Textures

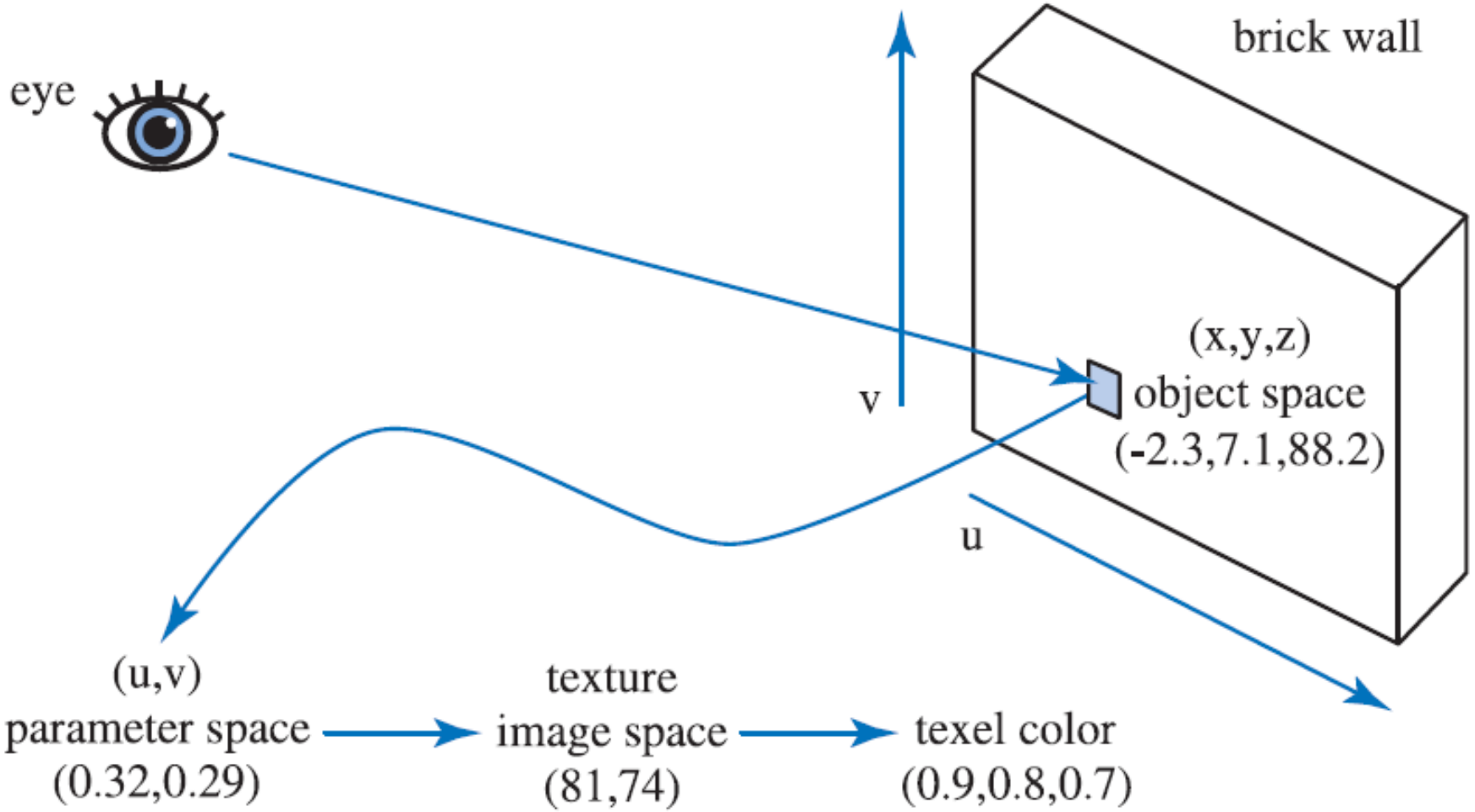
CS 4620 Lecture 20

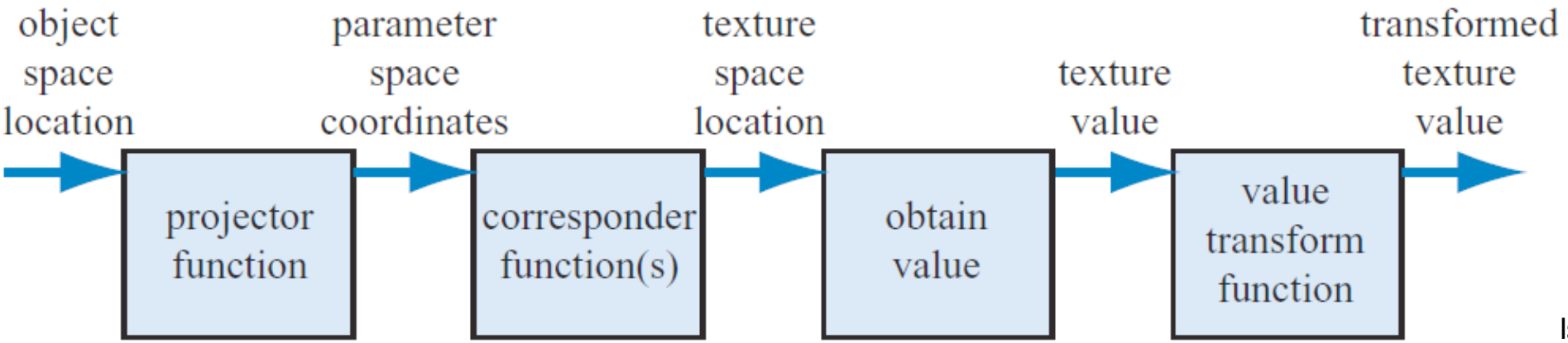
Announcements

- A4 out

- Prelim review
