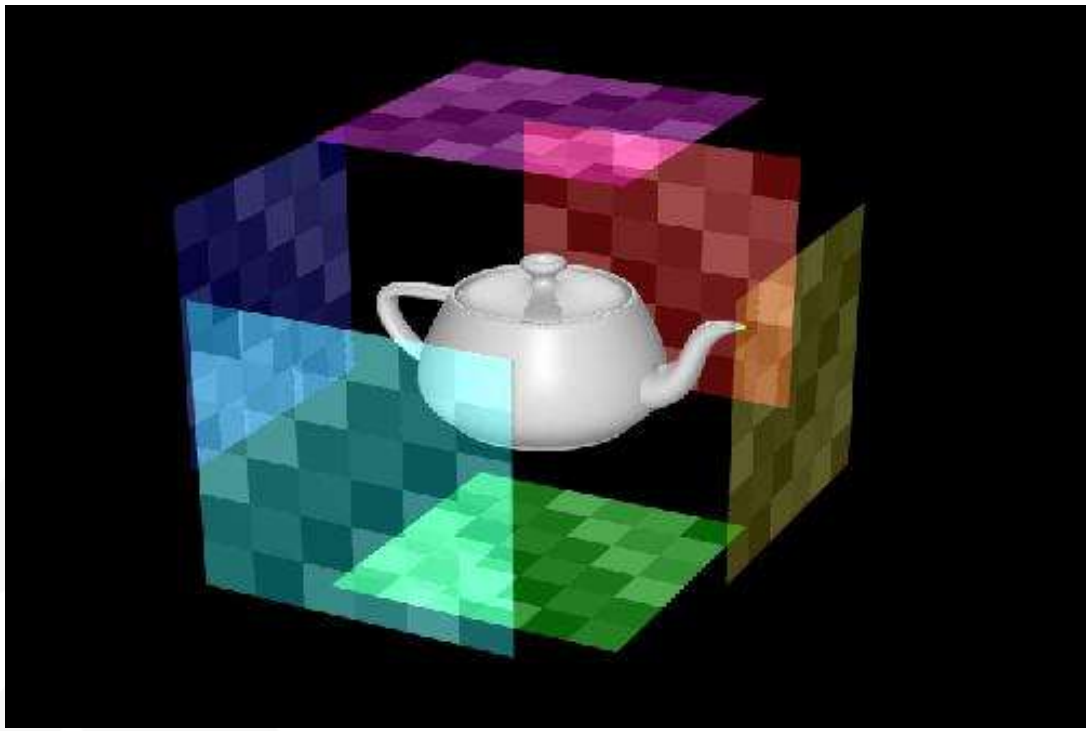
Spherical Map

- Similarly, using other directions:



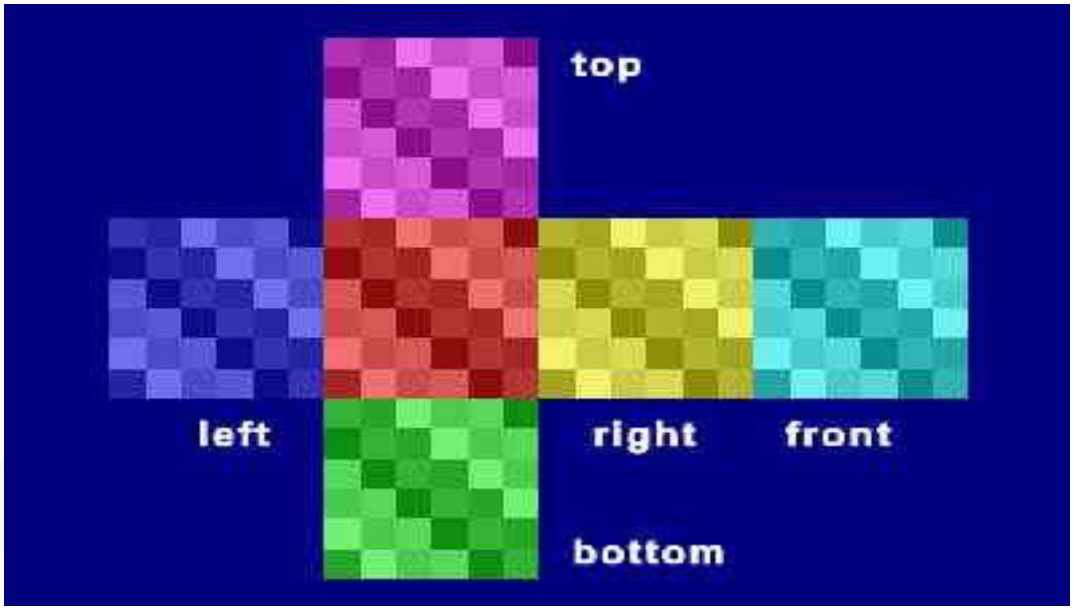
Box Map

- Use of a collection of planar maps to provide better coverage than using a single planar map:

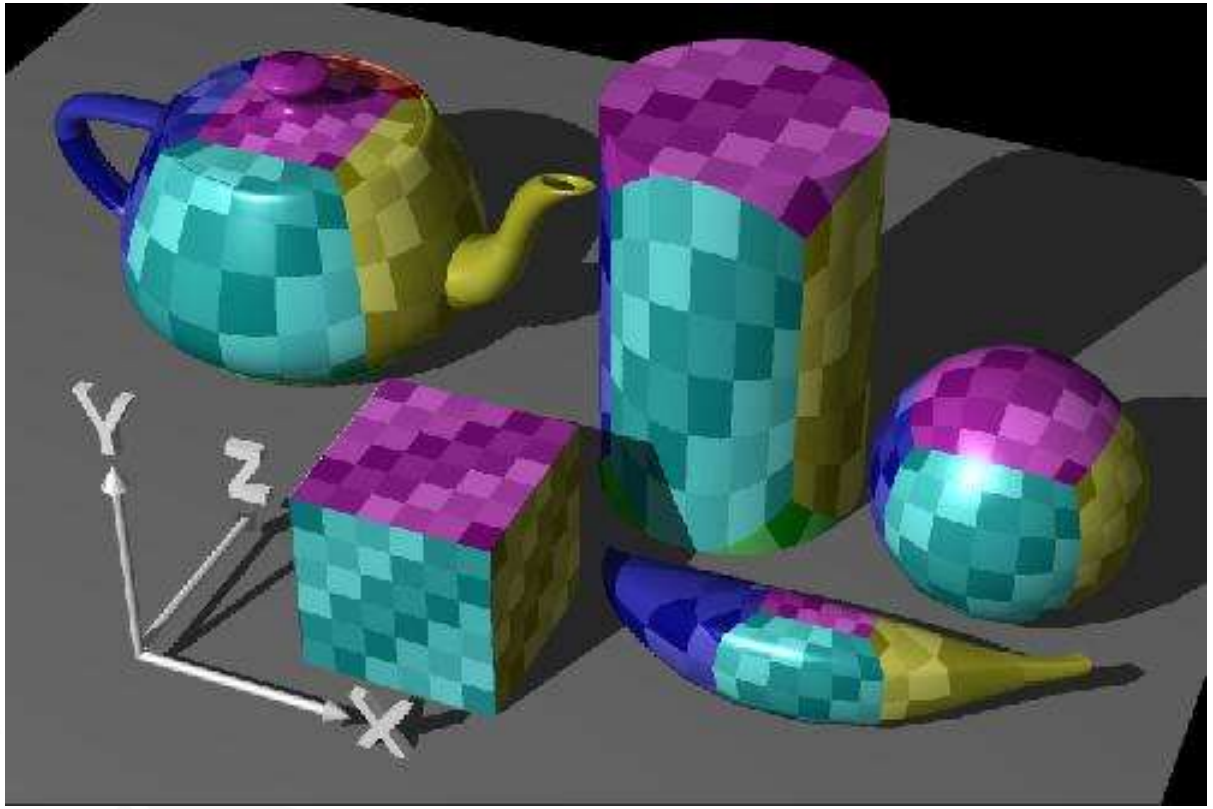


Box Map

- Projecting each plane onto its portion of the object
- Use of the object's normal to determine which texture to use

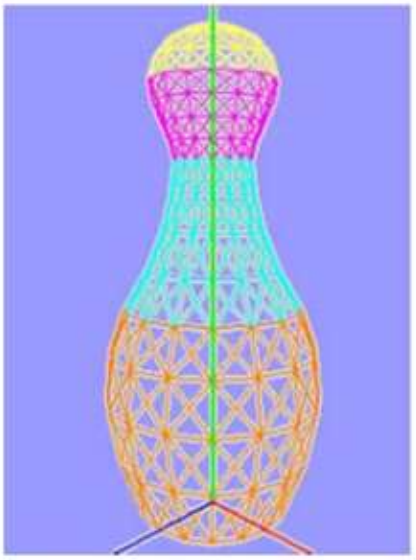


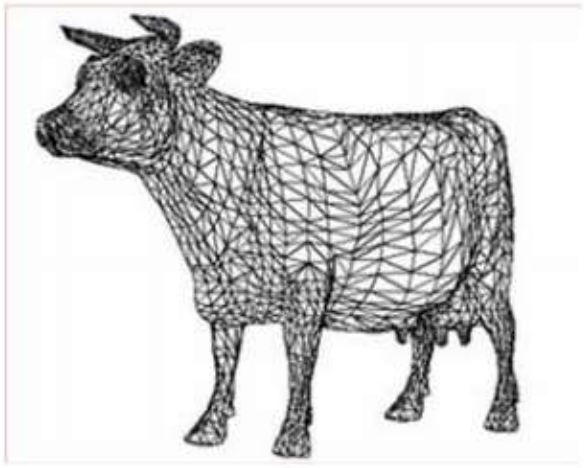
Box Map



Texturing Polygon Meshes

- How to assign texture coordinates to things like?



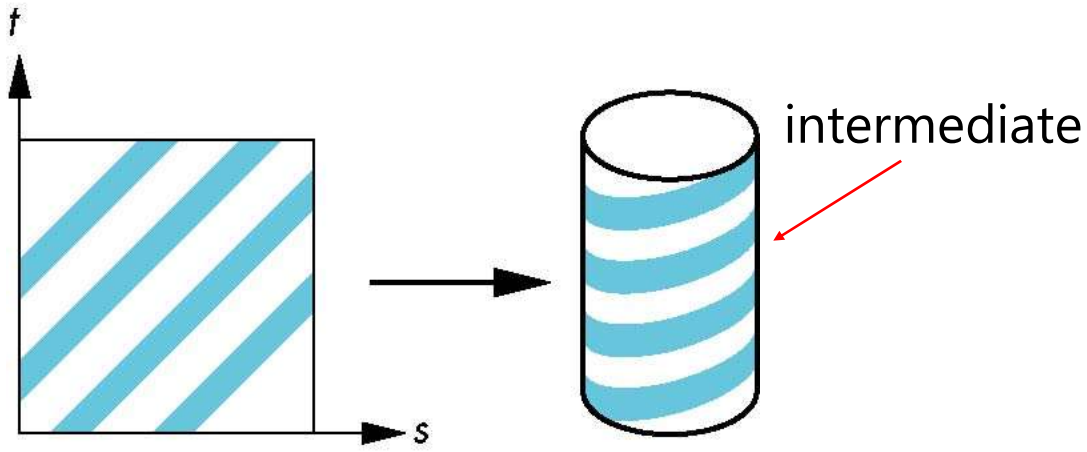


Two-part Mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface

Two-part Mapping

- **One solution to the mapping problem is to first map the texture to a simple intermediate surface**
  - Cylinders and sphere are most common
    - (planes or cubes also used)

