Interactive Computer Graphics: Lecture 8

Texture mapping

Some slides adopted from
H. Pfister, Harvard

The Problem:

- We don't want to represent all this detail with geometry

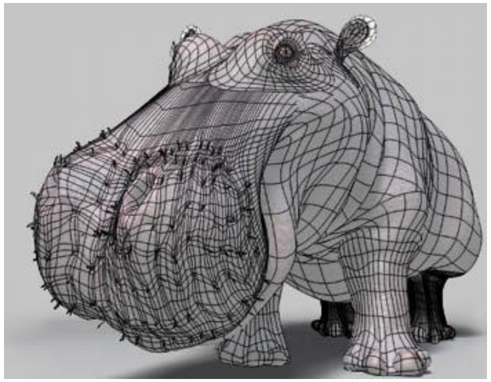
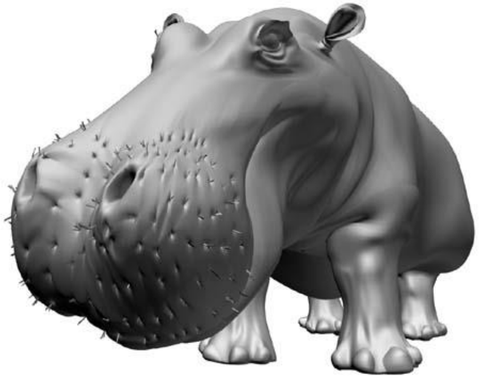
The Solution: Textures

- The visual appearance of a graphics scene can be greatly enhanced by the use of texture.

- Consider a brick building, using a polygon for every brick require a huge effort in scene design.

- So why not use one polygon and draw a repeating brick pattern (texture) onto it?

The Quest for Visual Realism


Texture Definition

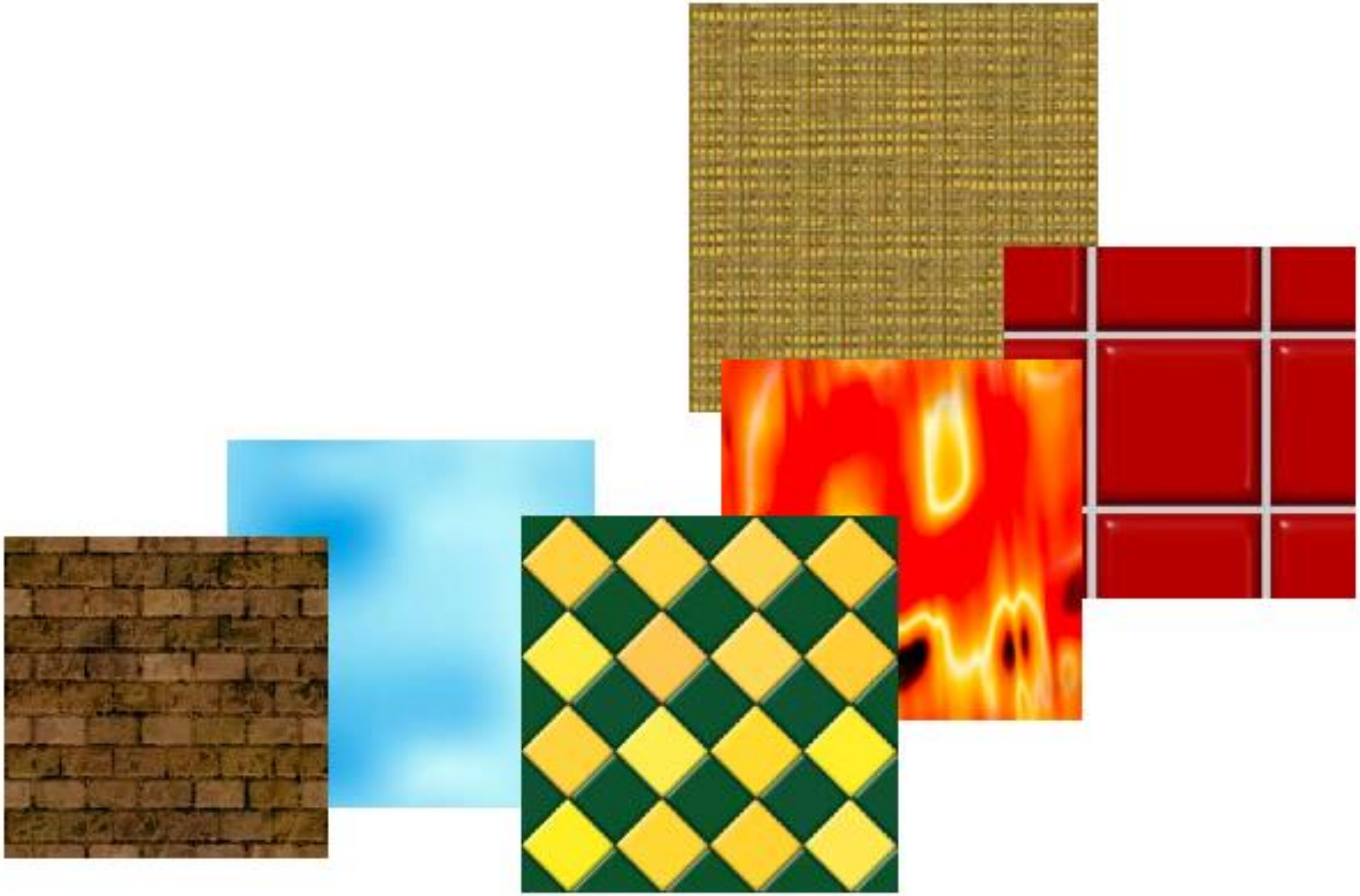
- Textures may be defined as:
  - One-dimensional functions
    - parameter can have arbitrary domain (e.g., incident angle)
  - Two-dimensional functions
    - information is calculated for every (u, v), many possibilities
  - Raster images ("texels")
    - Most often used method
    - Images from scanner, photos or calculation
  - Three-dimensional functions
    - Volume $T(u, v, w)$
- Procedural texture vs. raster data

Procedural textures

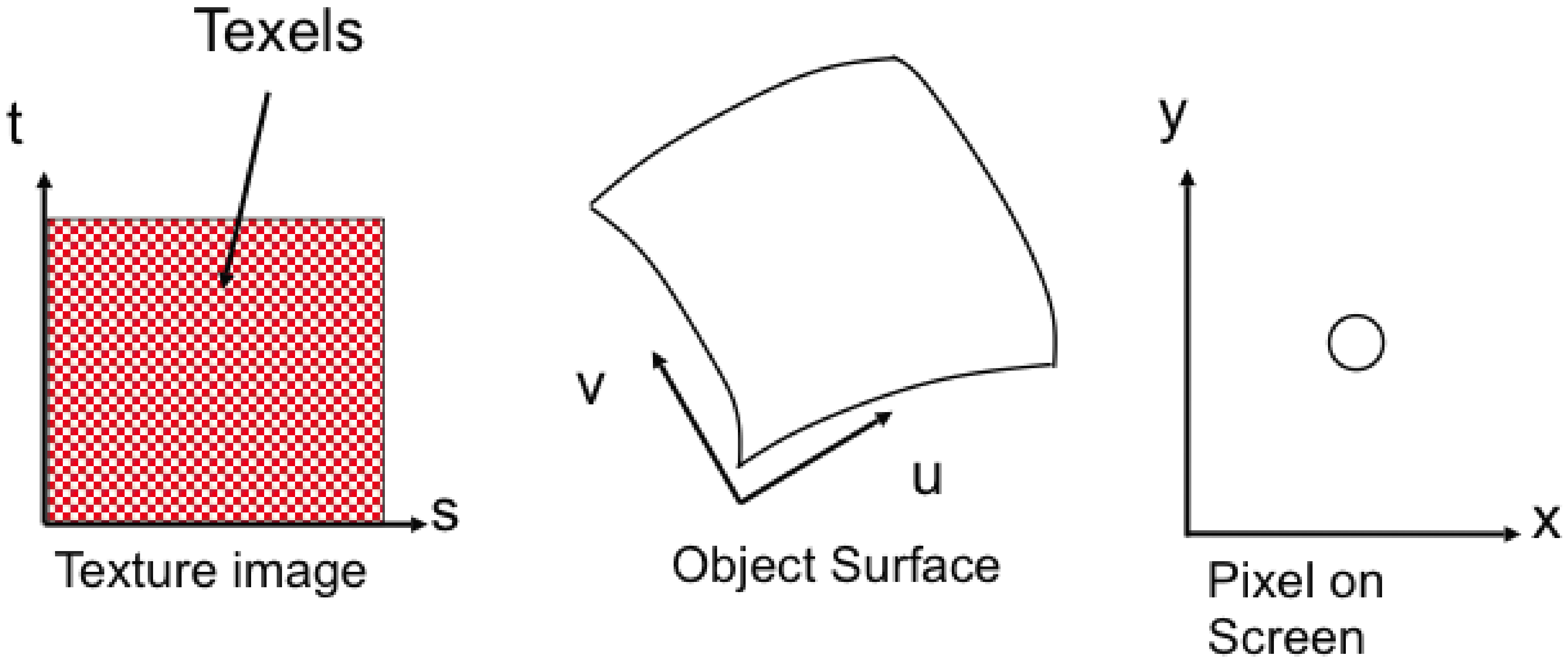
- Write a function: $F(\mathbf{p}) \rightarrow \text{color}$

- non-intuitive
- difficult to match a texture that already exists in the ‘real’ world