  - Monday, 7-9, Maybe G01 Gates

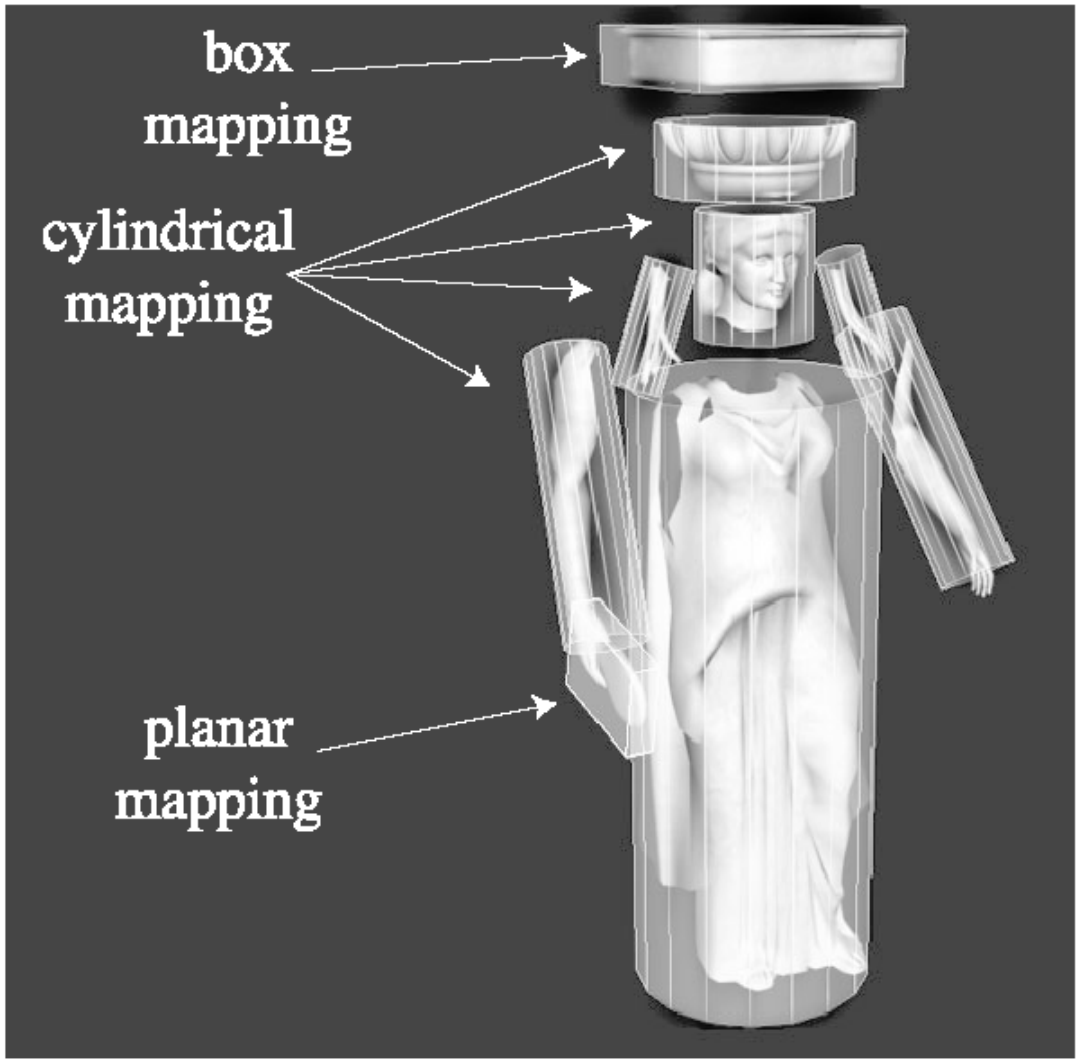
- Prelim next week
  - Oct 20th Tuesday 2015, 7:30, Olin Hall 155
  - Prelim makeups: 9am on Tuesday