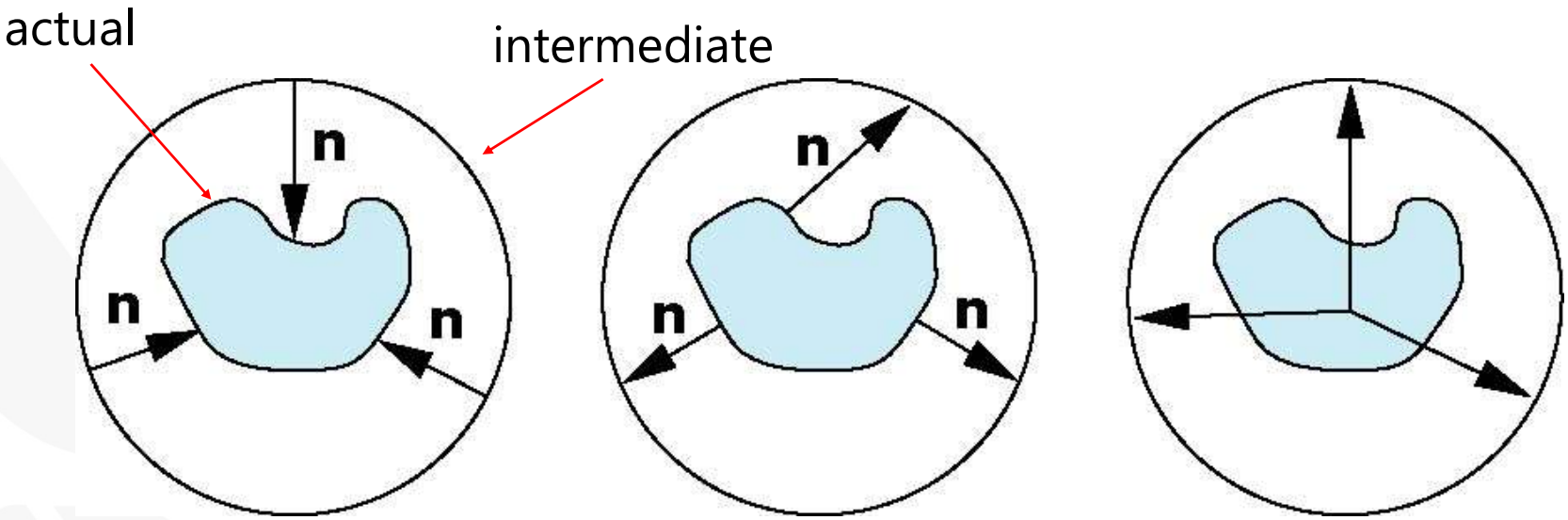


Two-part Mapping

- Map from intermediate object to actual object

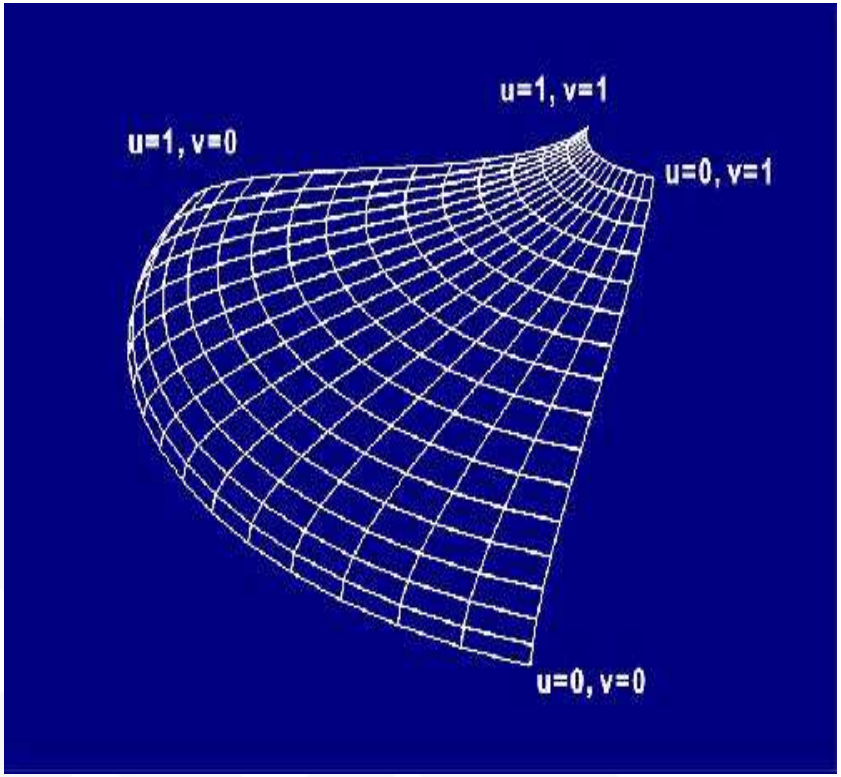
Two-part Mapping

- **Map from intermediate object to actual object**
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate



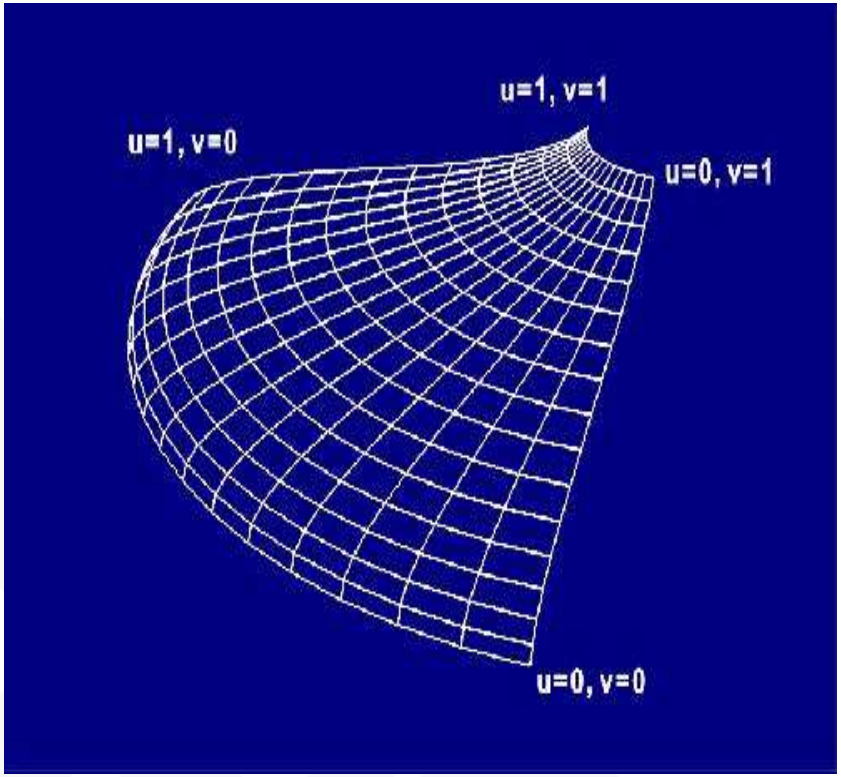
Mapping Parametric Surfaces

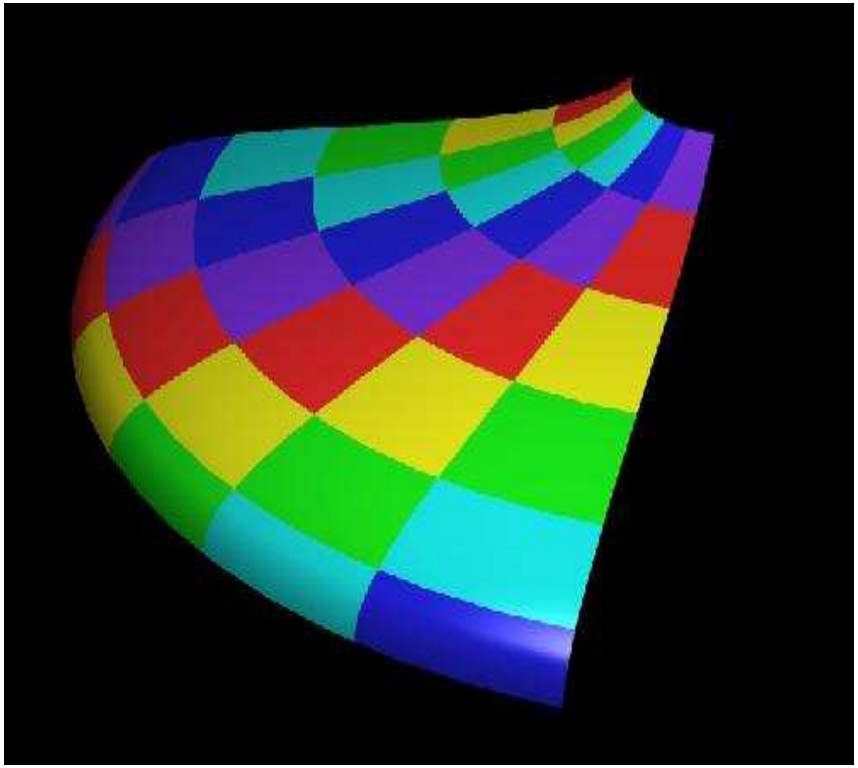
- Parametric surfaces are already parameterized by (u, v).
- Use the (u,v) parameters as the (s,t) texture parameters



Mapping Parametric Surfaces

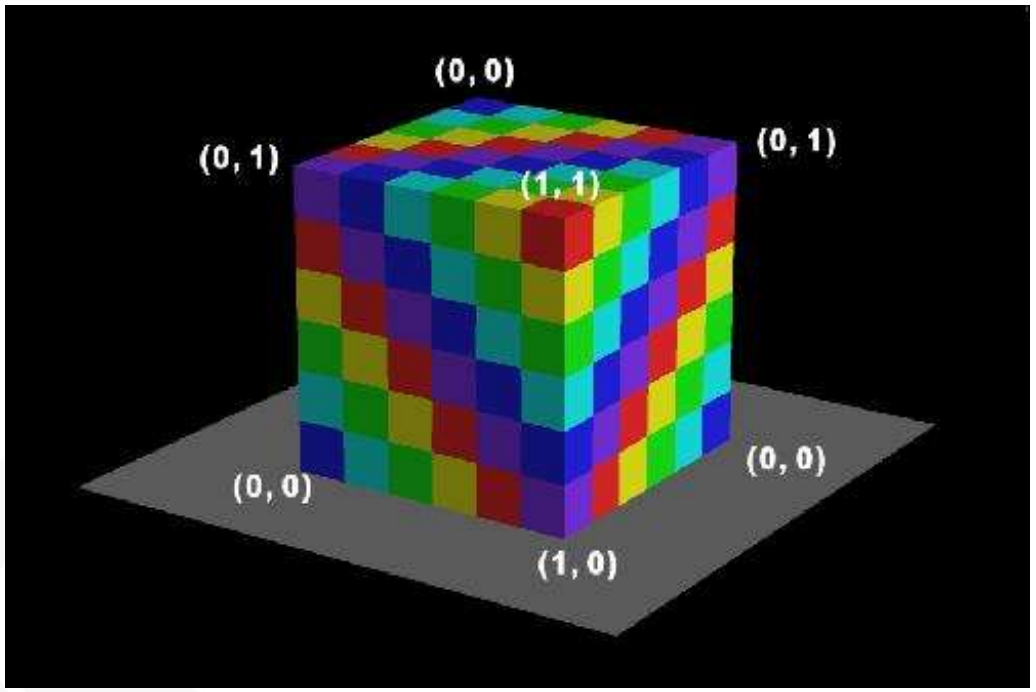
- Parametric surfaces are already parameterized by (u, v).
- Use the (u,v) parameters as the (s,t) texture parameters





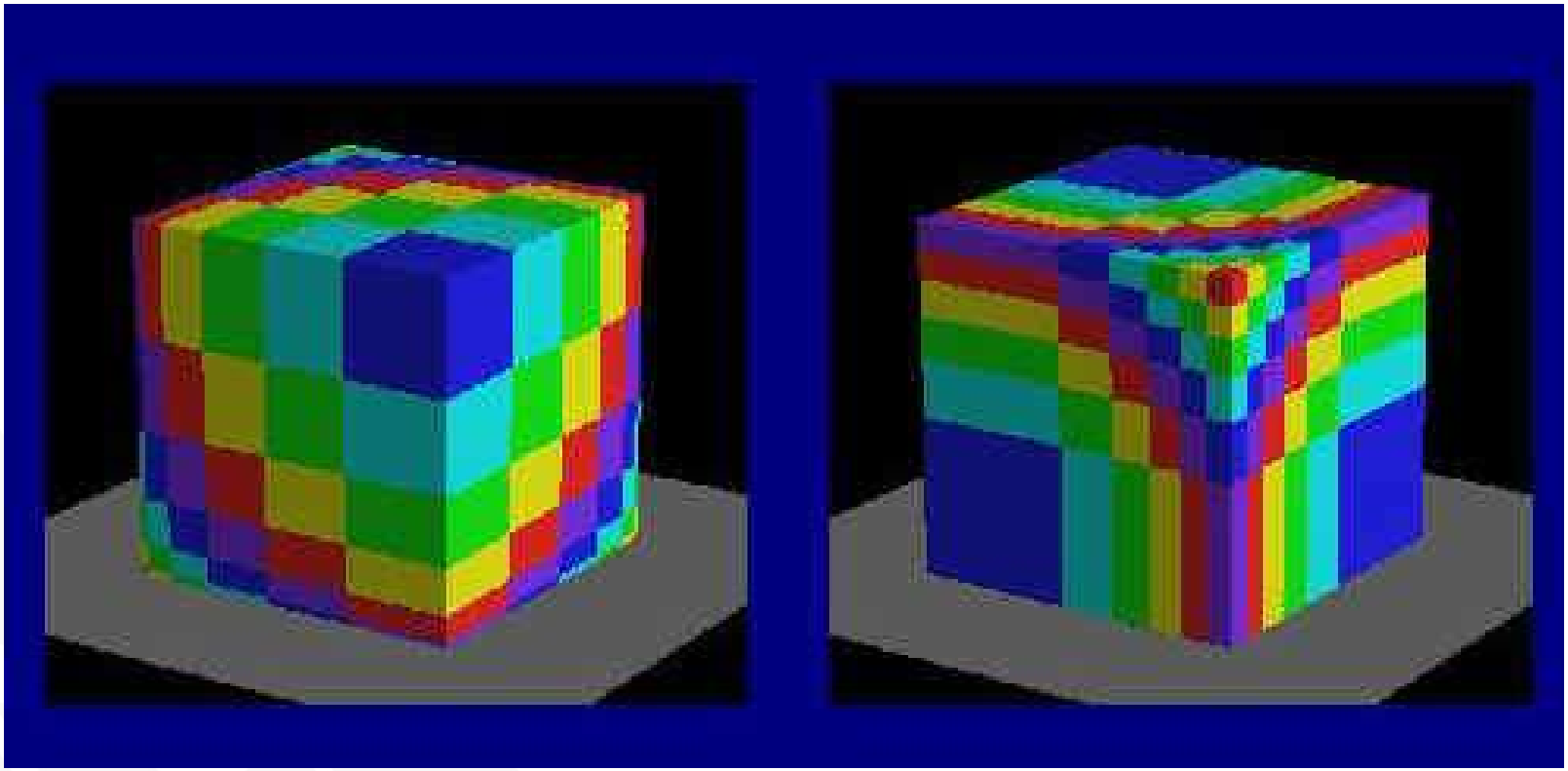
Non-Parametric Surfaces

- If we assign values to the vertices in the range (0,1), we can use the same texture mapping approach.