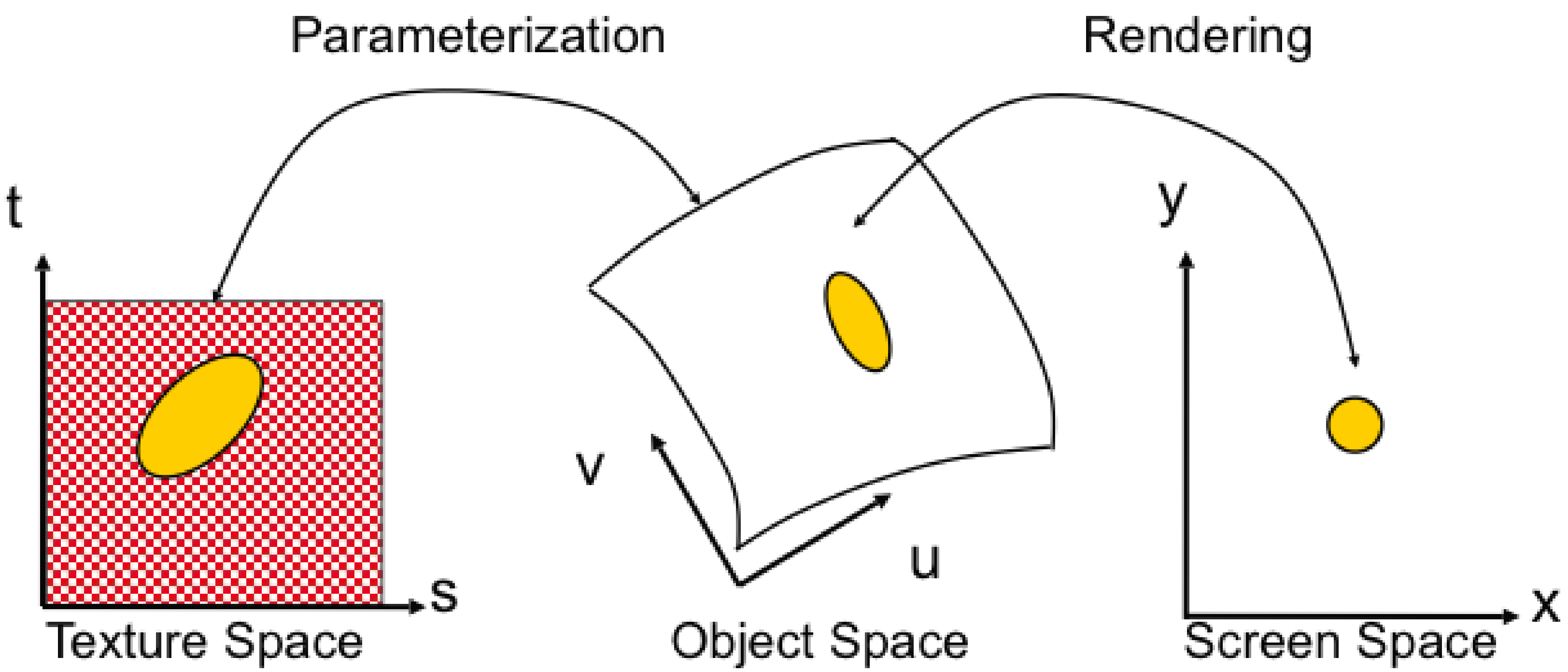
Photo textures



The concept of texture mapping

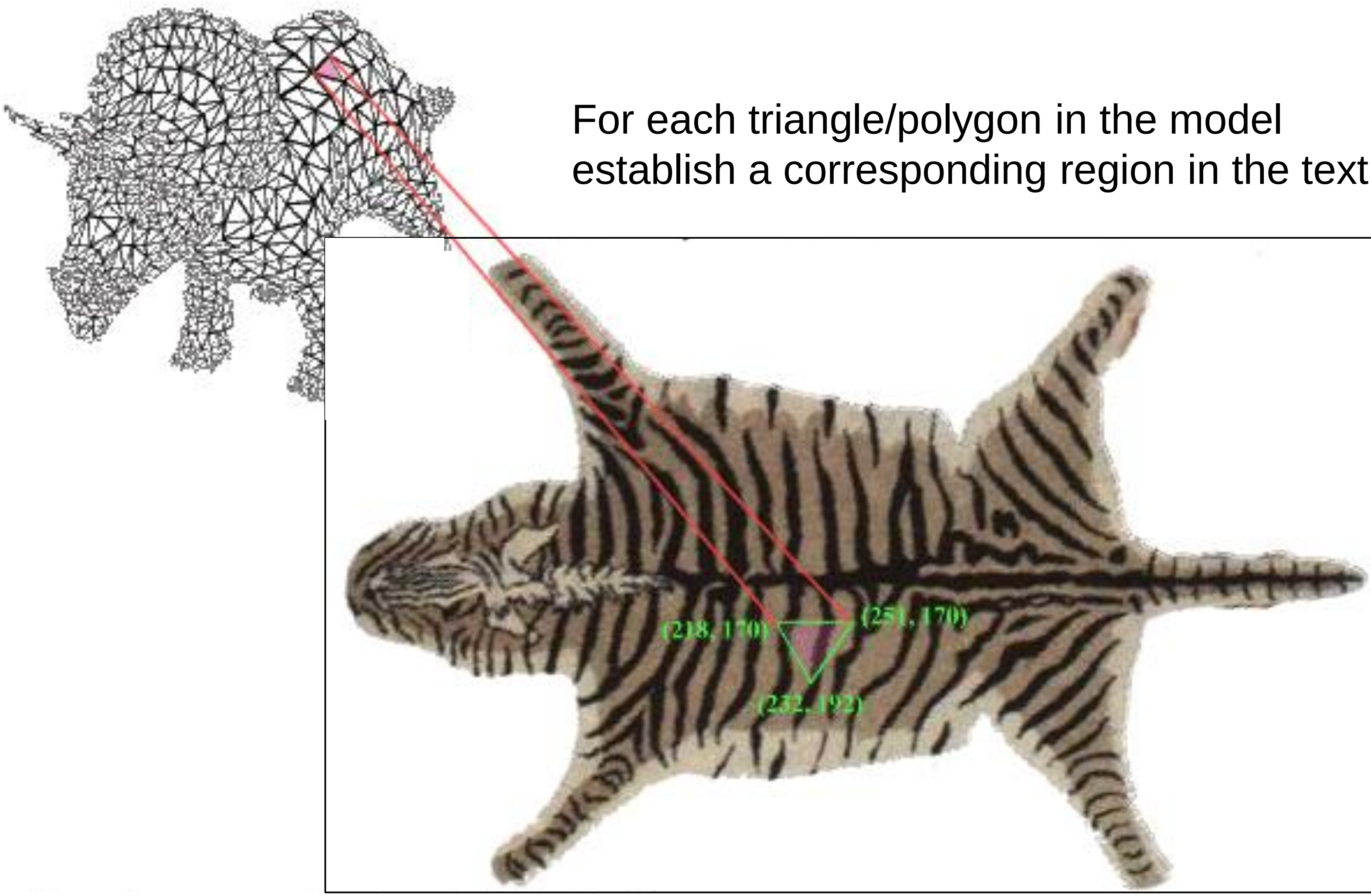


Texture mapping: Terminology



The concept of texture mapping

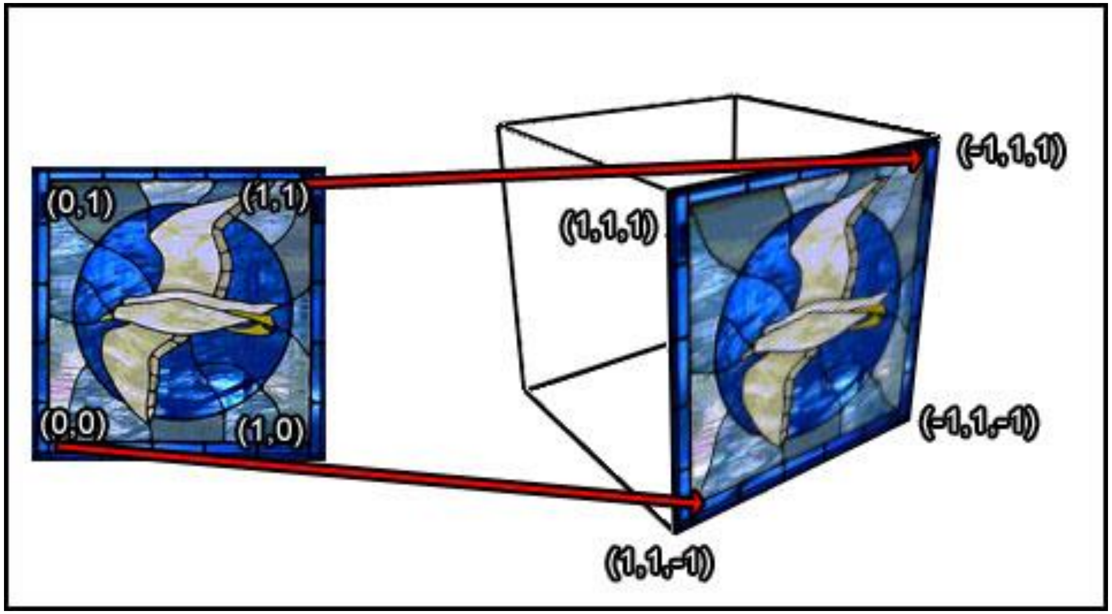
For each triangle/polygon in the model establish a corresponding region in the texture



During rasterization interpolate the coordinate indices into the texture map

Parametrization

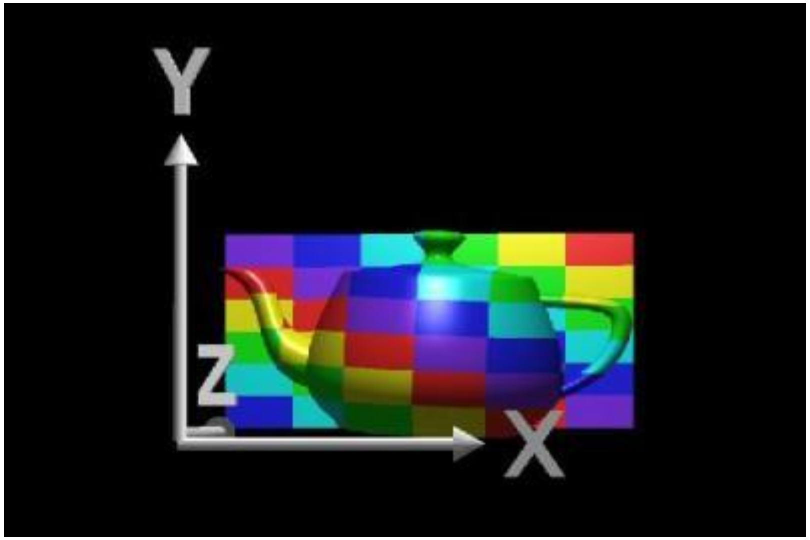
- How to do the mapping?

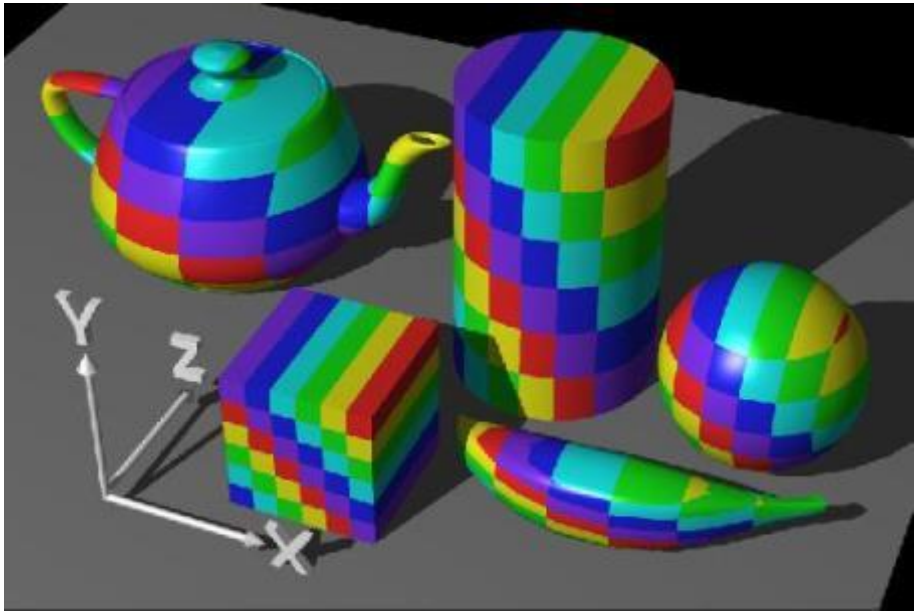


- Usually objects are not that simple

Parametrization: Planar

- Planar mapping: dump one of the coordinates

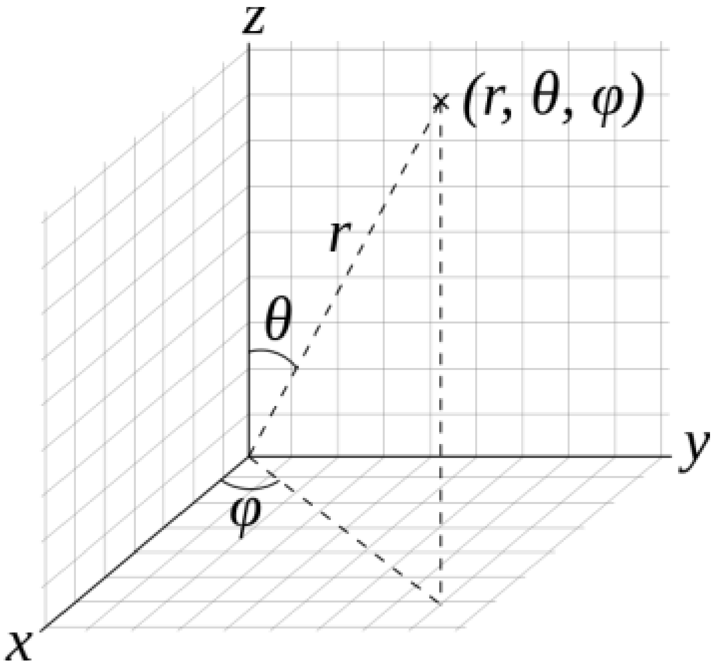




Only looks good from the front!

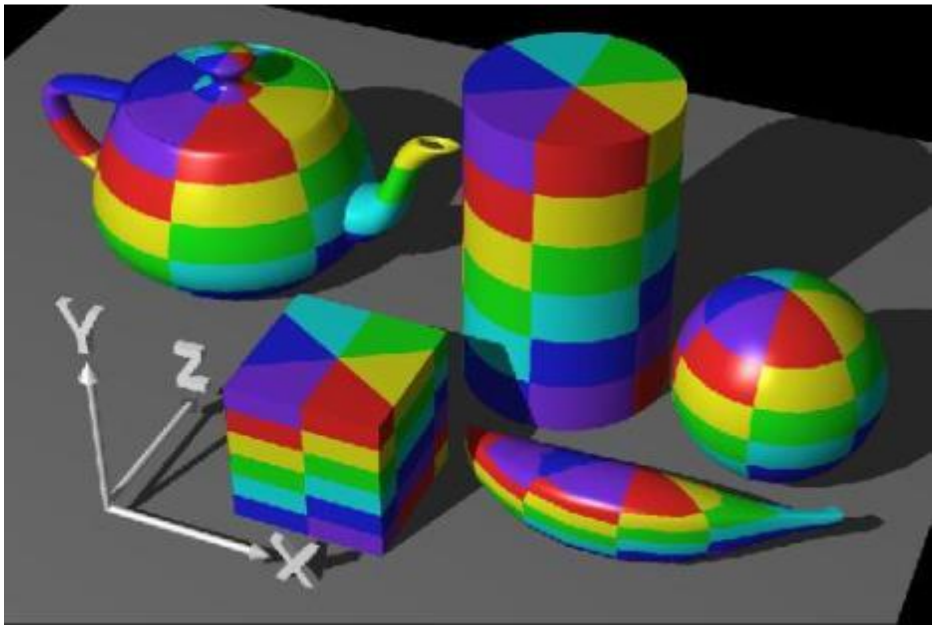
Parametrization

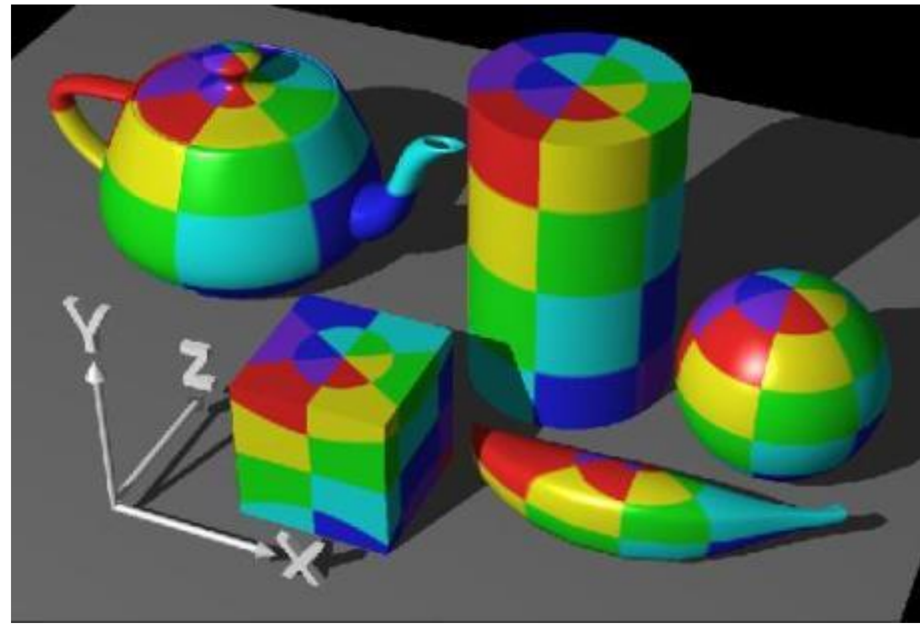
- Cylindrical and spherical mapping: compute angles between vertex and object center
- Compare to polar/spherical coordinate systems