eye
brick wall
v
(x,y,z)
object space
(-2.3,7.1,88.2)
u
(u,v)
parameter space
(0.32,0.29)
texture
image space
(81,74)
texel color
(0.9,0.8,0.7)


object space location
parameter space coordinates
texture space location
texture value
transformed texture value
projector function
corresponder function(s)
obtain value
value transform function

Projector Function: Arbitrary Surfaces

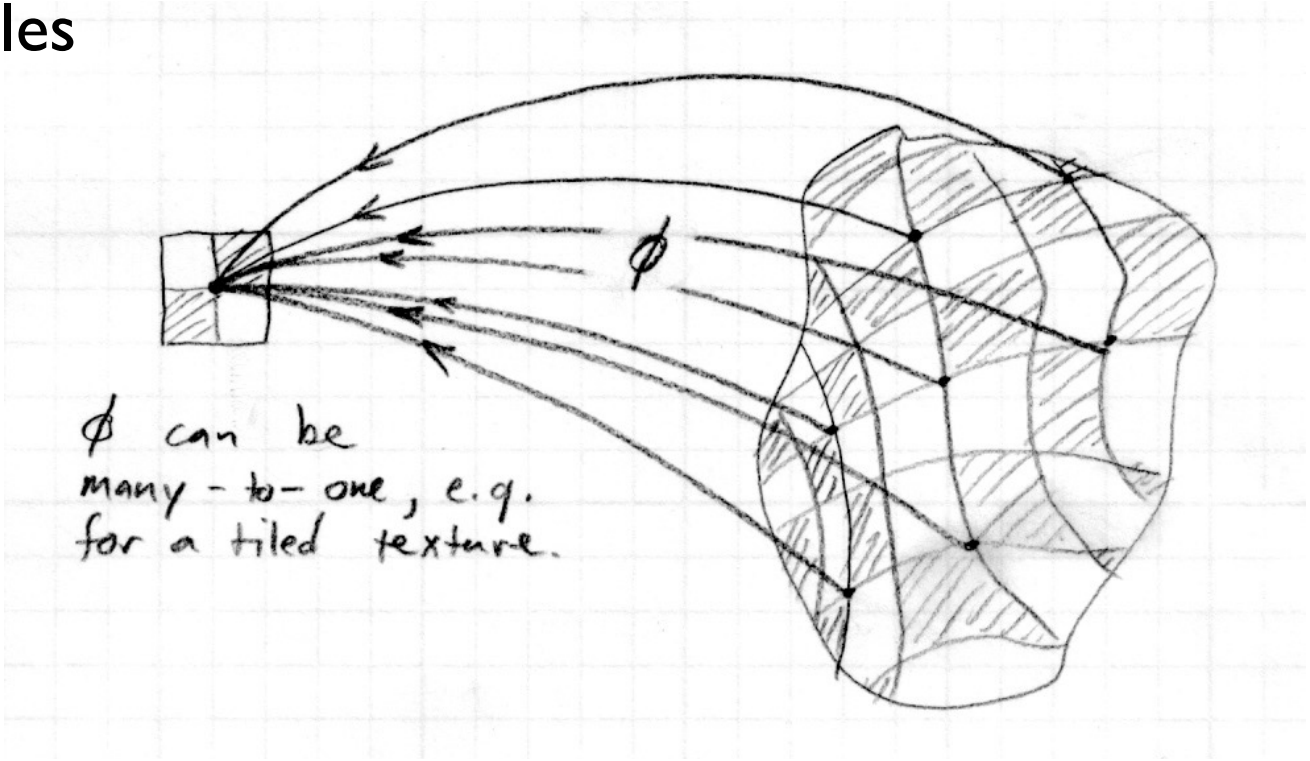
- Non-parametric surfaces: project to parametric surface