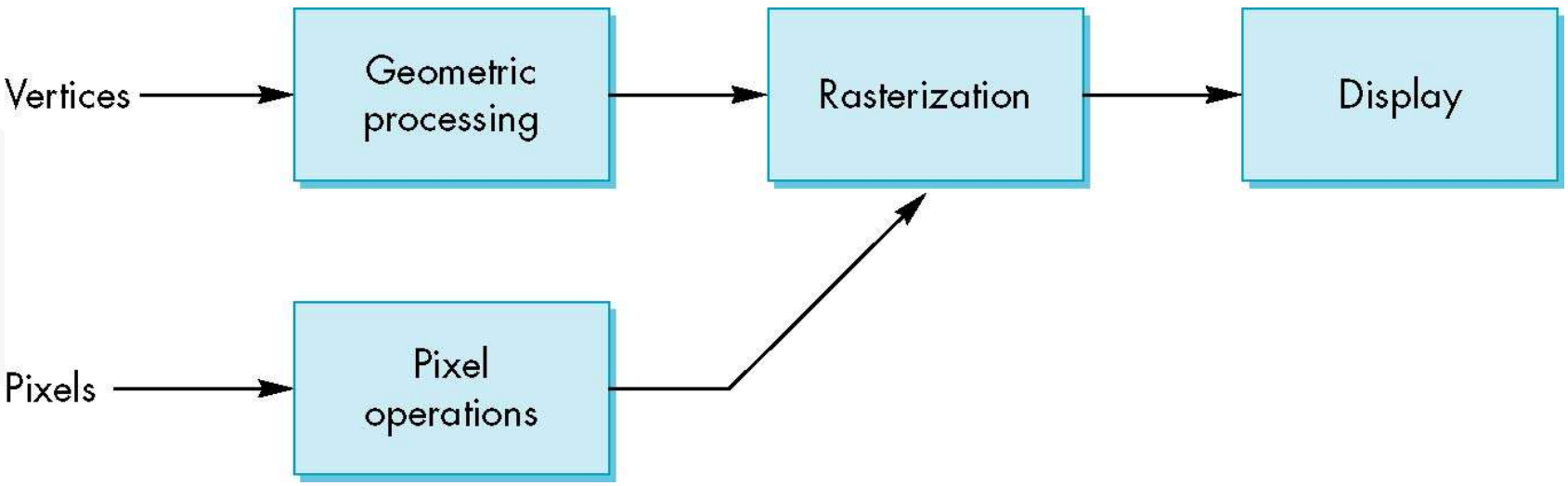


Non-linear Mapping

- We can distort the texture using a non-linear mapping:

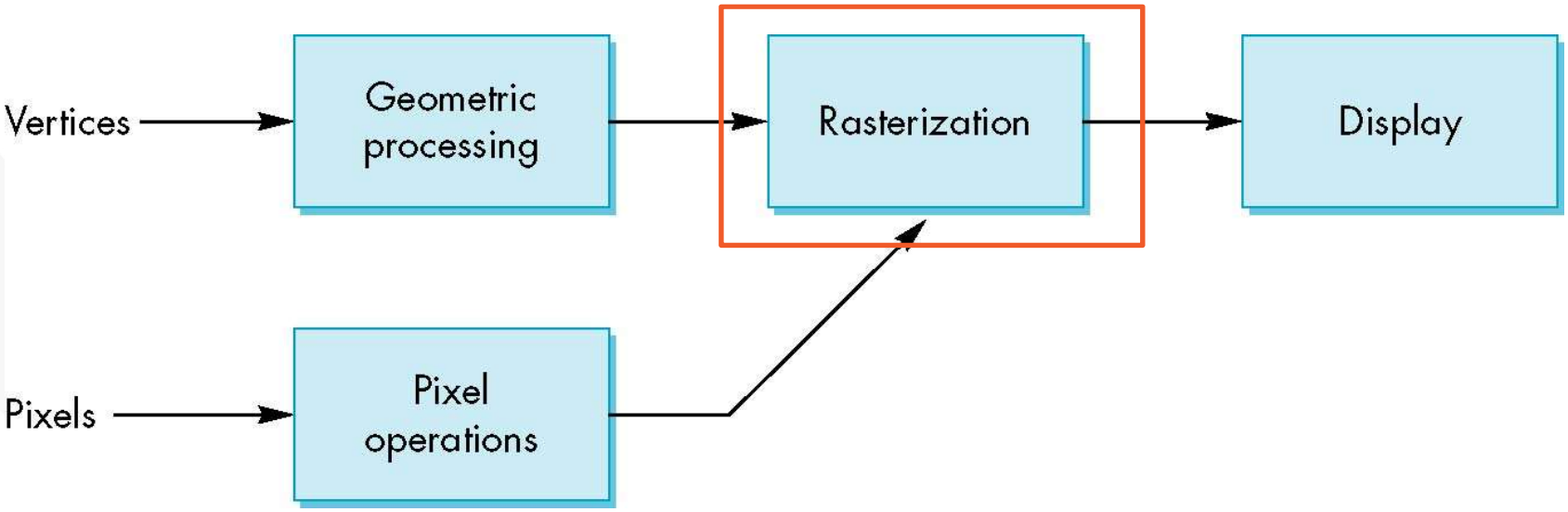


Where Does Mapping Take Place?



Where Does Mapping Take Place?

- **Mapping techniques are implemented at the end of the rendering pipeline**
  - Very efficient because few polygons pass down the geometric pipeline