Parametrization

- Cylindrical and spherical mapping



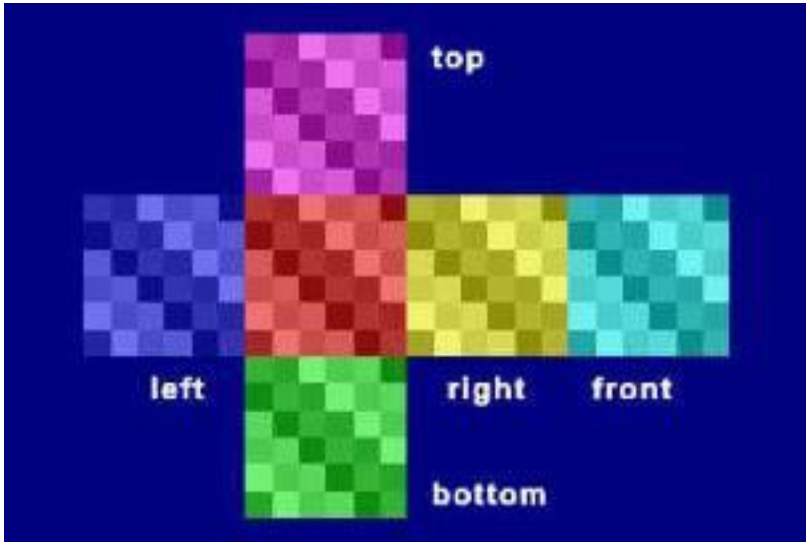


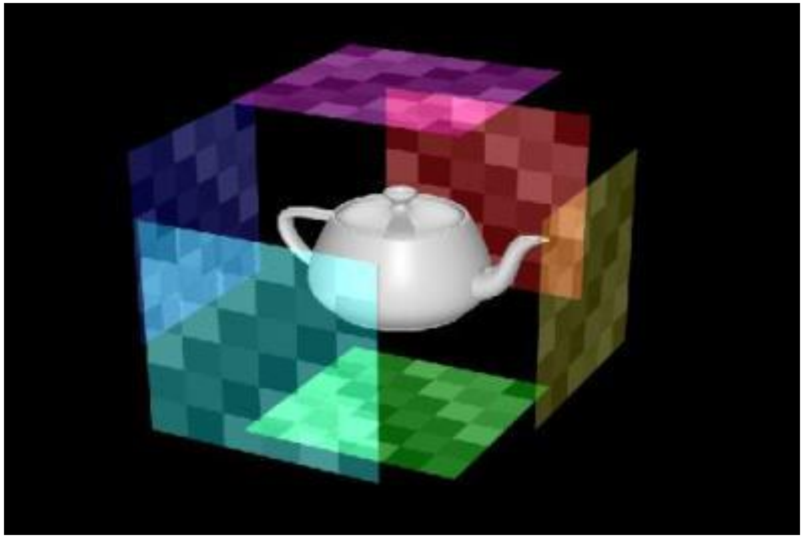
Cylindrical

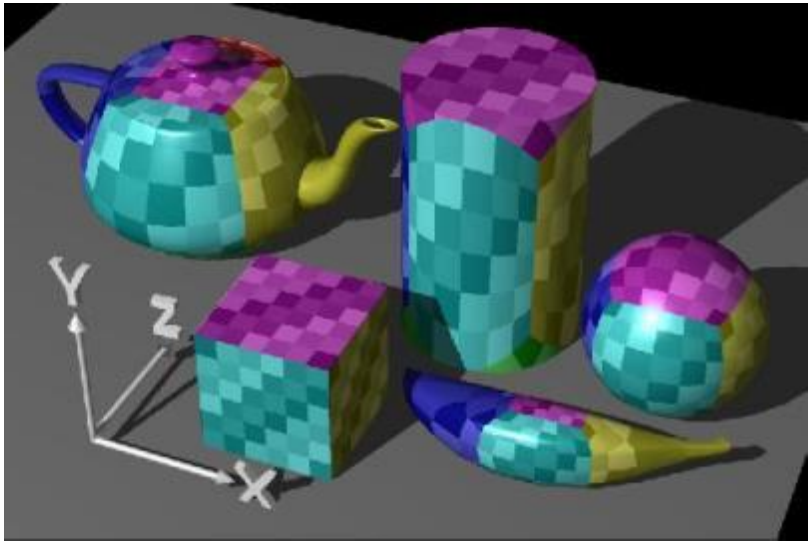
Spherical

Parametrization: Box

- Box mapping: used mainly for environment mapping (see later)







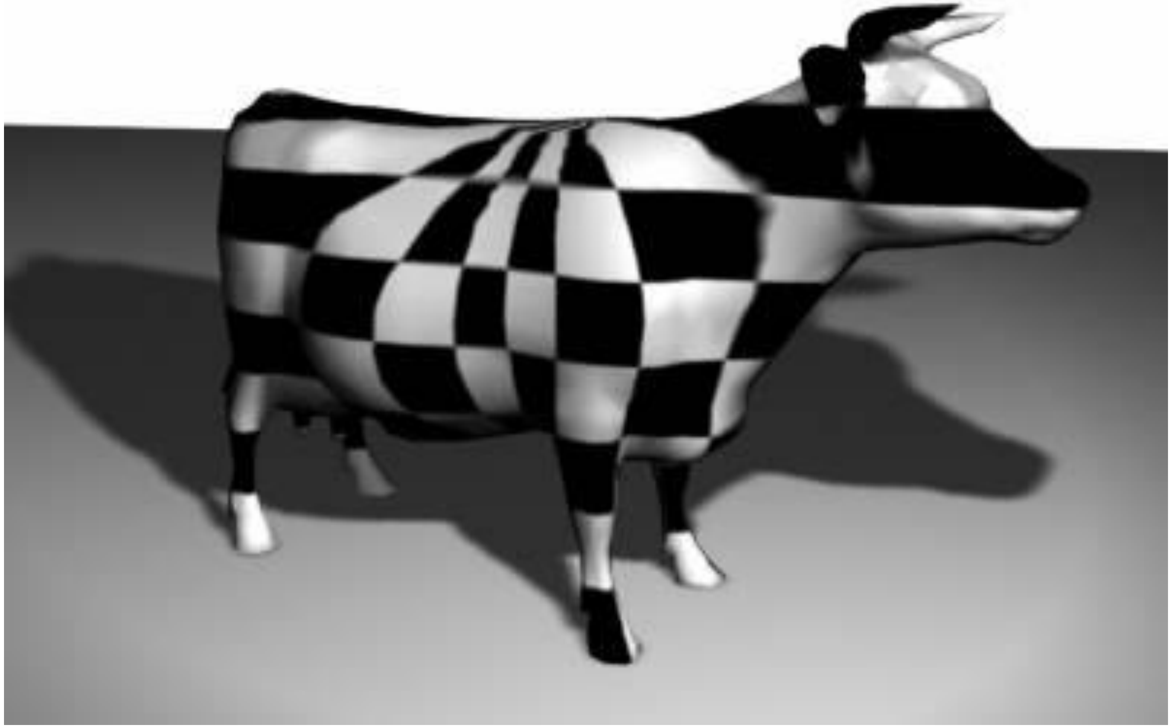
Parametrization

- Manual mapping using CAD Software
  - “Unwrapping”

Images by Martin Kenzel

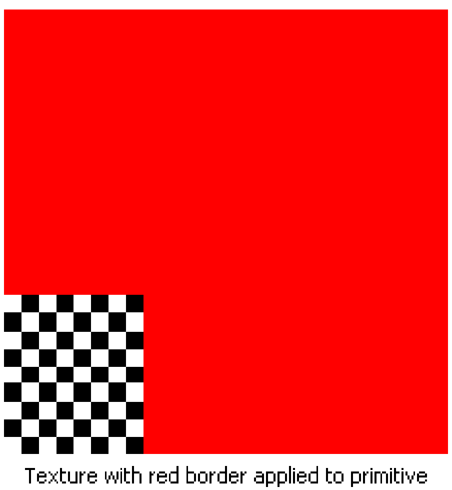
Parameterization problems

- All mappings have distortions and singularities
- Often they need to be fixed manually (CAD software)

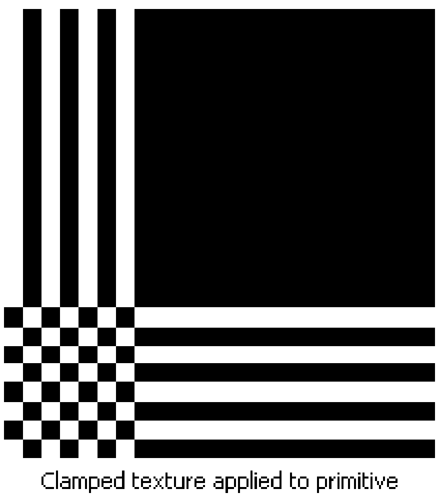


Texture Adressing

- What happens outside [0,1]?
- Border, repeat, clamp, mirror
- Also called texture addressing modes