Images courtesy Tito Pagan

Corresponder functions

- Mapping from *S* to *D* can be many-to-one
  - that is, every surface point gets only one color assigned
  - but it is OK (and in fact useful) for multiple surface points to be mapped to the same texture point
    - e.g. repeating tiles



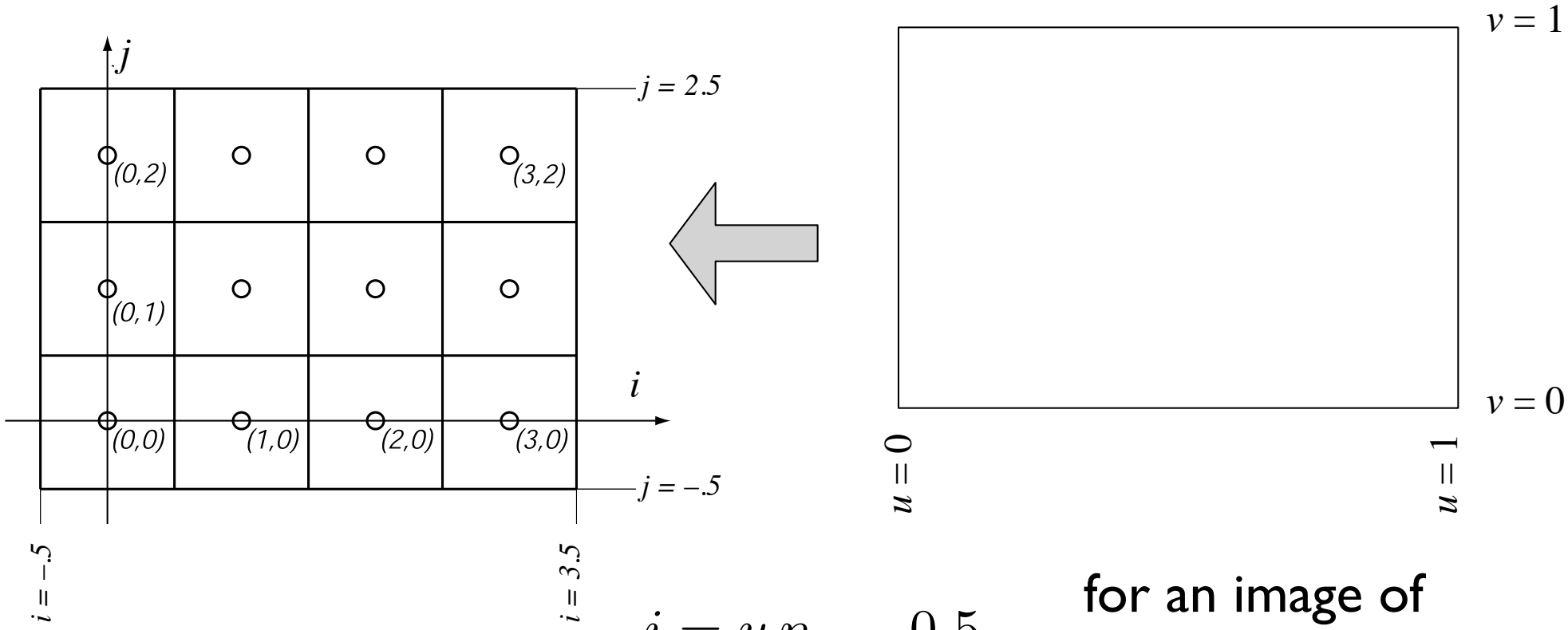
Corresponder Function

- Why?
  - Flexibility

- Examples:
  - Select a subset of the image for texturing
  - Tile textures
  - Decide what happens at boundaries

Pixels in texture images (texels)