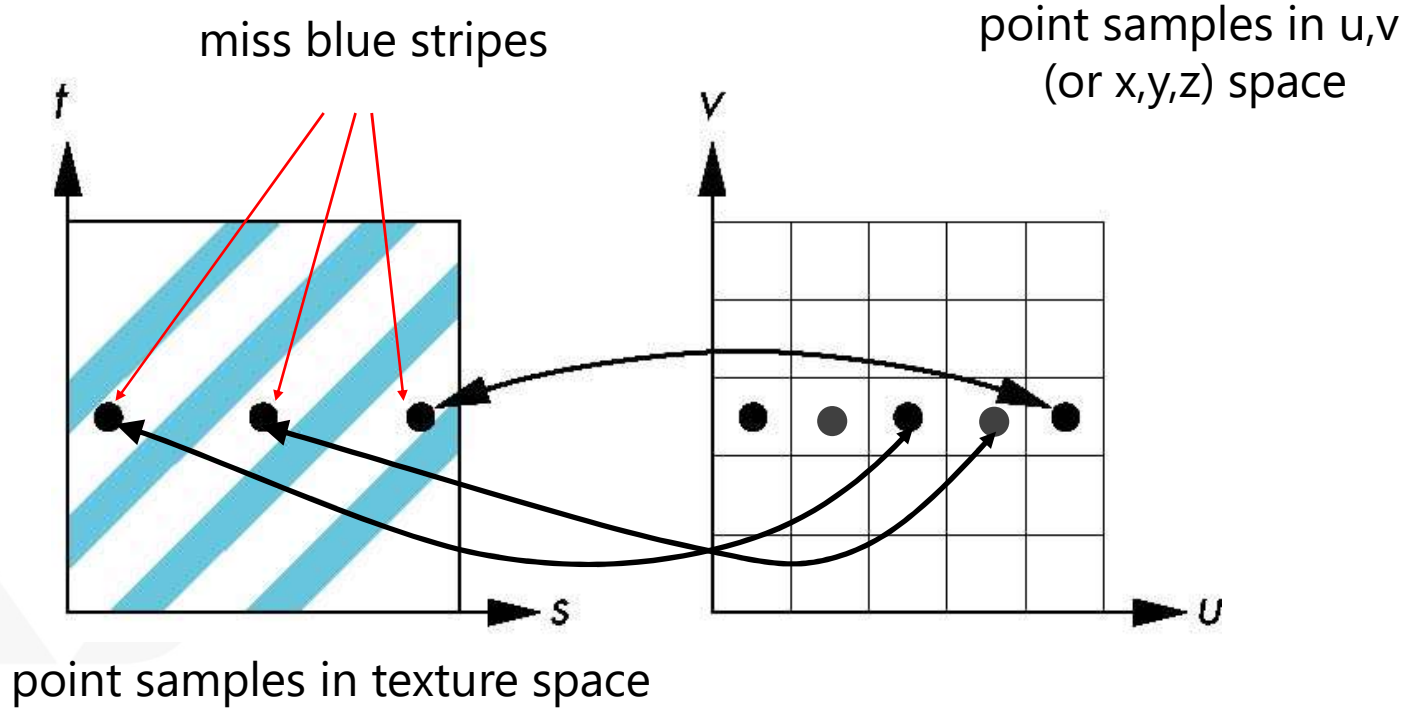


Aliasing

- Point sampling of the texture can lead to aliasing errors

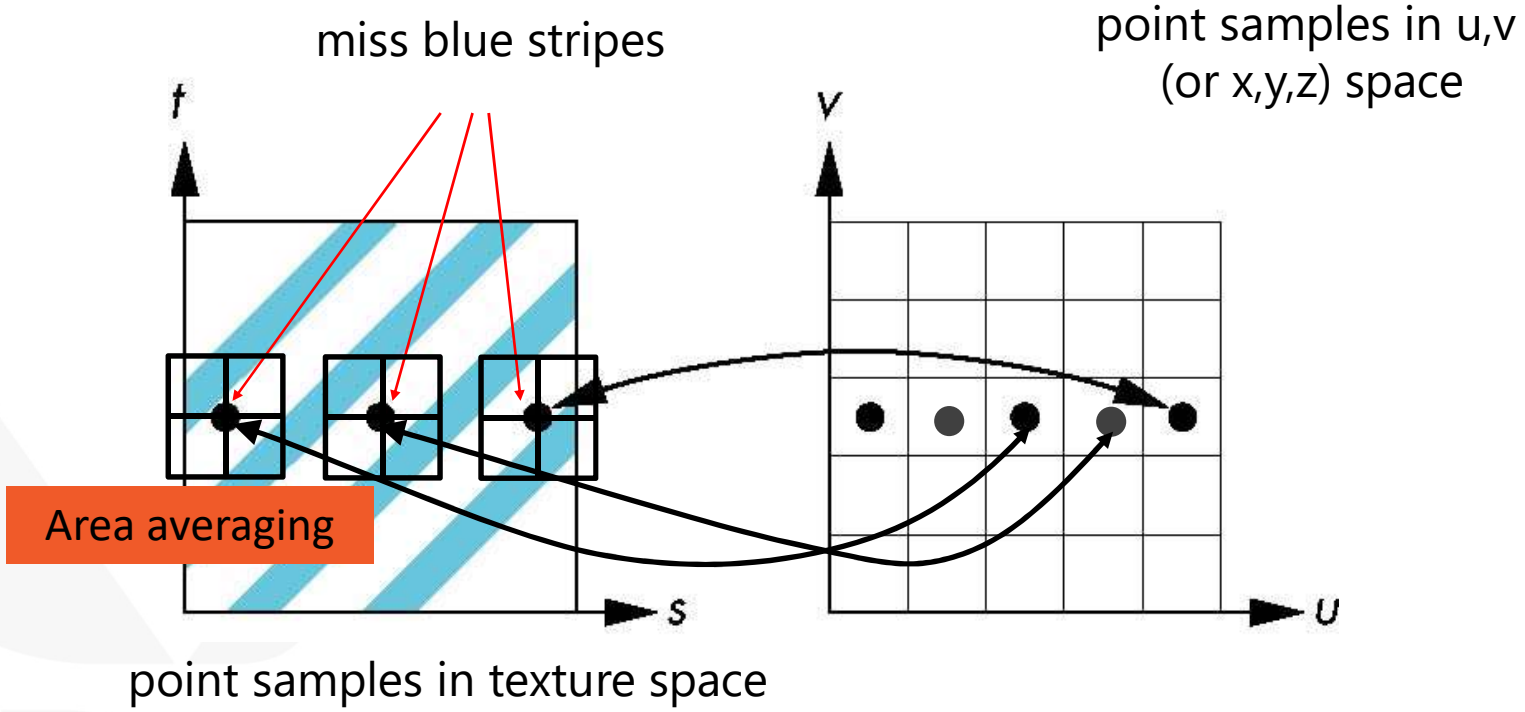
Aliasing

- Point sampling of the texture can lead to aliasing errors



Aliasing

- Point sampling of the texture can lead to aliasing errors

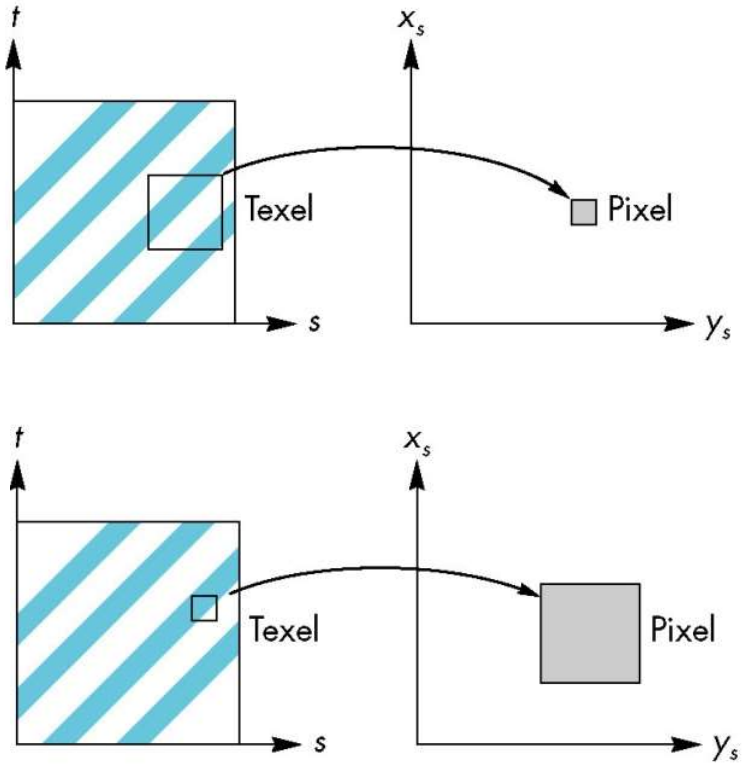


Sampling Issues

- Works fine as long as the size of the rendered image is approximately the same size as the texture source

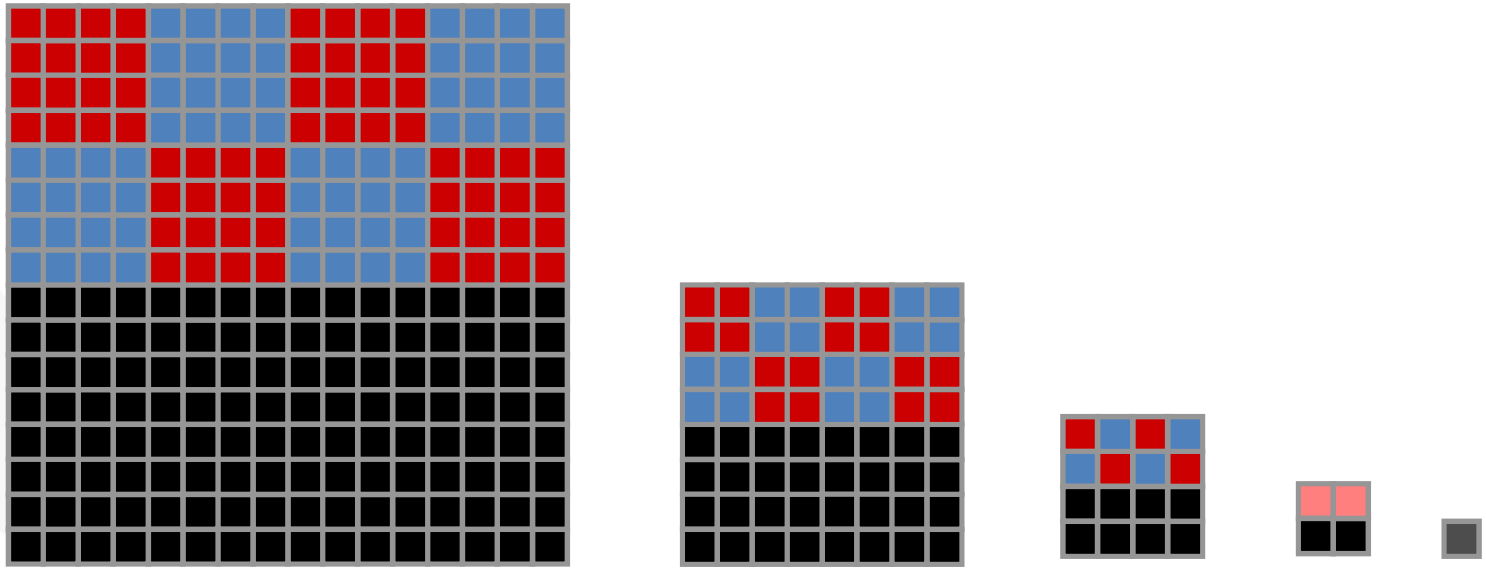
Sampling Issues

- **Works fine as long as the size of the rendered image is approximately the same size as the texture source**
  - What if much smaller?
  - What if much bigger?



Mip-mapping

- **Mip-mapping is a technique that creates multiple resolutions of an image**
  - i.e. Takes a 512x512 image and filters it to create 256x256, 128x128, 64x64, ..., 1x1 versions of it



Original Texture

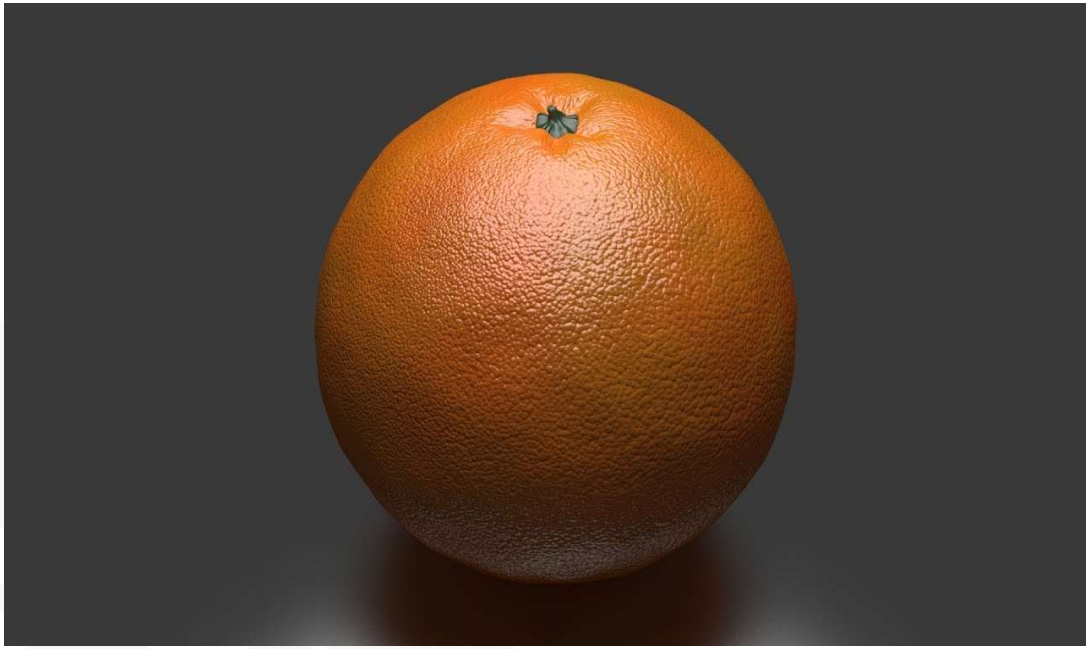
Lower Resolution Versions

Mip-mapping

- **Mip-mapping is a technique that creates multiple resolutions of an image**
  - i.e. Takes a 512x512 image and filters it to create 256x256, 128x128, 64x64, ..., 1x1 versions of it
- **When texture coordinates are looked up, the most appropriate mip-map level is used.**
  - Or, more likely, interpolates between the two closest

Modeling an Orange

- **Texture map a photo of an orange onto a surface**
  - Captures dimples
  - Not correct if we move viewer or light

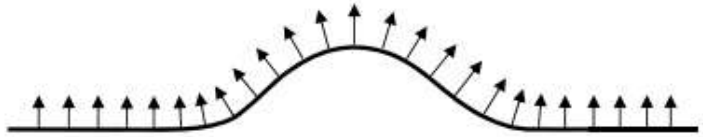


Bump Mapping by Jim Blinn

- Perturbed Normals

Bump Mapping by Jim Blinn

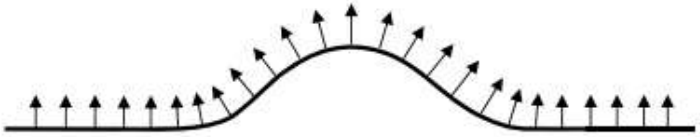
- Perturbed Normals



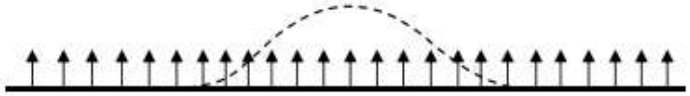
Surface normals on a real “bump”

Bump Mapping by Jim Blinn

- Perturbed Normals



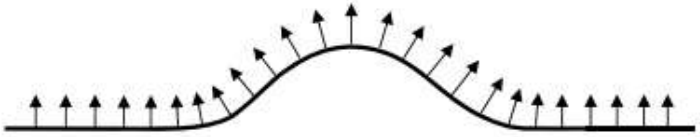
Surface normals on a real "bump"



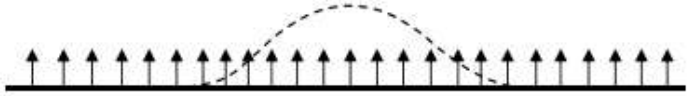
Surface normals on a flat polygon

Bump Mapping by Jim Blinn

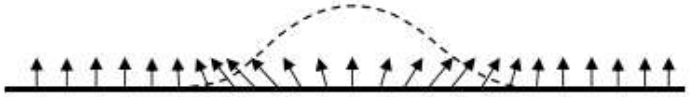
- Perturbed Normals



Surface normals on a real "bump"



Surface normals on a flat polygon

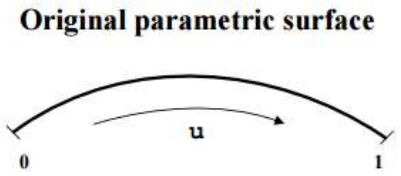


Modified ("perturbed") normals

Bump Mapping

- **Requires per-pixel (Phong) shading**
  - Just interpolating from the vertex normals gives a smooth-looking surface

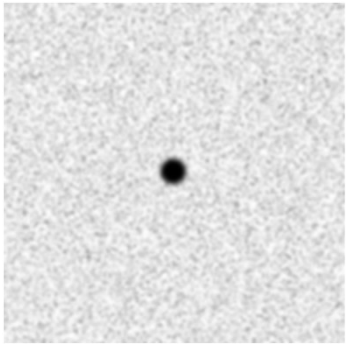
Phong shading



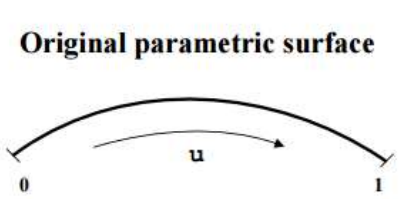
Bump Mapping

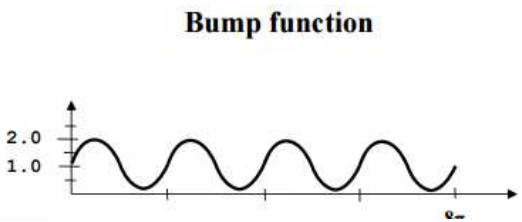
- Requires per-pixel (Phong) shading
  - Just interpolating from the vertex normals gives a smooth-looking surface
- Bump mapping uses a “texture” to define how much to perturb the normal at that point