Static color



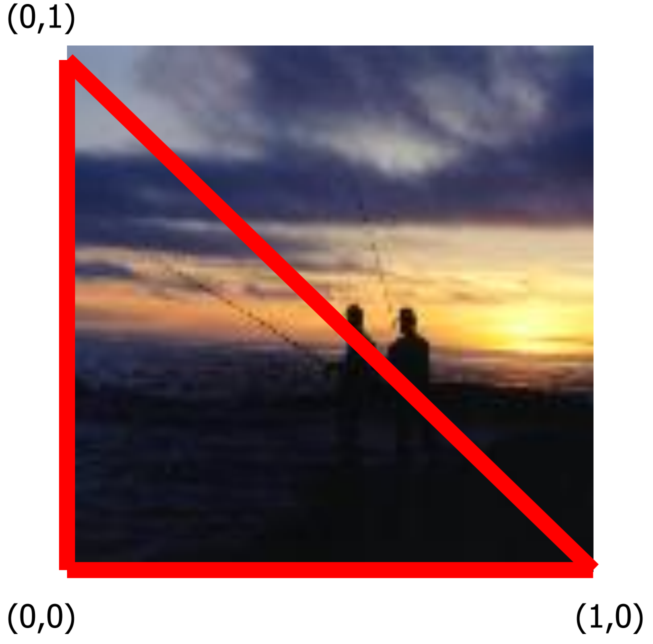
Clamped

Repeated

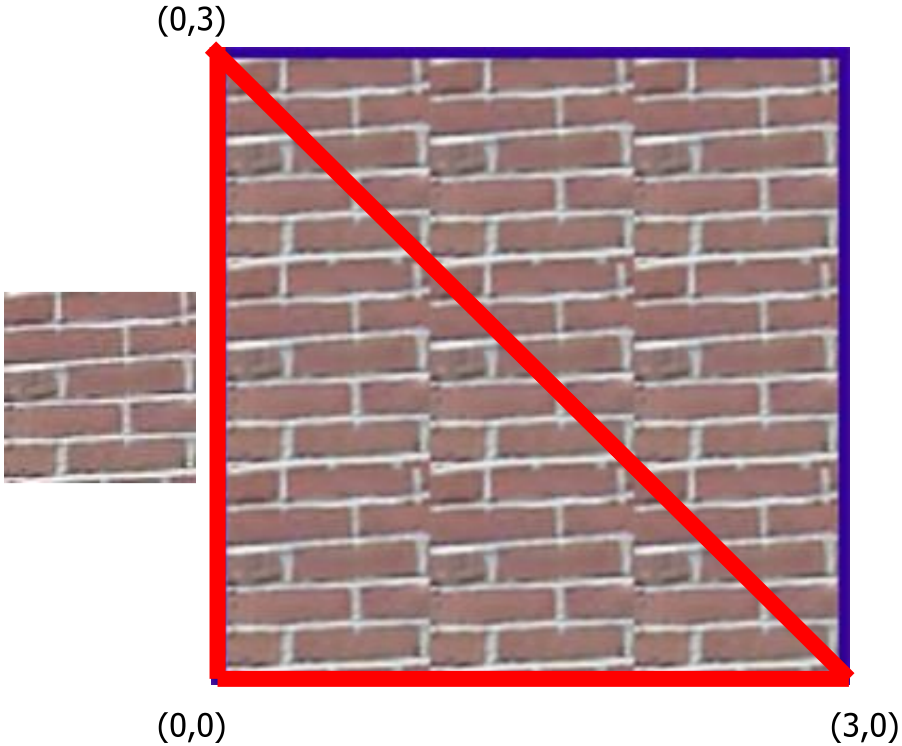
Mirrored

Texture Coordinates

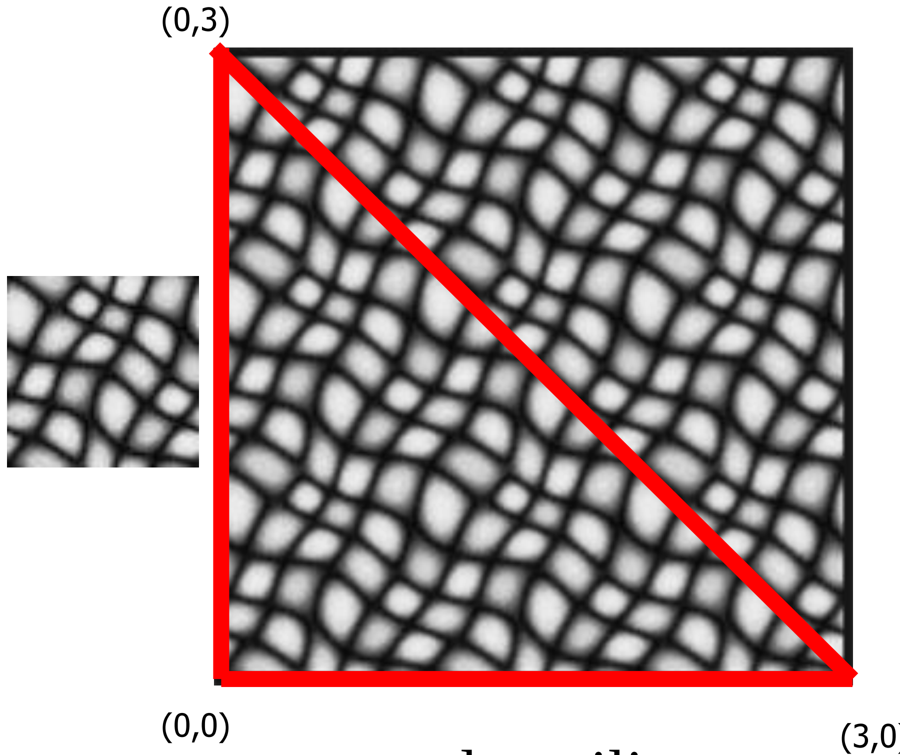
- Specify a texture coordinate at each vertex
- Canonical texture coordinates (0,0) → (1,1)
- Often the texture size is a power of 2
(but it doesn't have to be)
- How can we tile this texture?



Tiling Texture



tiles with
visible seams

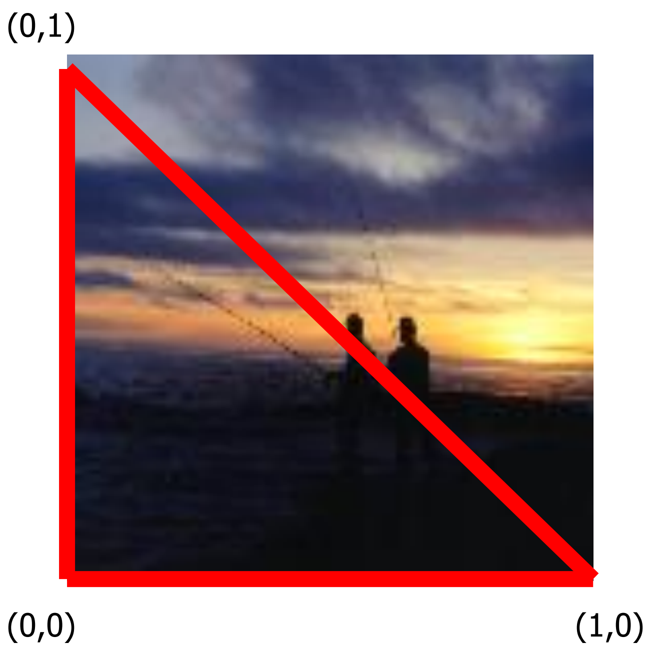


seamless tiling
(repeating)

Texture synthesis

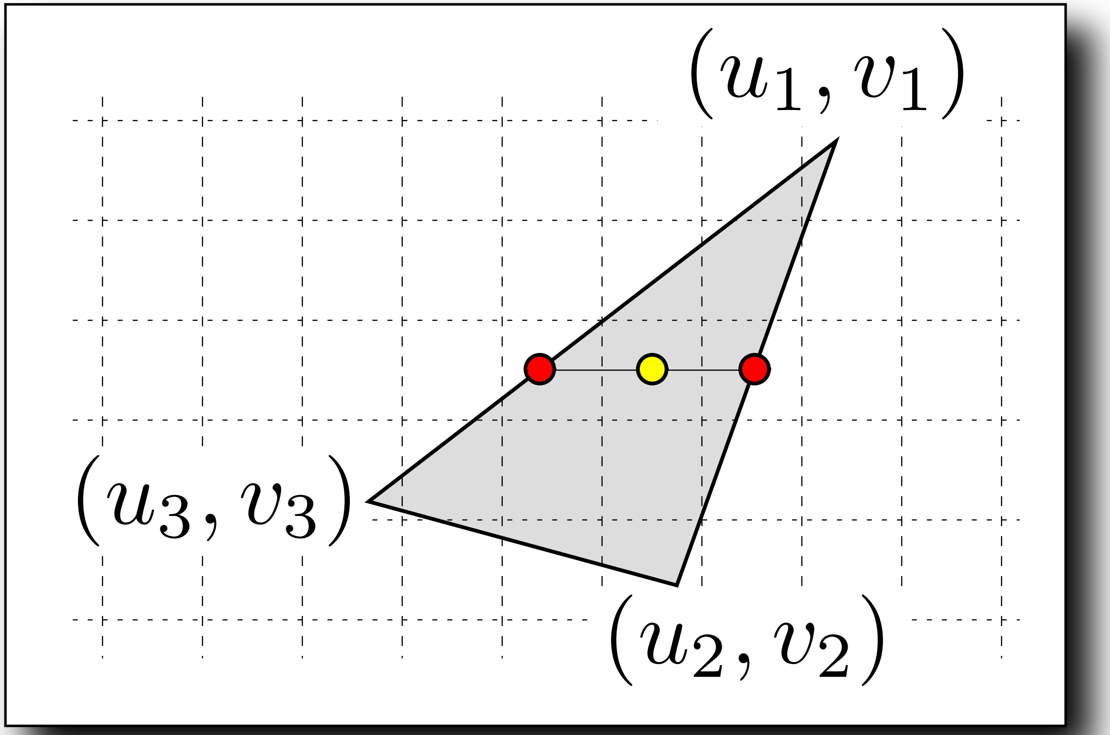
Texture coordinates

- Specify a texture coordinate at each vertex
- Canonical texture coordinates (0,0) → (1,1)
- Linearly interpolate the values in screen space

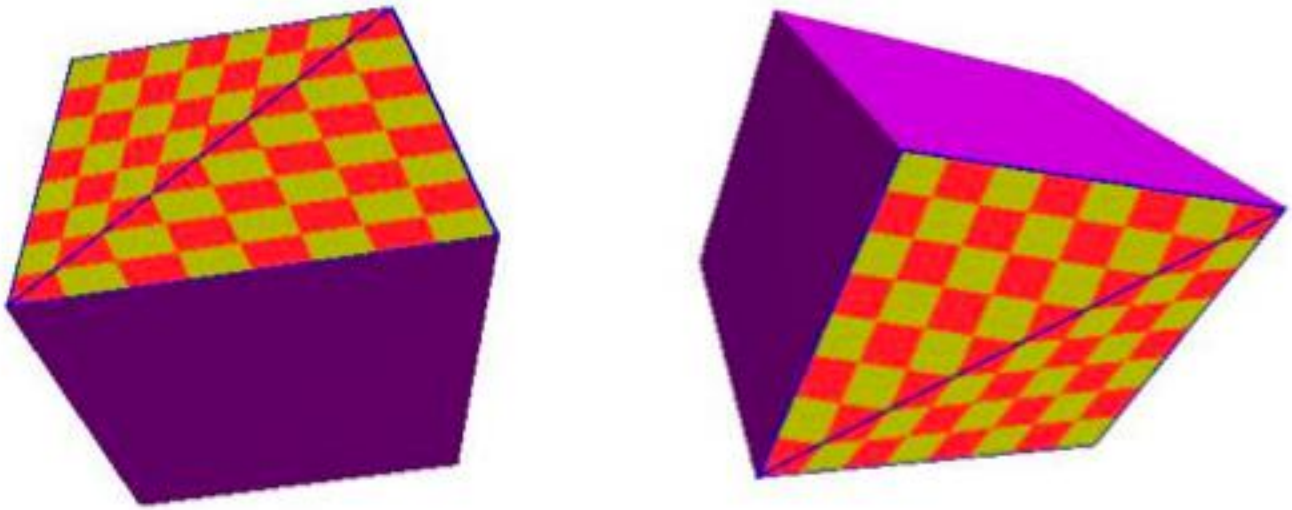


Mapping texture to individual pixels

- Interpolate texture coordinates across scanlines
- Same as Gouraud shading but now for texture coordinates not shading values



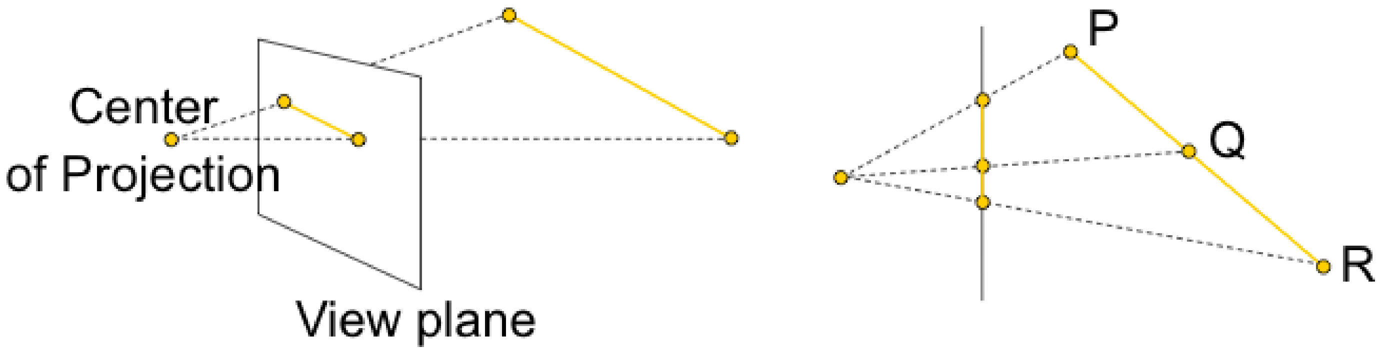
What goes wrong when we linearly interpolate texture coordinates?



- Notice the distortion along the diagonal triangle edge of the cube face after perspective projection

Perspective projection

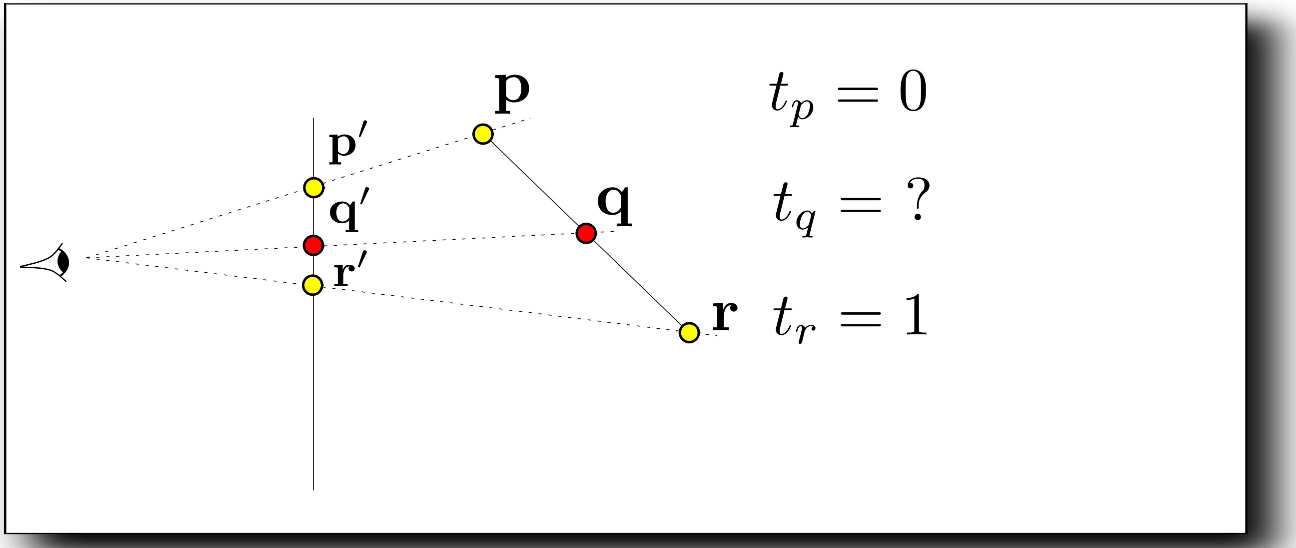
- The problem is that perspective projection does not preserve linear combinations of points!
- In particular; equal distances in 3D space **do not** map to equal distances in screen space



- Linear interpolation in screen space is not the same as linear interpolation in 3D space

How to fix?