- Related to texture coordinates in the same way as normalized image coordinates to pixel coordinates



$$i = u\,n_x - 0.5$$

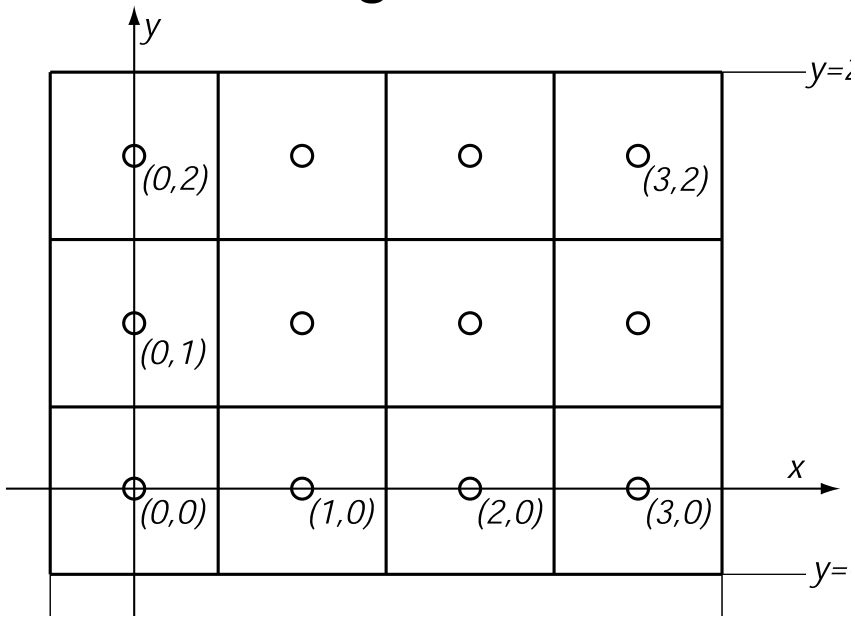
$$j = v\,n_y - 0.5$$

for an image of n_x by n_y pixels

Texture lookups and wrapping

- In shading calculation, when you need a texture value you perform a *texture lookup*

- Convert (*u*, *v*) texture coordinates to (*i*, *j*) texel coordinates, and read a value from the image
  - simplest: round to nearest (nearest neighbor lookup)
  - various ways to be smarter and get smoother results



Texture lookups and wrapping

- What if *i* and *j* are out of range?
  - option 1, clamp: take the nearest pixel that is in the image

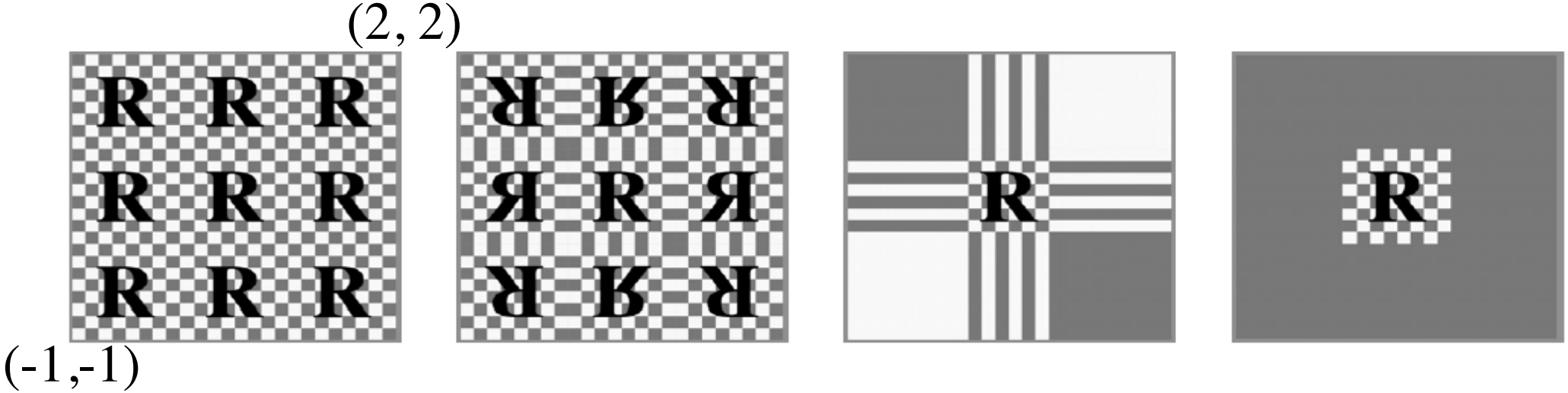
$$i_{\text{pixel}} = \max(0, \min(n_x - 1, i_{\text{lookup}}))$$