Phong shading



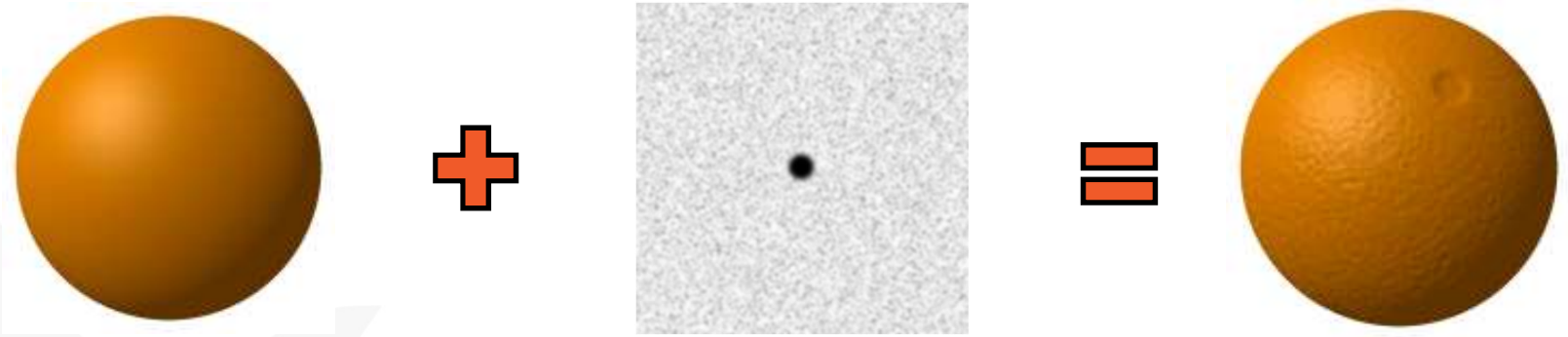
Bump Map



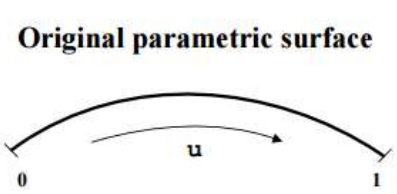


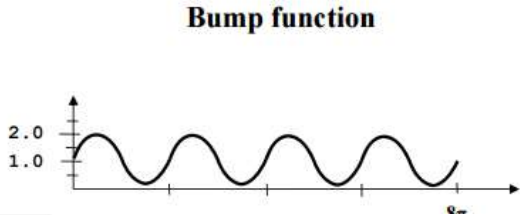
Bump Mapping

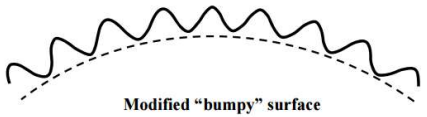
- **Requires per-pixel (Phong) shading**
  - Just interpolating from the vertex normals gives a smooth-looking surface
- **Bump mapping uses a "texture" to define how much to perturb the normal at that point**
  - Results in a "bumpy" surface



Phong shading | Bump Map | Bump Mapped Sphere

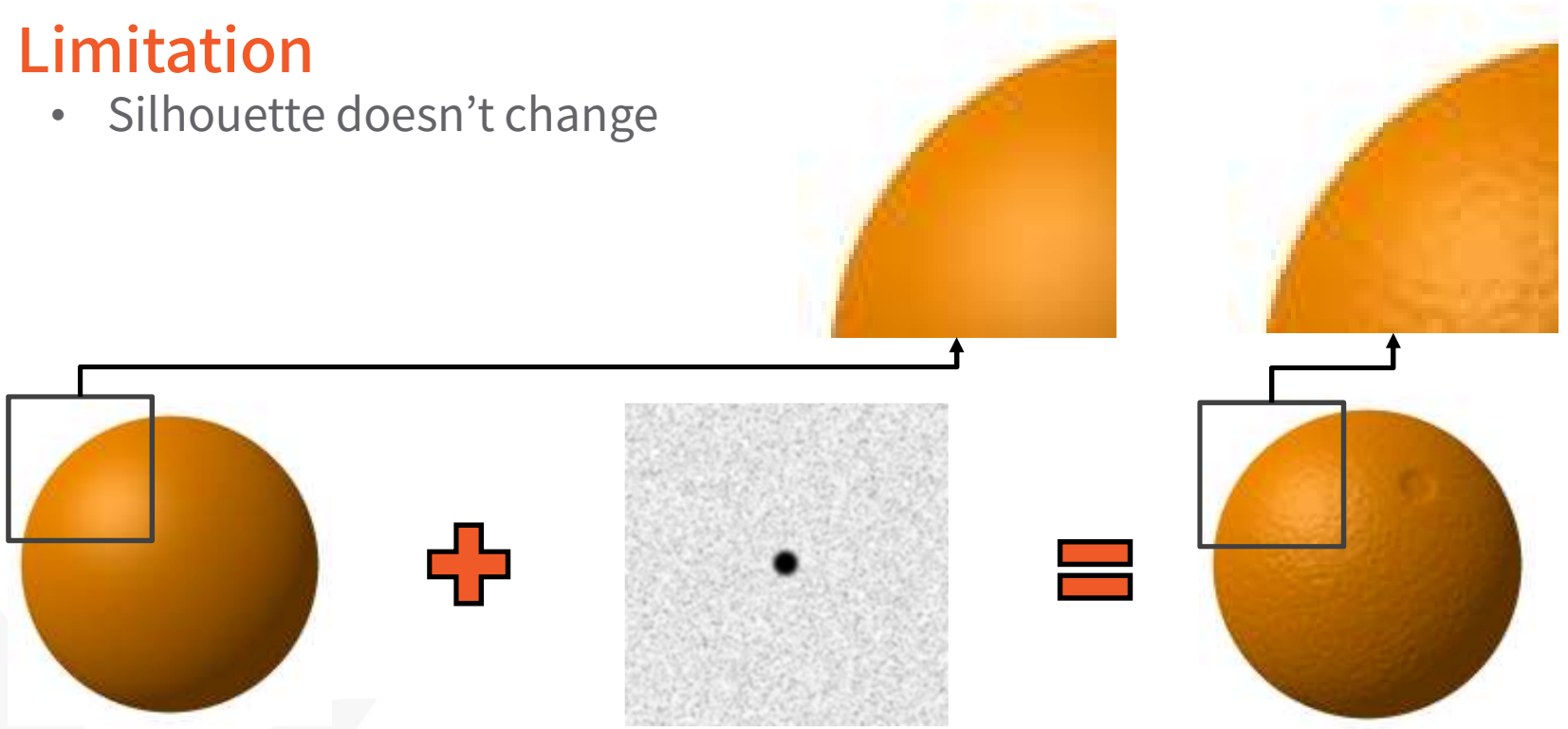




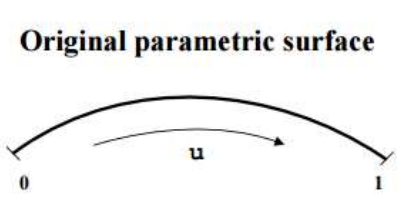


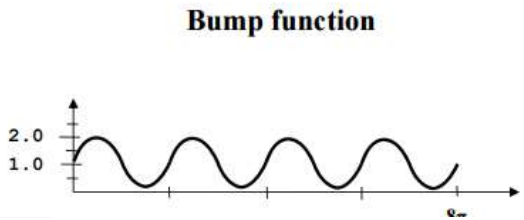
Bump Mapping

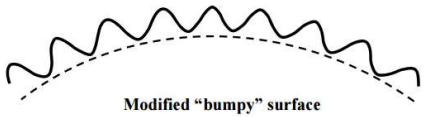
- **Limitation**
  - Silhouette doesn't change



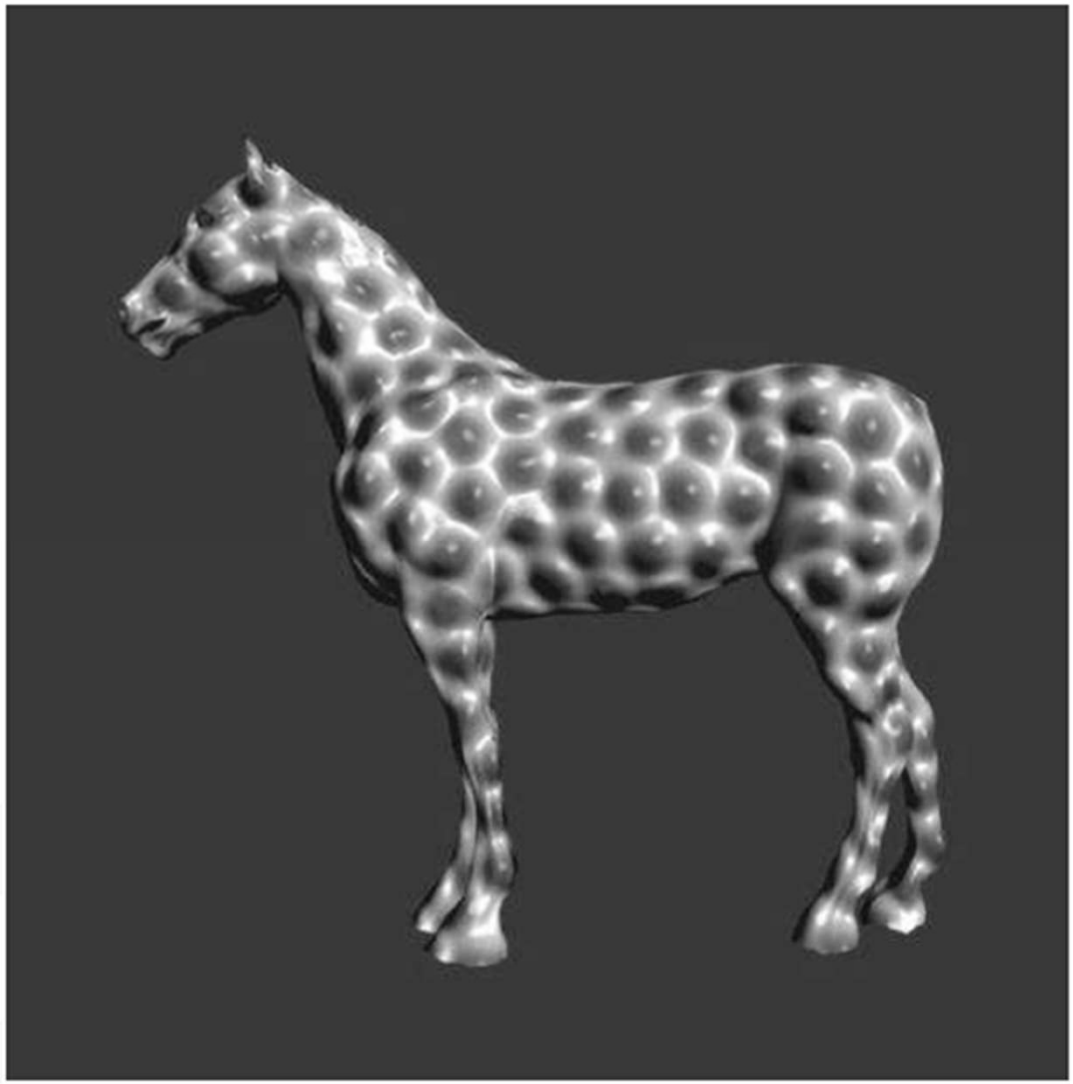
Phong shading | Bump Map | Bump Mapped Sphere