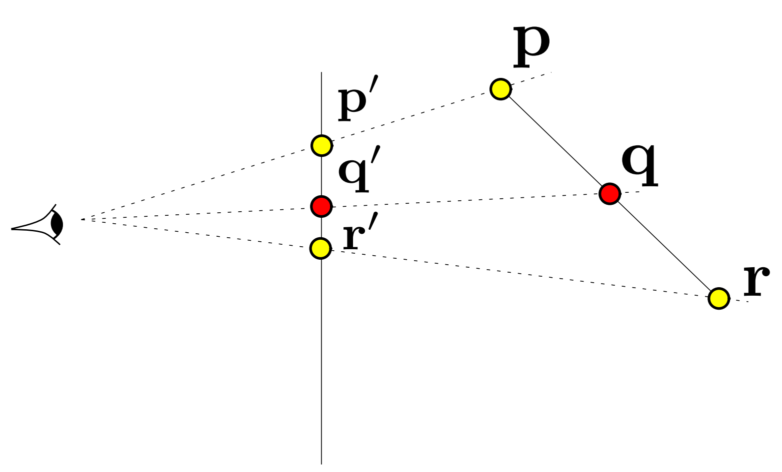
- Assign parameter t to 3-D vertices $\mathbf{p}$ and $\mathbf{r}$
- t controls linear blend of texture coordinates of $\mathbf{p}$ and $\mathbf{r}$
- Let $t = 0$ at $\mathbf{p}$, and $t = 1$ at $\mathbf{r}$
- Assume for simplicity that the image plane is at $z = 1$ *



*i.e. $f = 1$ in the projection matrix

How to fix?

- $\mathbf{p}$ projects to $\mathbf{p'}$ and $\mathbf{r}$ projects to $\mathbf{r'}$ (simply divide by z coordinate)
- What value should t have at location $\mathbf{q'}$?



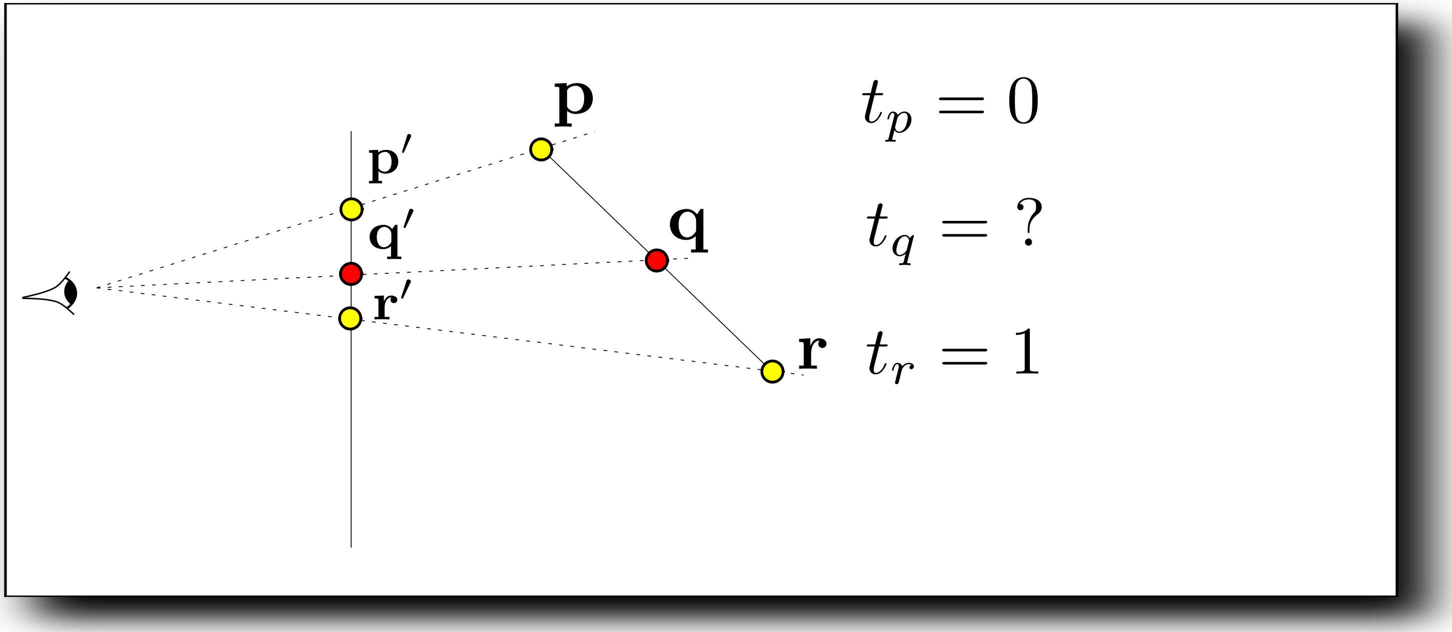
$$\mathbf{p}' = \frac{\mathbf{p}}{z_p}$$

$$\mathbf{q}' = \frac{\mathbf{q}}{z_q}$$

$$\mathbf{r}' = \frac{\mathbf{r}}{z_r}$$

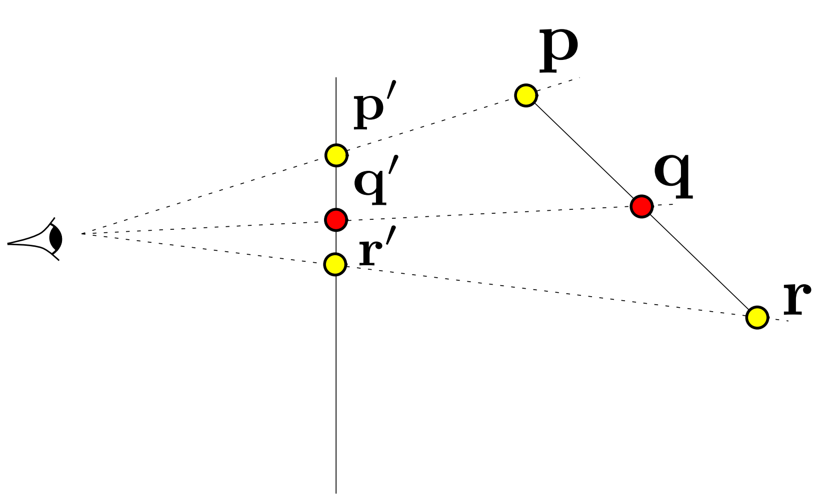
How to fix?

- We cannot linearly interpolate t between $\mathbf{p'}$ and $\mathbf{r'}$
- Only projected values can be linearly interpolated in screen space
- Solution: perspective-correct interpolation



Perspective Correct Interpolation

- Linearly interpolate t / z (not t) between **p'** and **r'**.
  - Compute $t_{\mathbf{p}'} = t_{\mathbf{p}} / z_{\mathbf{p}}$ $\quad t_{\mathbf{r}'} = t_{\mathbf{r}} / z_{\mathbf{r}}$
  - Linearly interpolate (lerp) $t_{\mathbf{p}'}$ and $t_{\mathbf{r}'}$ to get $t_{\mathbf{q}'}$ at location **q**'
- But, we want the un-projected parameter $t_{\mathbf{q}}$ (not $t_{\mathbf{q}'}$)



$$t_{p'} = \frac{t_p}{z_p}$$

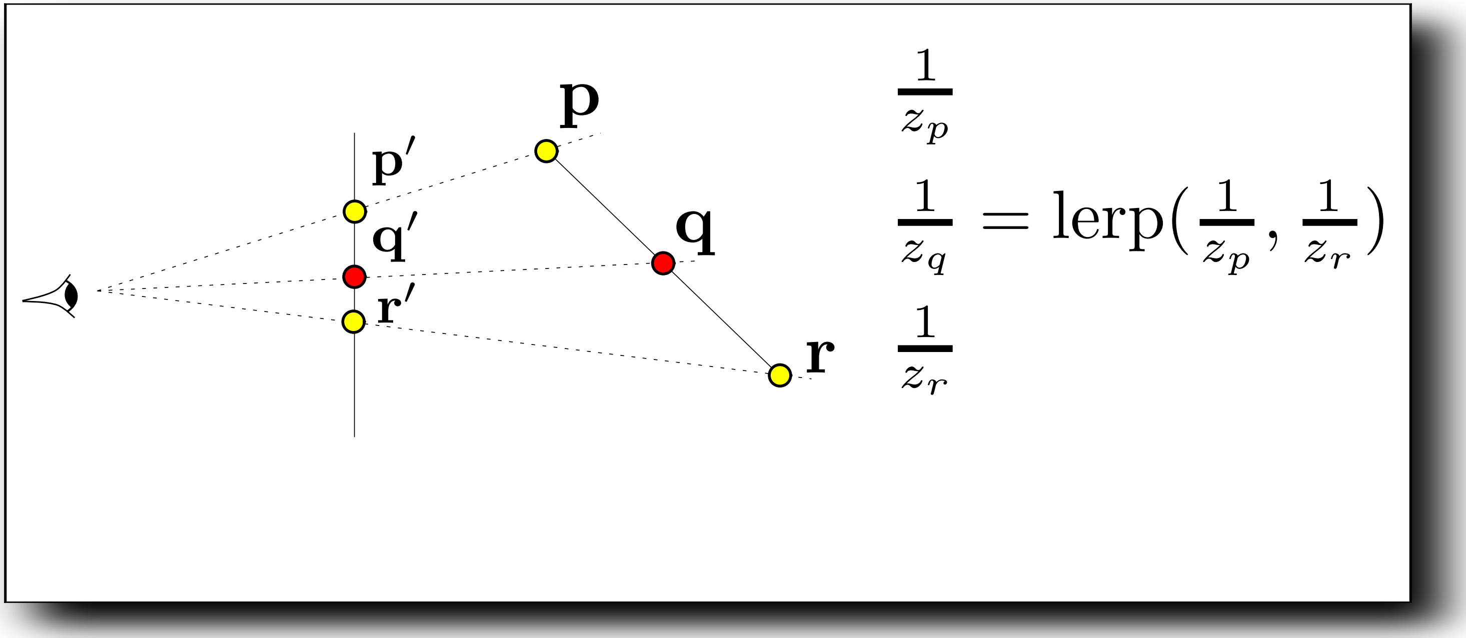
$$t_{q'} = \mathrm{lerp}(t_{p'}, t_{r'})$$

$$t_{r'} = \frac{t_r}{z_r}$$

$$t_q = ?$$

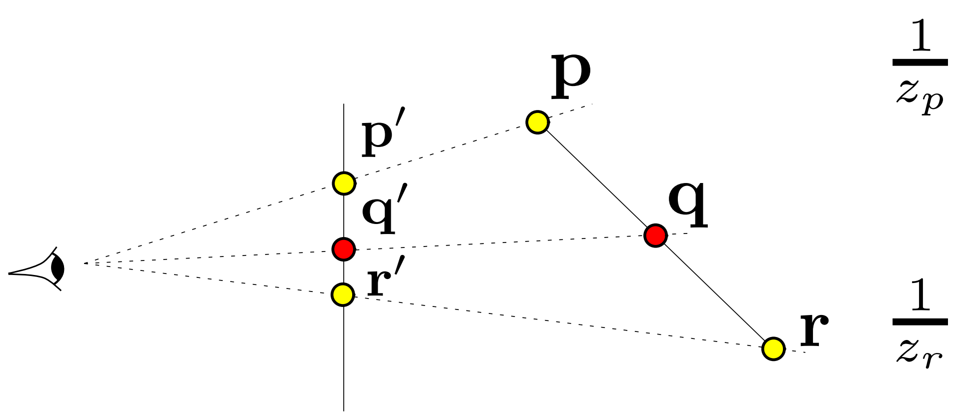
Perspective Correct Interpolation

- The parameters $t_{\mathbf{p}}$, & $t_{\mathbf{q}}$, are related to $t_{\mathbf{p}}$ & $t_{\mathbf{q}}$ by perspective factors of $1/z_{\mathbf{p}}$ and $1/z_{\mathbf{q}}$
  - lerp $1 / z_{\mathbf{p}}$ and $1 / z_{\mathbf{r}}$ to obtain $1 / z_{\mathbf{q}}$ at point $\mathbf{q}$ '



Perspective Correct Interpolation

- The parameters $t_{\mathbf{p}'}$ & $t_{\mathbf{q}'}$ are related to $t_{\mathbf{p}}$ & $t_{\mathbf{q}}$ by perspective factors of $1/z_{\mathbf{p}}$ and $1/z_{\mathbf{q}}$
  - lerp $1 / z_{\mathbf{p}}$ and $1 / z_{\mathbf{r}}$ to obtain $1 / z_{\mathbf{q}}$ at point $\mathbf{q}'$
  - Divide $t_{\mathbf{q}'}$ by $1 / z_{\mathbf{q}}$ to get $t_{\mathbf{q}}$



$$t_{\mathbf{q}} = t_{\mathbf{q}'} \times z_{\mathbf{q}} = \mathrm{lerp}(t_{\mathbf{p}'}, t_{\mathbf{r}'}) \div \mathrm{lerp}(\frac{1}{z_{\mathbf{p}}}, \frac{1}{z_{\mathbf{r}}})$$