  - option 2, wrap: treat the texture as periodic, so that falling off the right side causes the look up to come in the left

$$i_{\text{pixel}} = \text{remainder}(i_{\text{lookup}}, n_x)$$

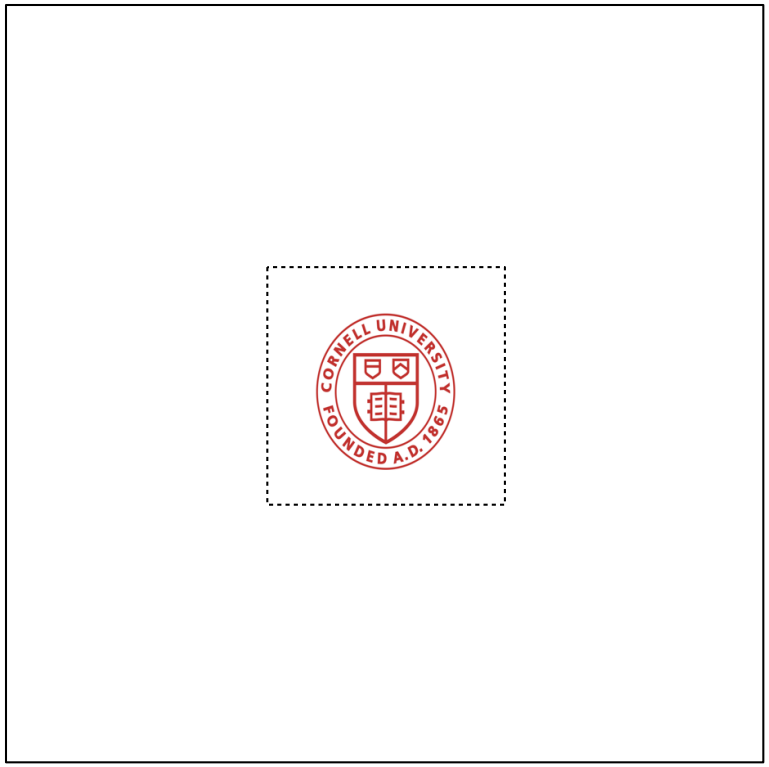
Corresponder Function

- In OpenGL: wrapping mode

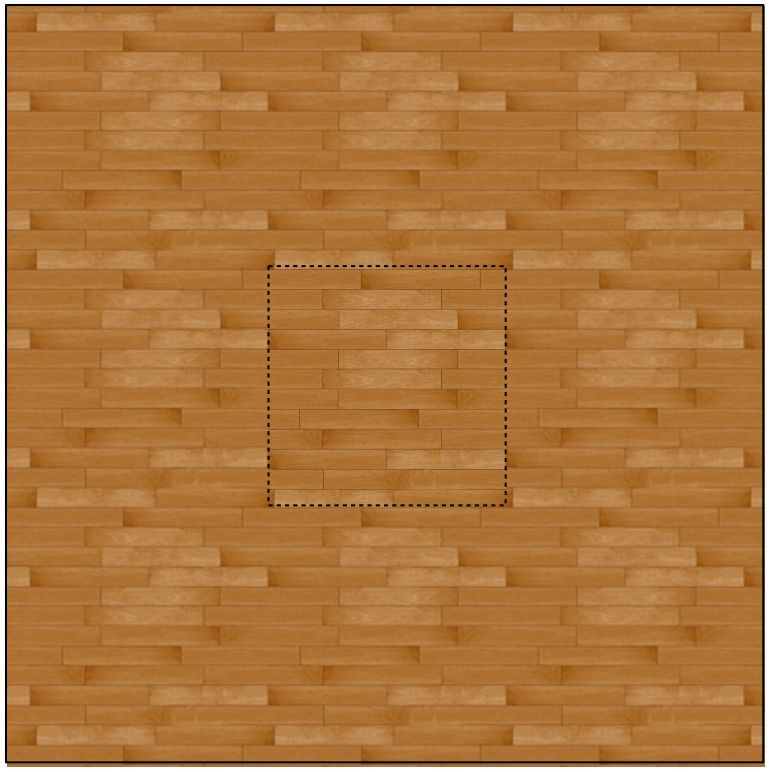


- Wrap: Repeats
- Mirror
  - Repeats but mirrored every other time; continuity across edges
- Clamp: Clamped to edge of texture
- Border: Clamped to border color