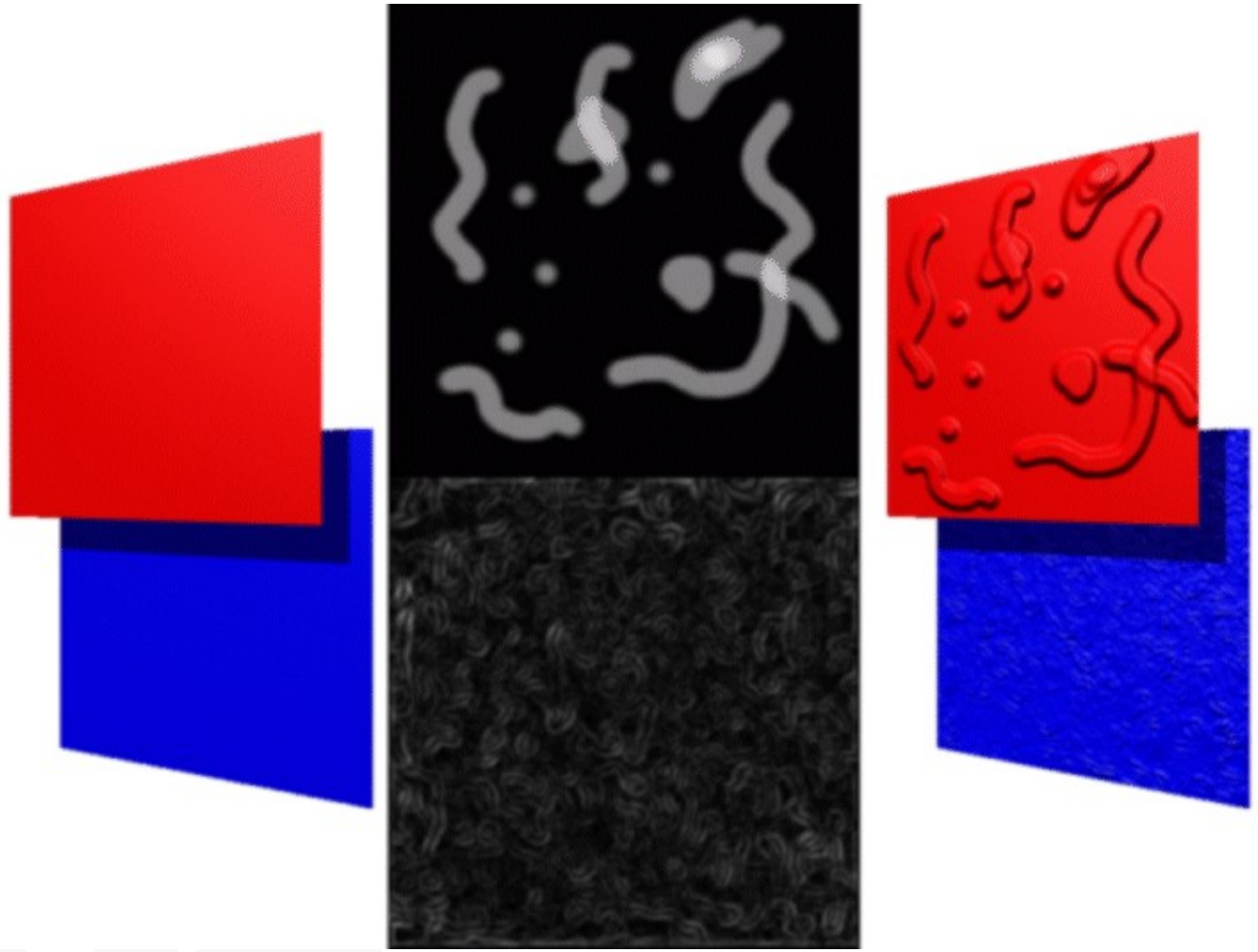


Bump Map Example



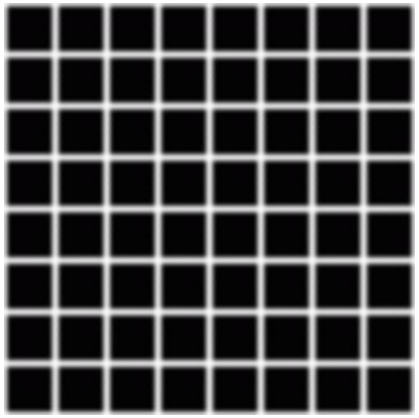
Greg Turk

Bump Map Example

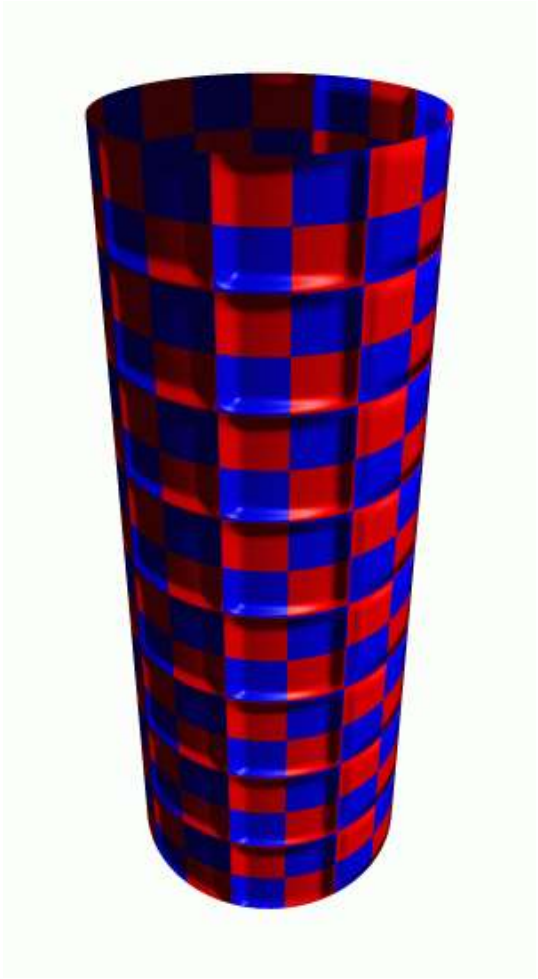


Bump Map Example

Cylinder w/Diffuse Texture Map



Bump Map

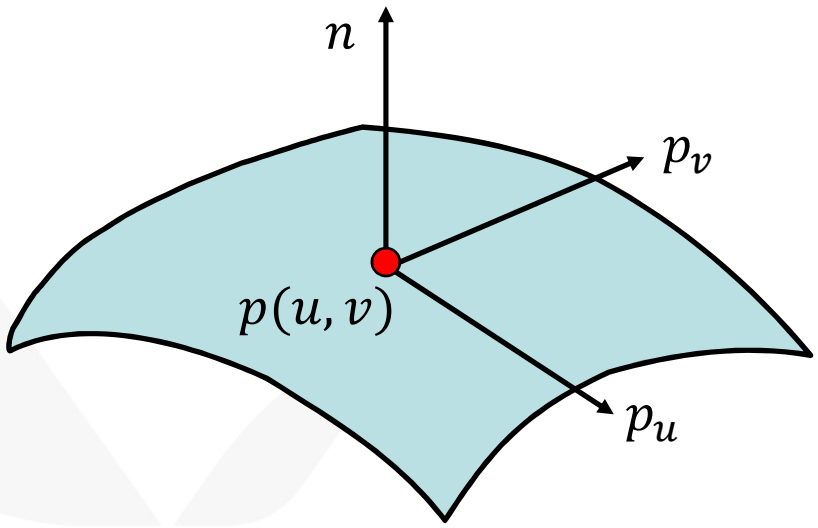


Cylinder w/Texture Map & Bump Map

Equations

- Let $p(u,v)$ be a point on a parametric surface.

$$p(u,v) = [x(u,v), y(u,v), z(u,v)]^T$$



$$p_u = \left[\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right]^T$$

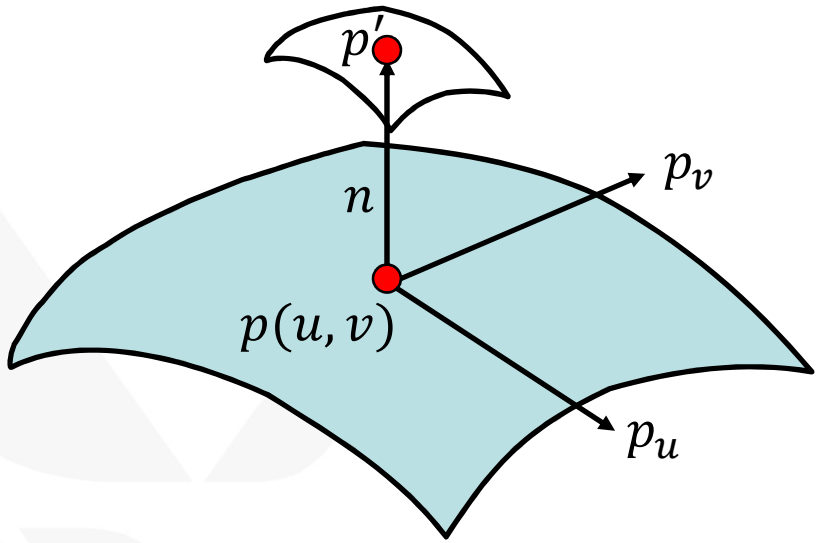
$$p_v = \left[\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right]^T$$

$$n = \frac{(p_u \times p_v)}{|p_u \times p_v|}$$

Equations

- $\mathrm{d}(u, v)$ **is the bump or displacement function, which we can assume is known and small** $(|d(u, v)| \ll 1)$
  - The displaced surface is given by

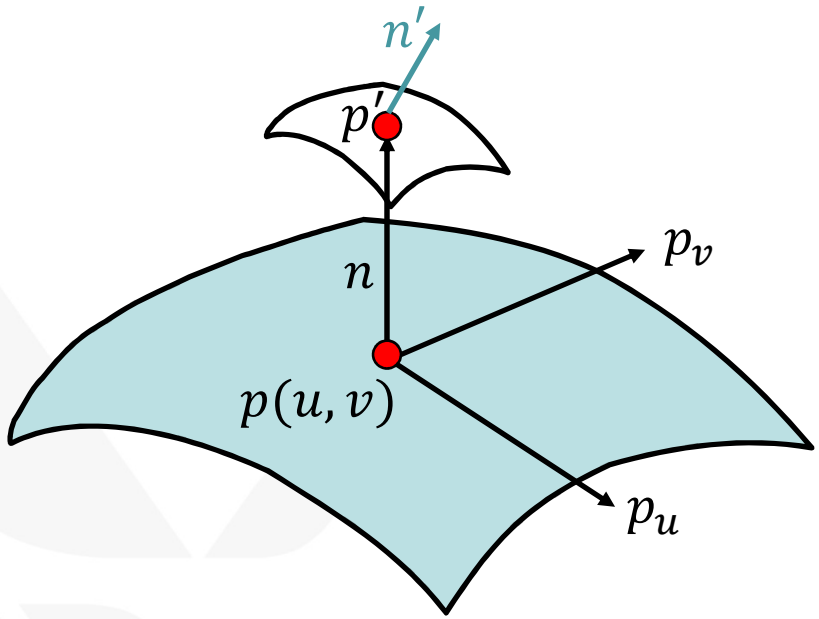
$$p' = p + d(u, v)n$$



Equations

- $\mathrm{d}(u, v)$ **is the bump or displacement function, which we can assume is known and small** $(|d(u,v)| \ll 1)$
  - The displaced surface is given by

$$p' = p + d(u, v)n$$

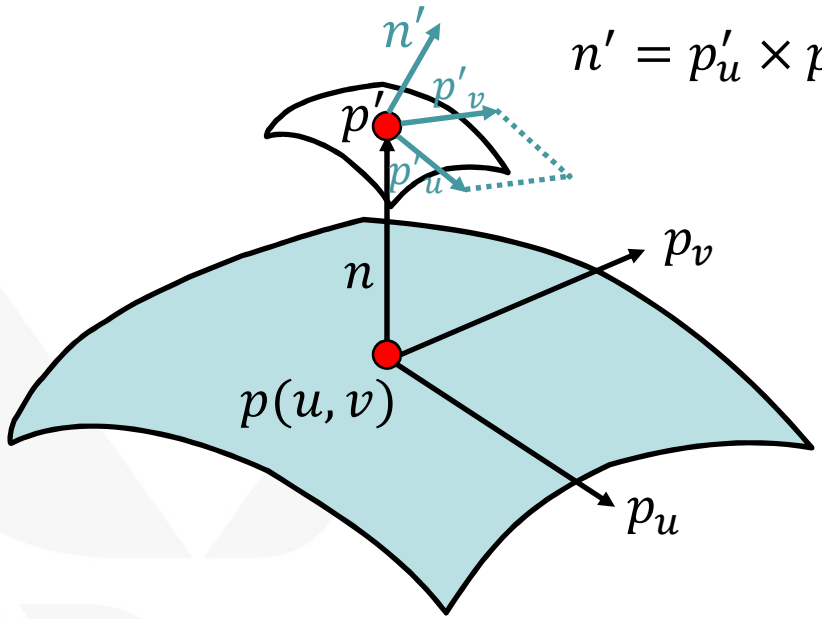


Equations

- $\mathrm{d}(u,v)$ **is the bump or displacement function, which we can assume is known and small** $(|d(u,v)| \ll 1)$
  - The displaced surface is given by

$$p' = p + d(u,v)n$$

$$n' = p'_u \times p'_v$$



Differentiating p' leads to :

$$p'_u = p_u + \frac{\partial d}{\partial u} n + d(u,v) n_u$$

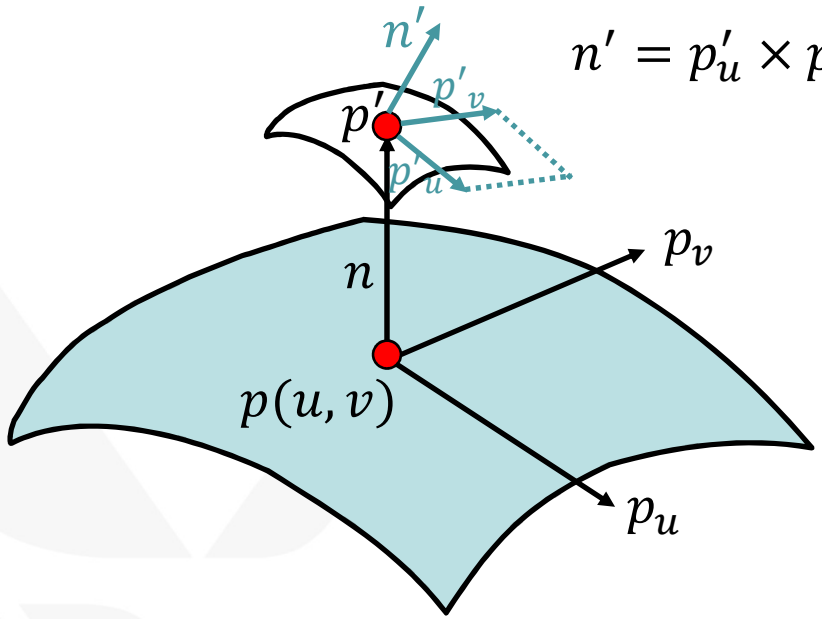
$$p'_v = p_v + \frac{\partial d}{\partial v} n + d(u,v) n_v$$