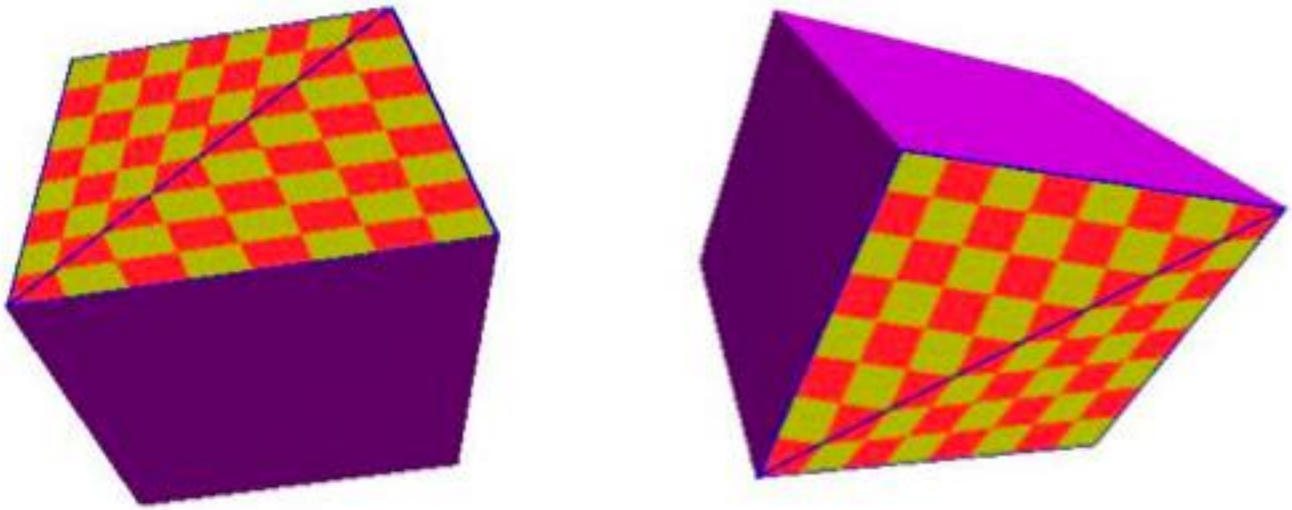
Perspective Correct Interpolation

- Summary:
  - Given texture parameter t at vertices:
    - Compute $1 / z$ for each vertex
    - Linearly interpolate $1 / z$ across the triangle
    - Linearly interpolate t / z across the triangle
    - Do perspective division:

      Divide t / z by $1 / z$ to obtain interpolated parameter t

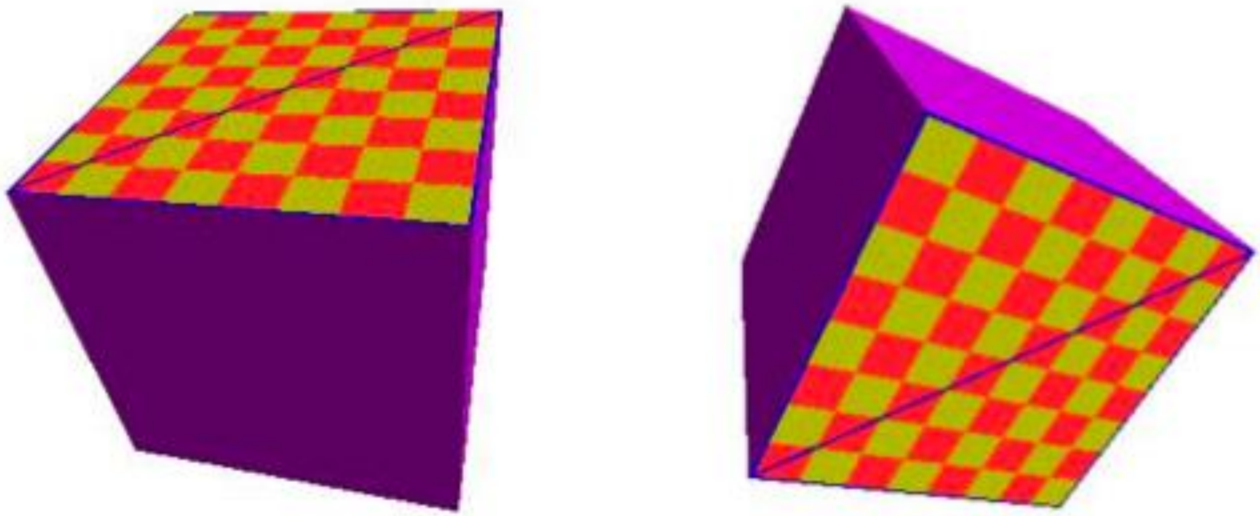
$$\boxed{t_{\mathbf{q}} = \frac{\operatorname{lerp}\left(\frac{t_{\mathbf{p}}}{z_{\mathbf{p}}}, \frac{t_{\mathbf{r}}}{z_{\mathbf{r}}}\right)}{\operatorname{lerp}\left(\frac{1}{z_{\mathbf{p}}}, \frac{1}{z_{\mathbf{r}}}\right)}}$$

What Goes Wrong?



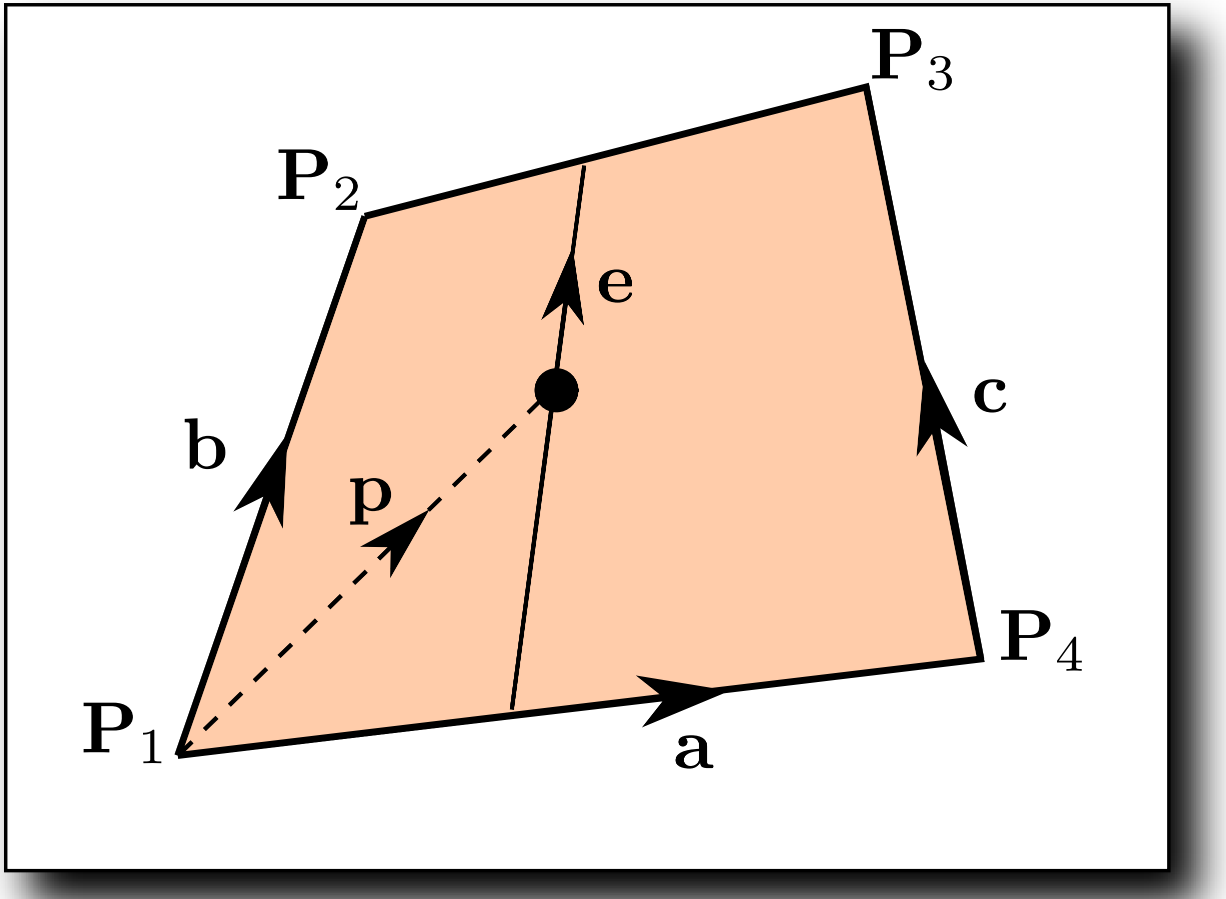
- Notice the distortion along the diagonal triangle edge of the cube face

Perspective Correct Interpolation



Mapping texture to individual pixels

Alternative



Bilinear Texture mapping

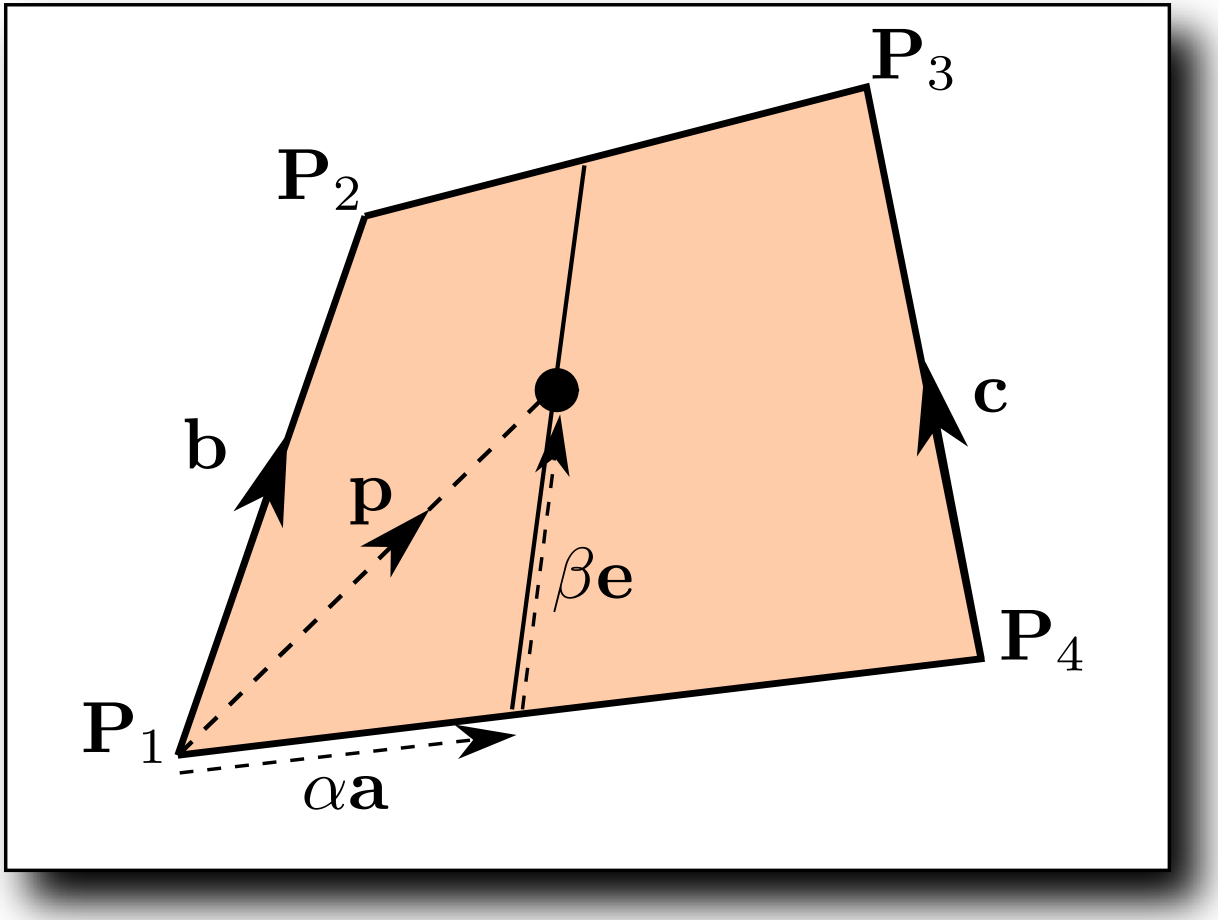
$\mathbf{P}_{1..4}$: Polygon vertices

$\mathbf{p}$: Pixel to be textured

Bi-linear Map - Solving for a and b

$\mathbf{p} = \alpha\,\mathbf{a} + \beta\,\mathbf{e}$

$\mathbf{e} = \mathbf{b} + \alpha\,(\mathbf{c} - \mathbf{b})$



so

$\mathbf{p} = \alpha\,\mathbf{a} + \beta\,\mathbf{b} + \alpha\beta\,(\mathbf{c} - \mathbf{b})$ Quadratic in the unknowns !