Wrapping modes



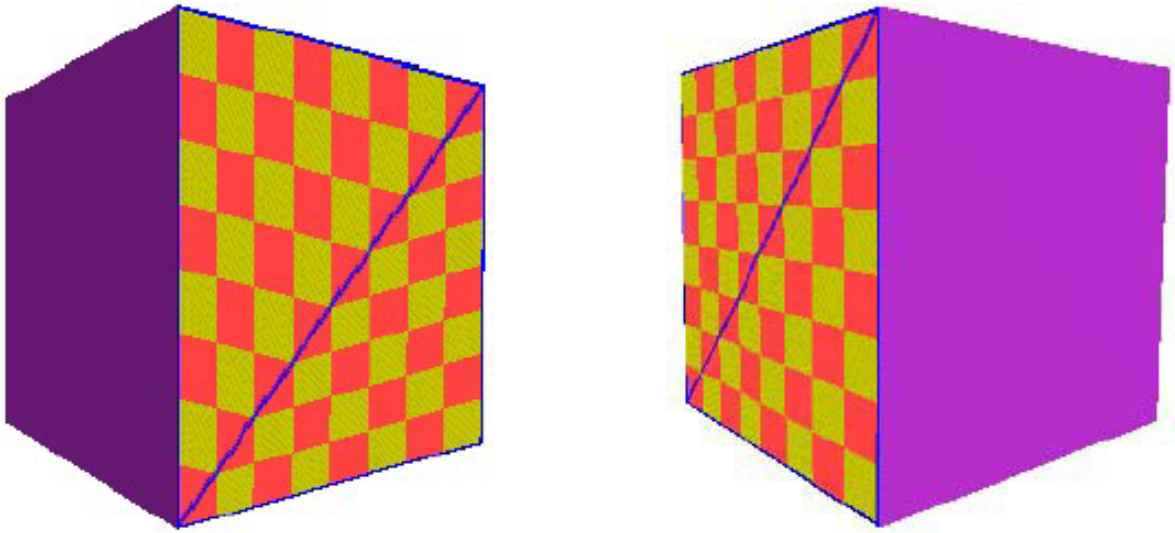
clamp

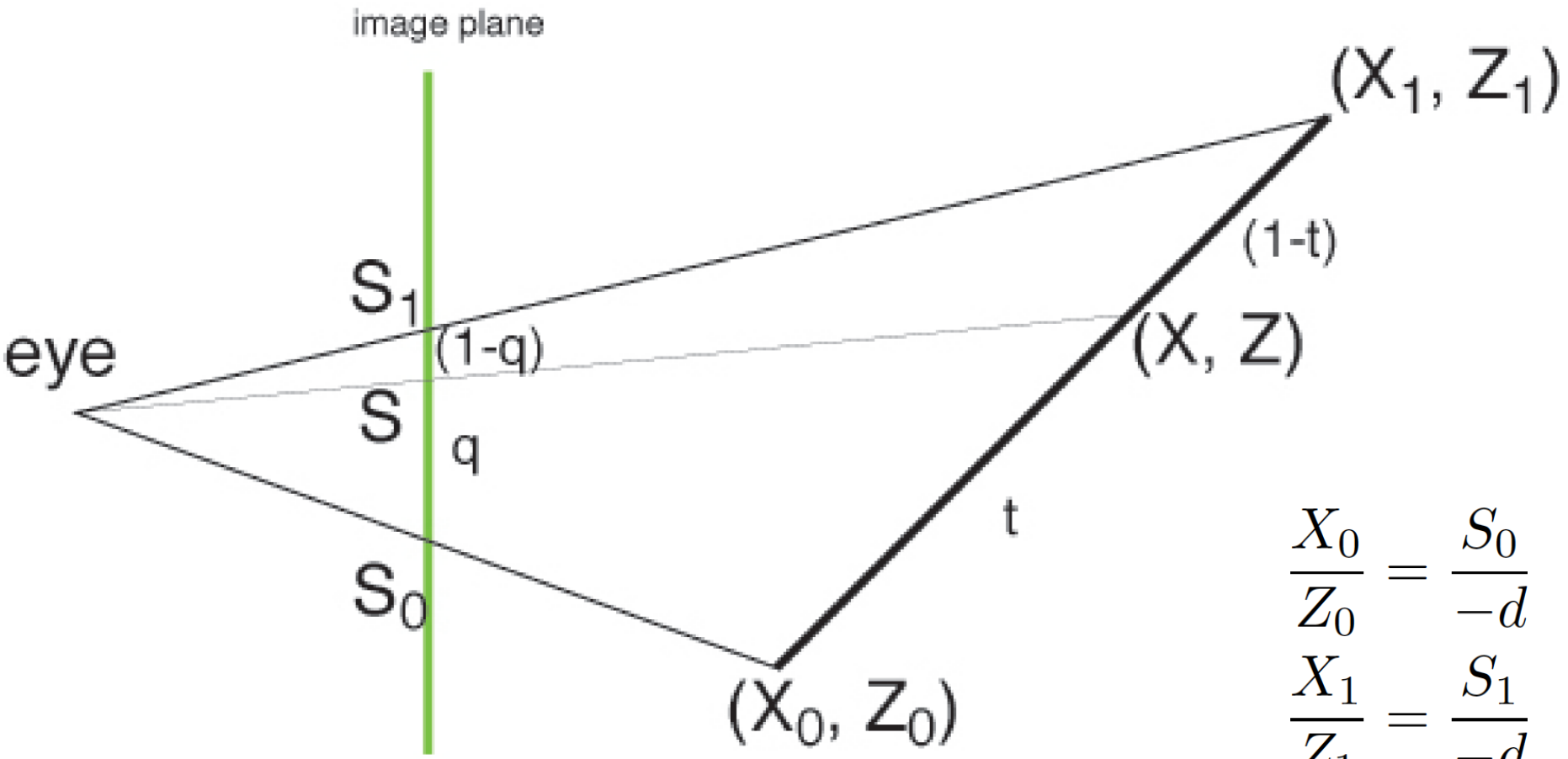


wrap

Perspective-Correct Texturing

- In hardware rendering
  - Must be careful to interpolate texture coordinates correctly





$$\frac{X_0}{Z_0} = \frac{S_0}{-d}$$

$$\frac{X_1}{Z_1} = \frac{S_1}{-d}$$

$$S = S_0 + q(S_1 - S_0)$$

$$[X, Z] = [X_0 + t(X_1 - X_0), Z_0 + t(Z_1 - Z_0)]$$

$$
\begin{aligned}
Z &= \frac{-dX}{S} \\
&= \frac{-d(X_0 + t(X_1 - X_0))}{S_0 + q(S_1 - S_0)}
\end{aligned}
\qquad
\begin{aligned}
Z &= \frac{-d(\frac{S_0 Z_0}{-d} + t\frac{(S_1 Z_1 - S_0 Z_0)}{-d}}{S_0 + q(S_1 - S_0)} \\