Equations

- $\mathrm{d}(u,v)$ **is the bump or displacement function, which we can assume is known and small** $(|d(u,v)| \ll 1)$
  - The displaced surface is given by

$$p' = p + d(u,v)n$$

$$n' = p'_u \times p'_v$$



Differentiating of p' leads to :

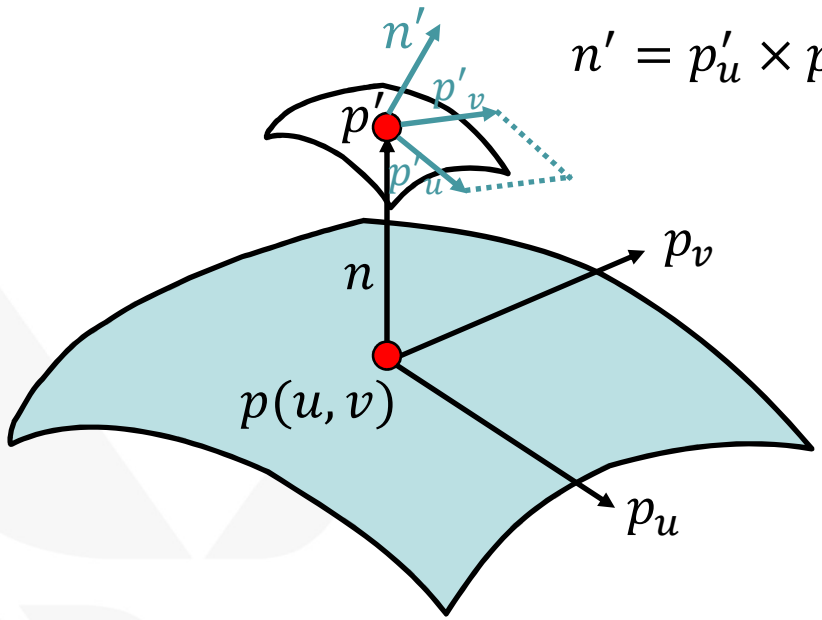
$$p'_u = p_u + \frac{\partial d}{\partial u} n + d(u,v) n_u$$

$$p'_v = p_v + \frac{\partial d}{\partial v} n + d(u,v) n_v$$

If d is small, we can neglect last term

Equations

- $\mathrm{d}(u,v)$ **is the bump or displacement function, which we can assume is known and small** $(|d(u,v)| \ll 1)$
  - The normal n' at the perturbed point p' is given by the cross product



$$n' = p'_u \times p'_v = \left(p_u + \frac{\partial d}{\partial u} n\right) \times \left(p_v + \frac{\partial d}{\partial v} n\right)$$

$$\approx (p_u \times p_v) + p_u \times \frac{\partial d}{\partial v} n + p_v \times \frac{\partial d}{\partial u} n$$

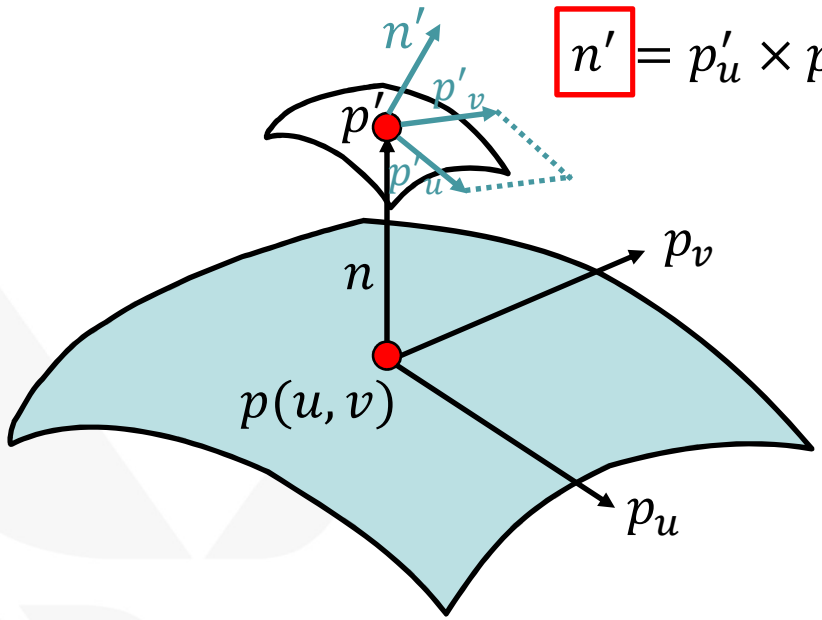
$$\approx n + \underbrace{p_u \times \frac{\partial d}{\partial v} n + p_v \times \frac{\partial d}{\partial u} n}$$

These are displacement, difference between the original and perturbed normal

The vectors $n \times p_v$ and $n \times p_u$ lie in the tangent plane.

Equations

- $\mathrm{d}(u,v)$ **is the bump or displacement function, which we can assume is known and small** $(|d(u,v)| \ll 1)$
  - The normal n' at the perturbed point p' is given by the cross product



$$n' = p'_u \times p'_v = \left(p_u + \frac{\partial d}{\partial u} n\right) \times \left(p_v + \frac{\partial d}{\partial v} n\right)$$

$$\approx (p_u \times p_v) + p_u \times \frac{\partial d}{\partial v} n + p_v \times \frac{\partial d}{\partial u} n$$

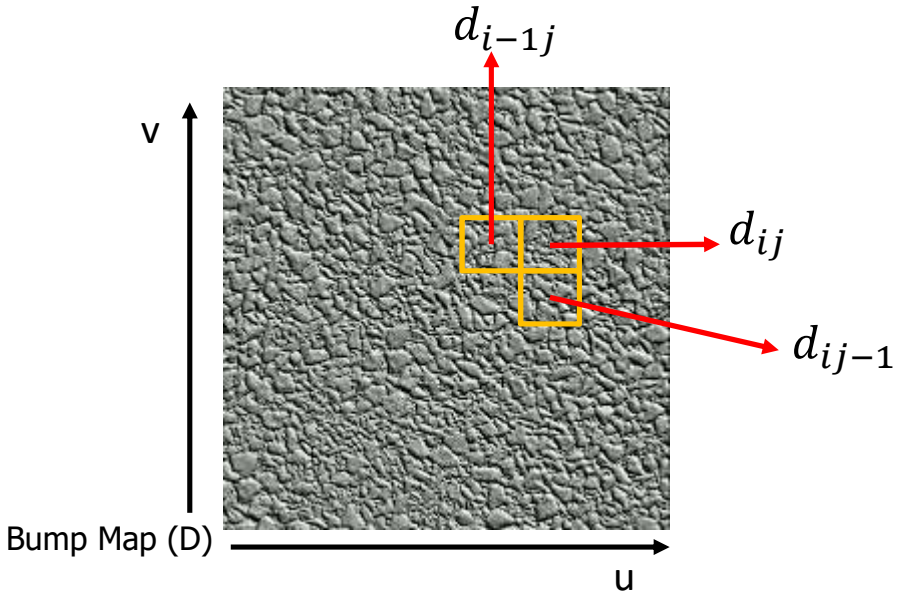
$$\approx n + \underbrace{p_u \times \frac{\partial d}{\partial v} n + p_v \times \frac{\partial d}{\partial u} n}$$

These are displacement, difference between the original and perturbed normal

The vectors $n \times p_v$ and $n \times p_u$ lie in the tangent plane.

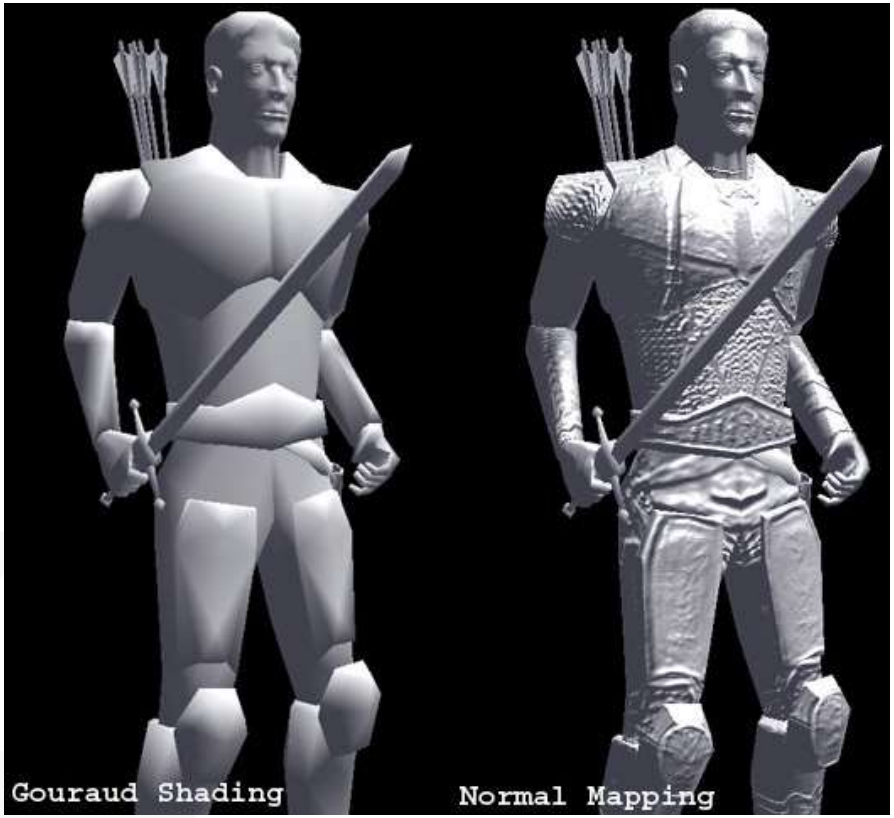
Image Processing

- Suppose that we start with a function $d(u, v)$
- We can sample it to form an array $\mathrm{D} = [\mathrm{d_{ij}}]$
- Then $\frac{\partial d}{\partial u} \approx d_{ij} - d_{i-1,j}$ and $\frac{\partial d}{\partial v} \approx d_{ij} - d_{i,j-1}$