Non Linearities in texture mapping

- The second order term means that straight lines in the texture may become curved when the texture is mapped.
- However, if the mapping is to a parallelogram:

$$\mathbf{p} = \alpha\,\mathbf{a} + \beta\,\mathbf{b} + \alpha\beta\,(\mathbf{c} - \mathbf{b})$$

and

$$\mathbf{b} = \mathbf{c}$$

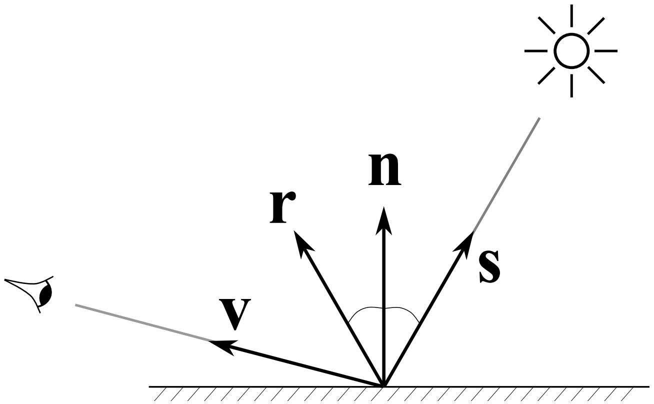
so

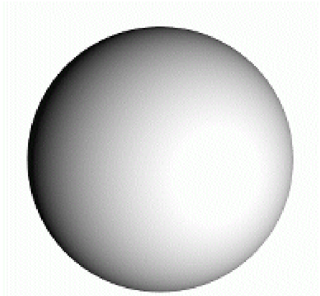
$$\mathbf{p} = \alpha\,\mathbf{a} + \beta\,\mathbf{b}$$

Texture Mapping & Illumination

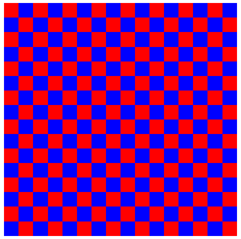
- Texture mapping can be used to alter parts of the illumination equation

$$L(\omega_r) = k_a I_a + \left(k_d I_d (\mathbf{n}\cdot\mathbf{s}) + k_s I_s (\mathbf{v}\cdot\mathbf{r})^q\right)$$

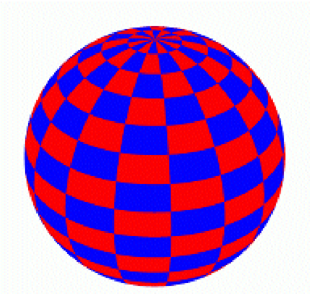




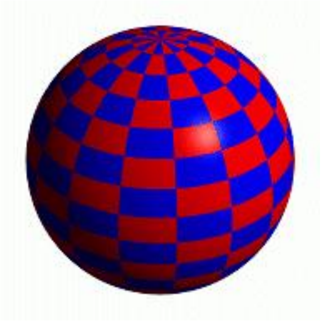
Constant Diffuse Color



Texture Image



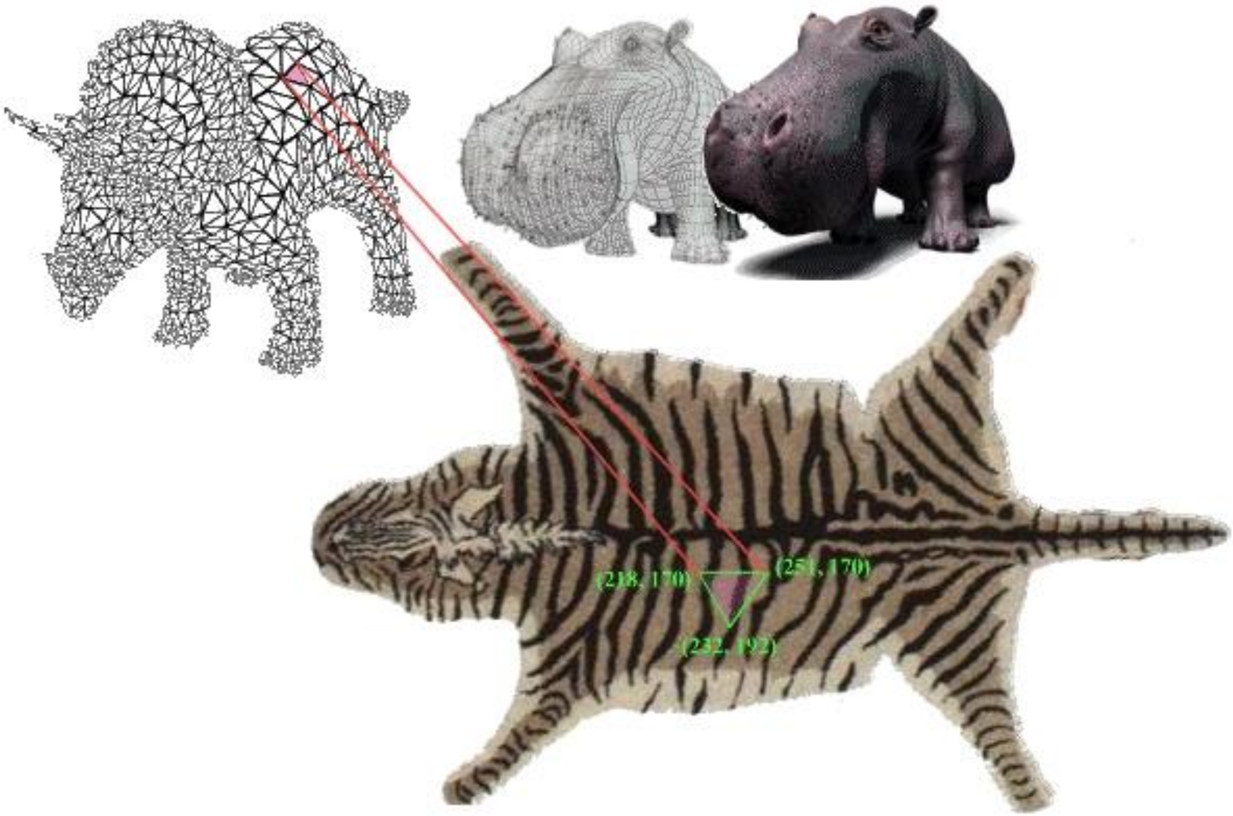
Texture used as Label



Texture used as Diffuse Color

2D Texture Mapping

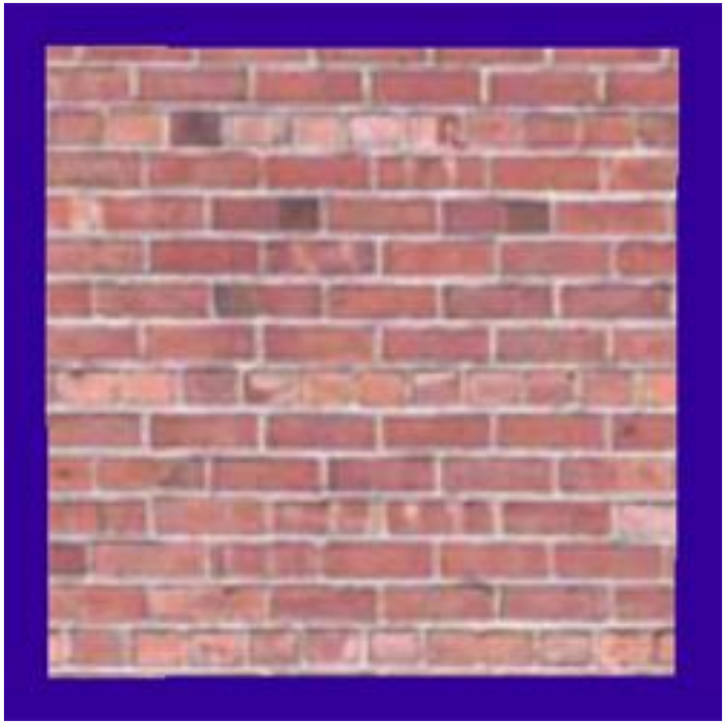
- Increases the apparent complexity of simple geometry
- Requires perspective projection correction
- Can specify variations in shading within a primitive:
  - Illumination
  - Surface Reflectance

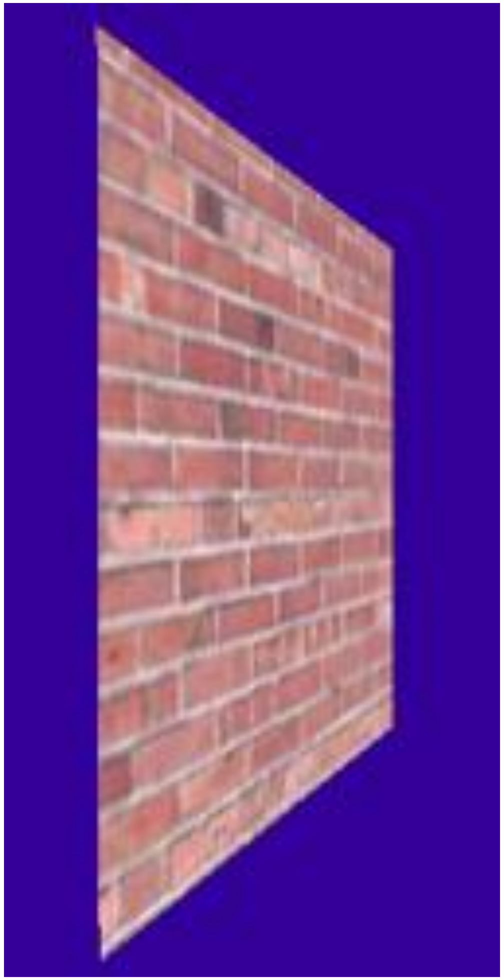


What's Missing?

- What's the difference between a real brick wall and a photograph of the wall texture-mapped onto a plane?

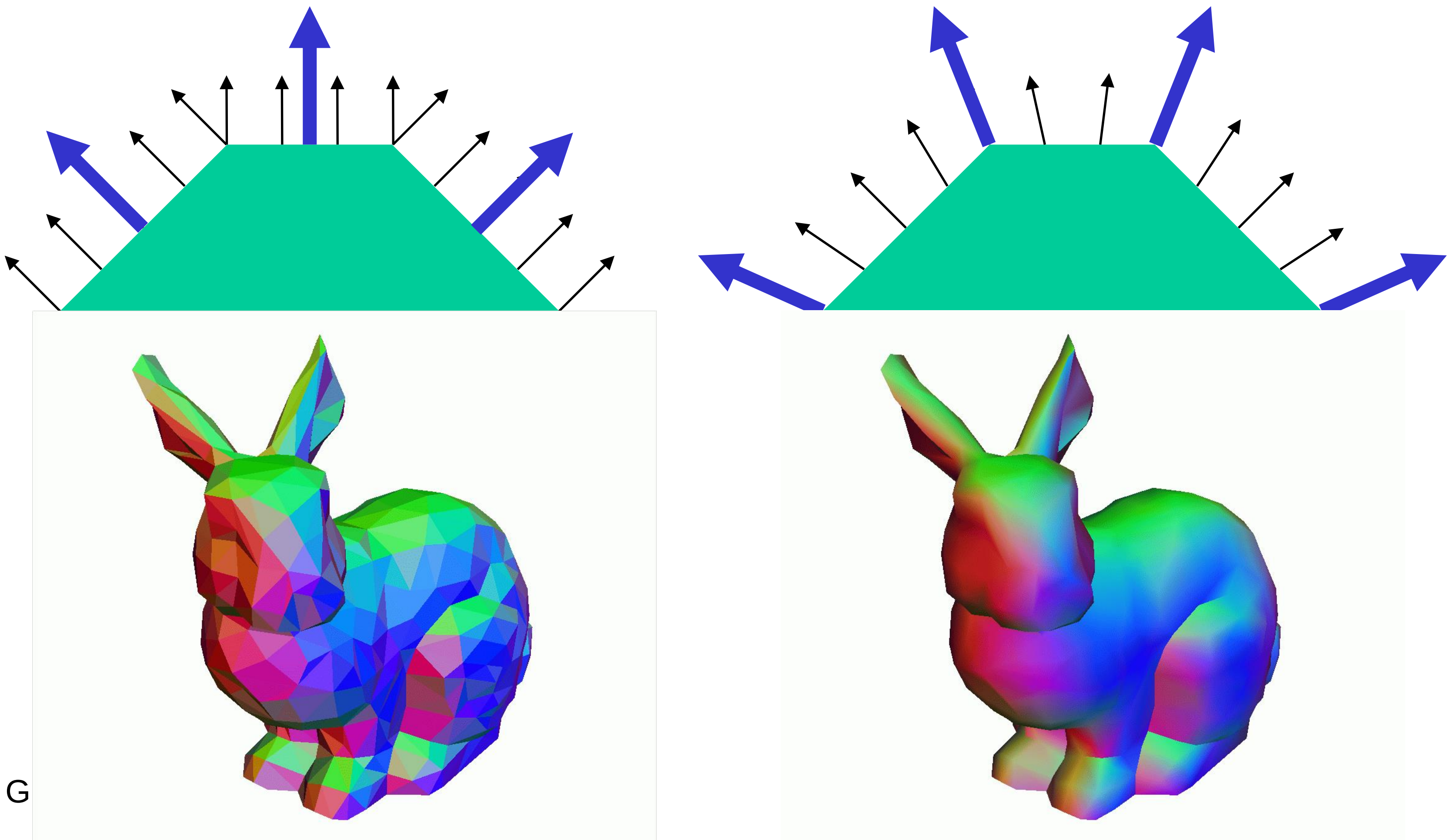
- What happens if we change the lighting or the camera position?





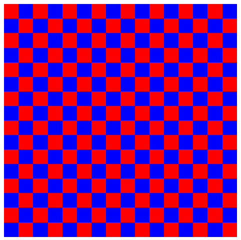
Remember Normal Averaging for Shading?