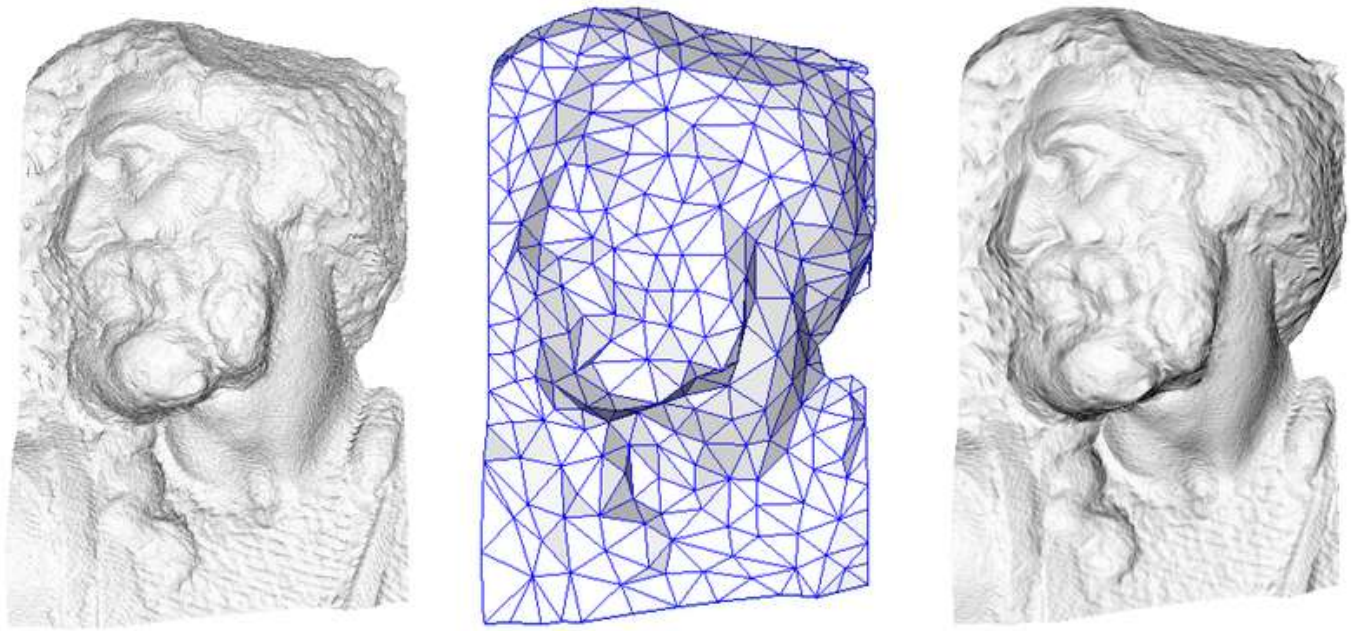


Normal Mapping

- Exploiting a normal map coming from a high resolution model



Normal Mapping

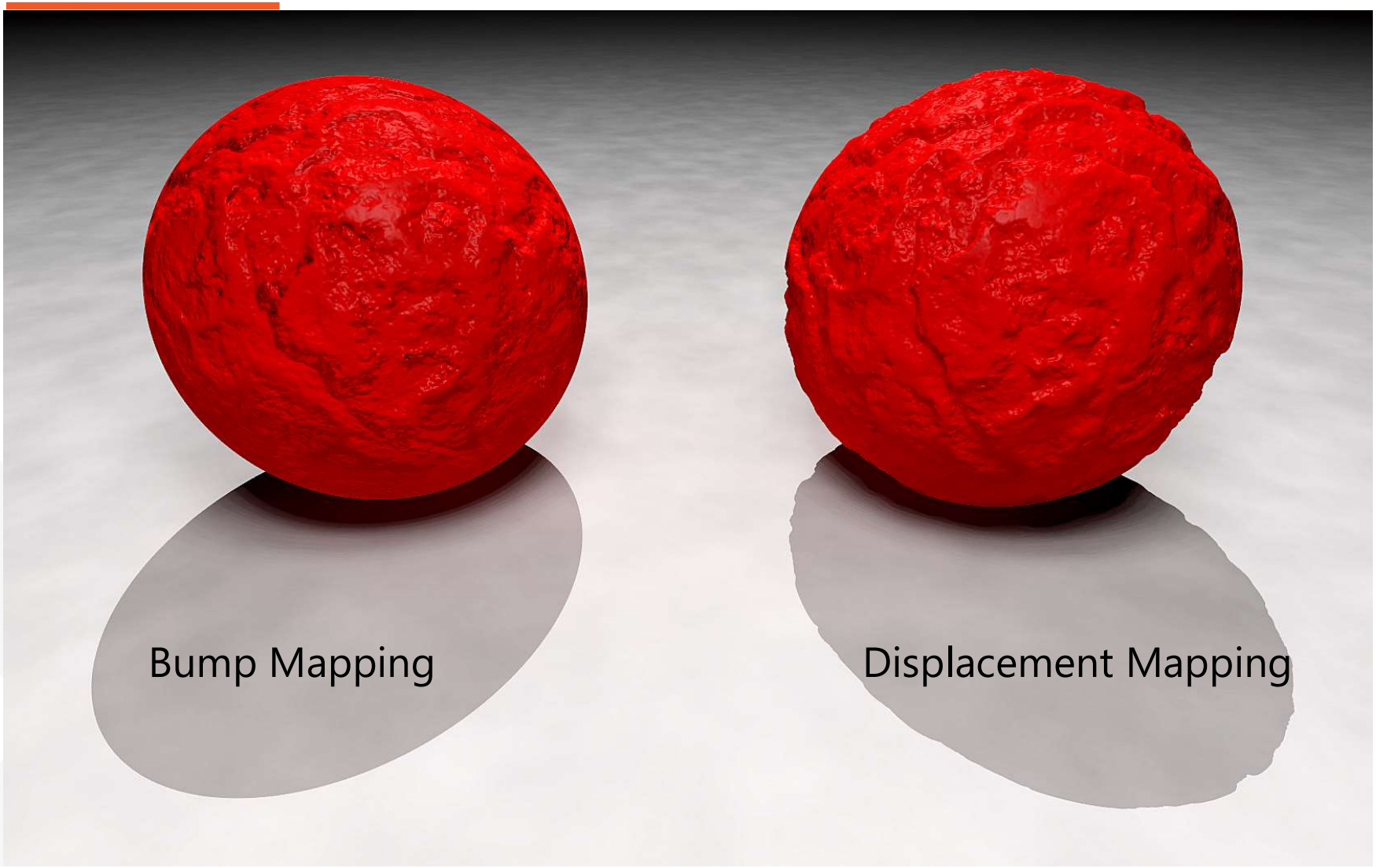


original mesh
4M triangles

simplified mesh
500 triangles

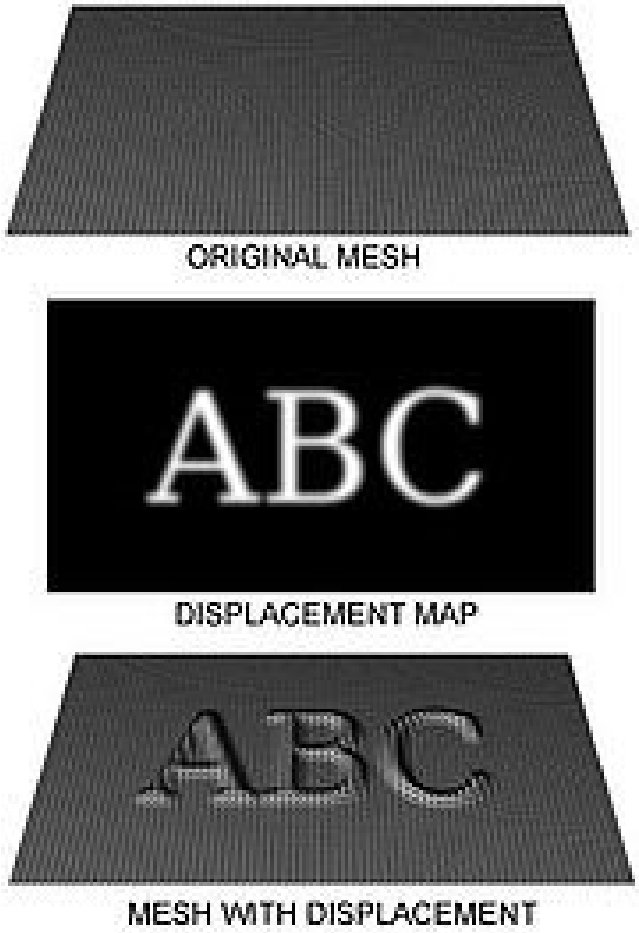
simplified mesh
and normal mapping
500 triangles

Displacement Mapping

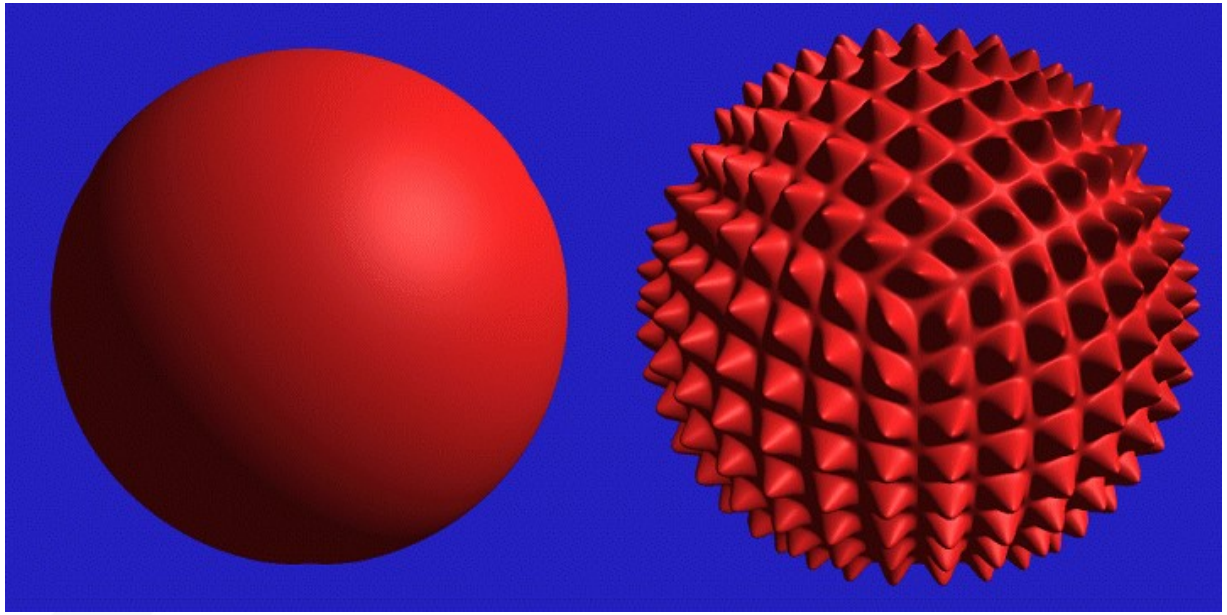


Displacement Mapping

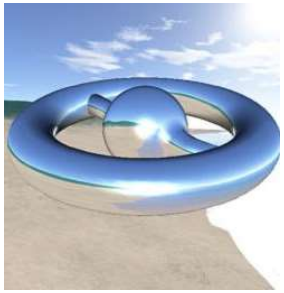
- We use the texture map to actually move the surface point. This is called displacement mapping.



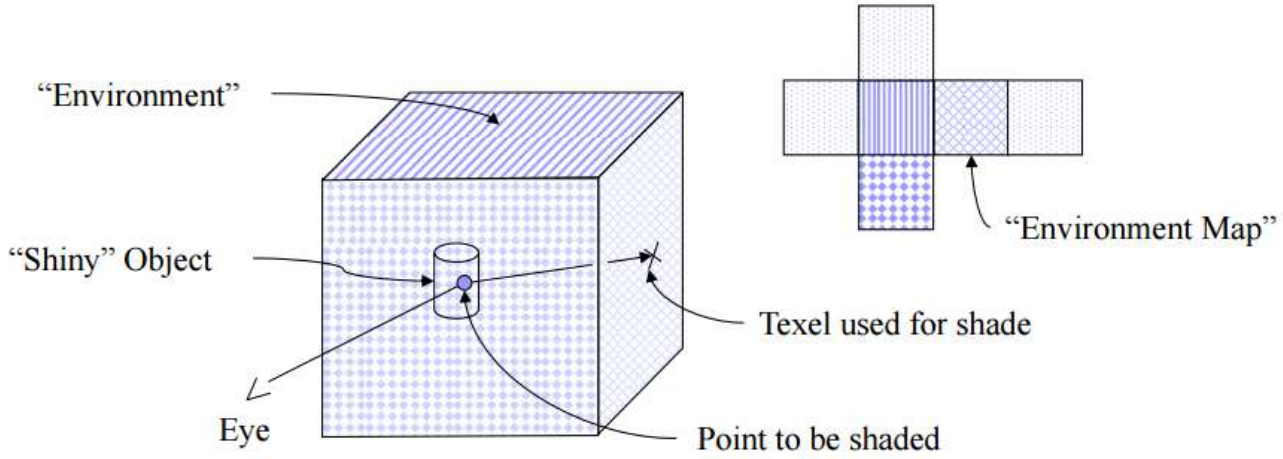
Displacement Mapping



Environment Mapping



- Texture contains an image of the surroundings
- Cube mapping is commonly used to implement environment maps



This allows us to "hack" reflection/refraction

- Render the scene from the center of the cube in each face direction
- Store each of these results into a texture
- Then render the scene from the actual viewpoint, applying the environment textures

Environment Mapping Example