- Instead of using the normal of the triangle, interpolate an averaged normal at each vertex across the face

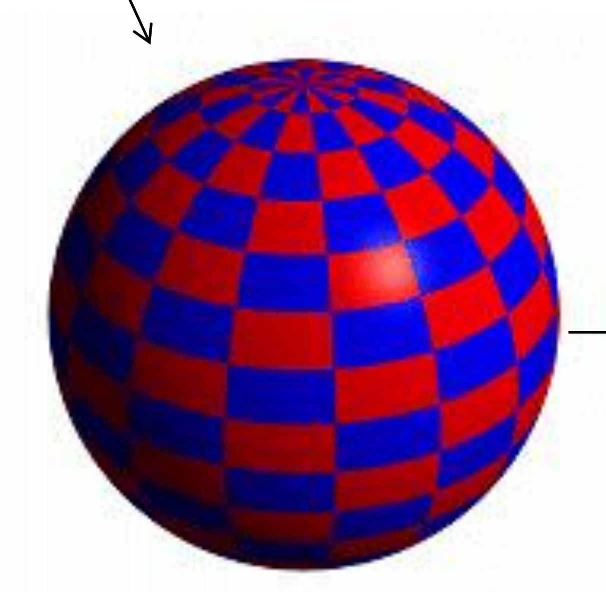


G

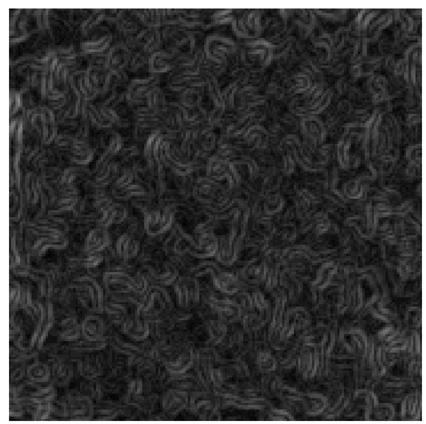
Bump Mapping



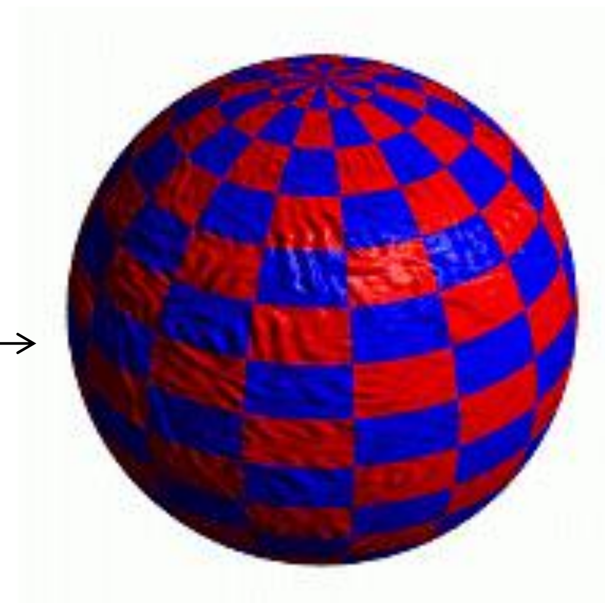
- Textures can be used to alter the surface normal of an object.
- Does not change actual shape of the surface – we only shade it as if it were a different shape!



Sphere w/Diffuse Texture



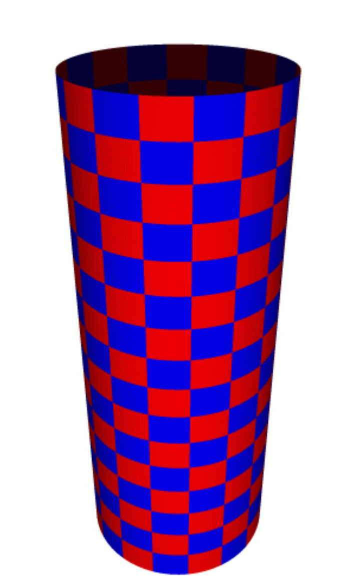
Swirly Bump Map



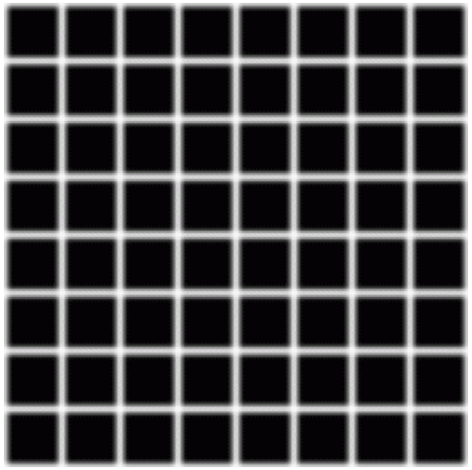
Sphere w/Diffuse Texture & Bump Map

Bump Mapping

- The texture map is treated as a single-valued height function.
- The partial derivatives of the texture tell us how to alter the true surface normal at each point to make the object appear as if it were deformed by the height function.



Cylinder w/Diffuse Texture Map



Bump Map

Cylinder w/Texture Map & Bump Map

Another Bump Map Example

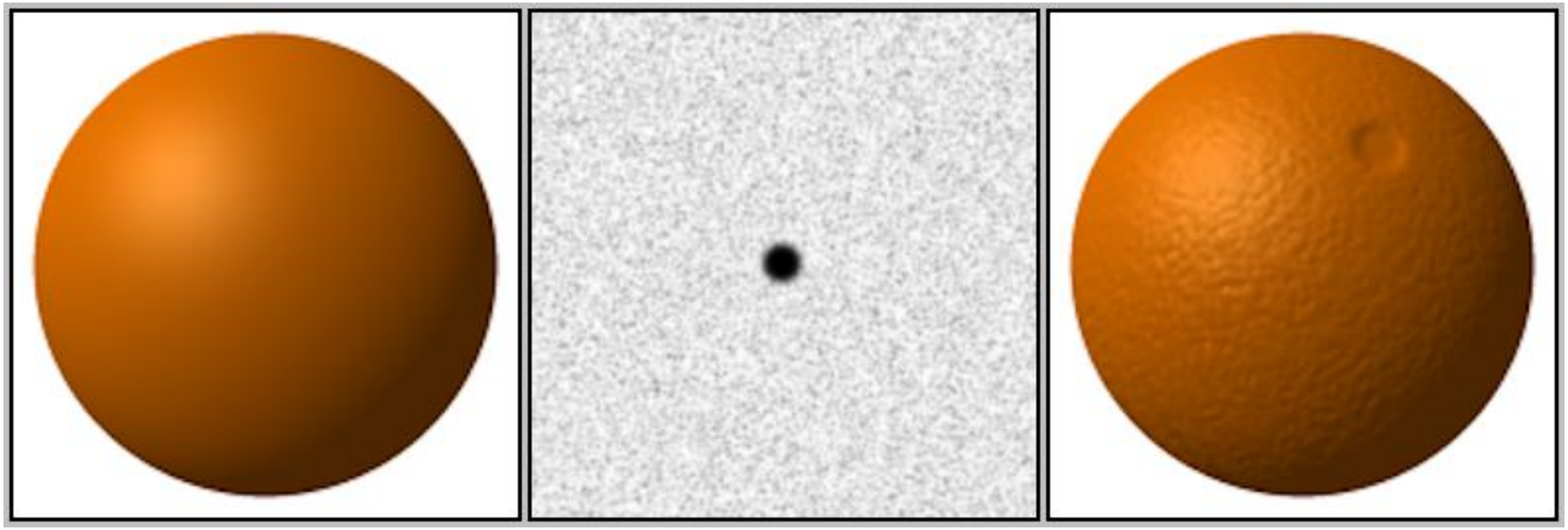
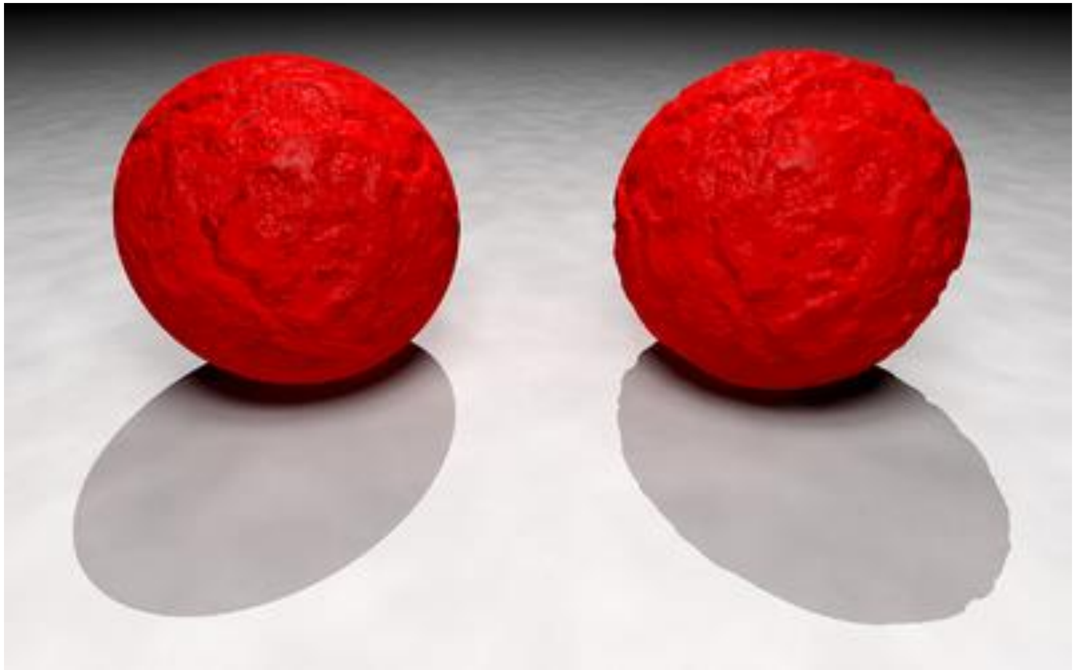


Image source: Wikipedia, 2016

What's Missing?

- What does a texture- & bump-mapped object look like as you move the viewpoint?

- What does the silhouette of a bump-mapped object look like?

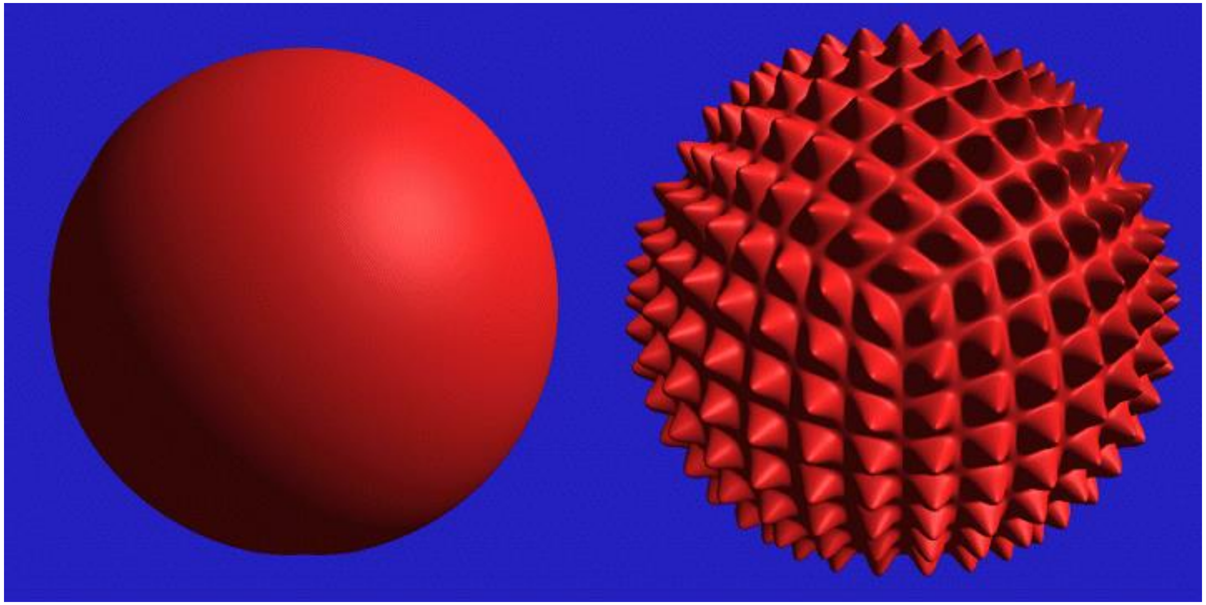


https://threejs.org/examples/webgl_materials_bumpmap.html

Image source: Wikipedia, 2016

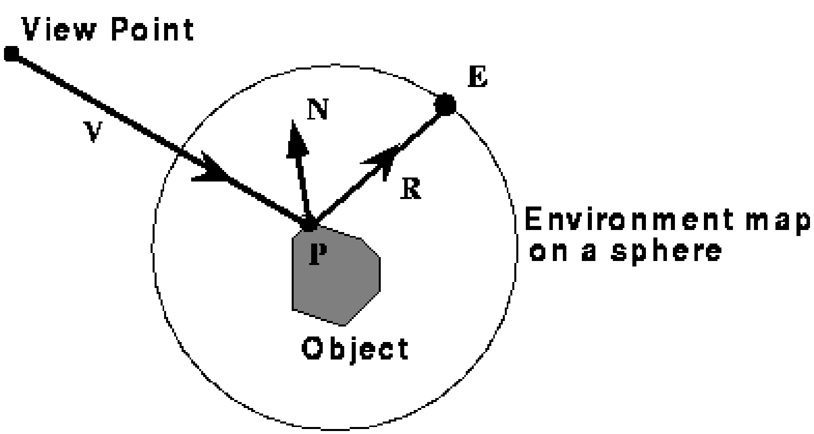
Displacement Mapping

- Use the texture map to actually move the surface point.
  - How is this different than bump mapping?
- The geometry must be displaced before visibility is determined.



Environment Maps

- We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".
- Assumes that all reflected rays begin from the same point.



Environment Mapping Example

https://threejs.org/examples/webgl_materials_cubemap.html

Environment Mapping Example