&= \frac{S_0 Z_0 + t(S_1 Z_1 - S_0 Z_0)}{S_0 + q(S_1 - S_0)}
\end{aligned}
$$

$$
Z_0 + t(Z_1 - Z_0) = \frac{S_0 Z_0 + t(S_1 Z_1 - S_0 Z_0)}{S_0 + q(S_1 - S_0)}
$$

$$
t = \frac{Z_0 q}{q Z_0 + (1 - q) Z_1}
$$

$$
\begin{aligned}
Z &= Z_0 + t(Z_1 - Z_0) = Z_0 + \frac{Z_0 q(Z_1 - Z_0)}{qZ_0 + (1-q)Z_1} \\
&= \frac{qZ_0^2 + (1-q)Z_0Z_1 + qZ_0Z_1 - qZ_0^2}{qZ_0 + (1-q)Z_1} \\
&= \frac{Z_0Z_1}{qZ_0 + (1-q)Z_1} \\
&= \frac{1}{\frac{1}{Z_0} + q(\frac{1}{Z_1} - \frac{1}{Z_0})}
\end{aligned}
$$

$$
\frac{1}{Z} = \frac{1}{Z_0} + q(\frac{1}{Z_1} - \frac{1}{Z_0})
$$

$$W = W_0 + q(W_1 - W_0)$$

$$U = U_0 + t(U_1 - U_0)$$

$$U = \frac{U_0 W_0 + q(U_1 W_1 - U_0 W_0)}{W_0 + q(W_1 - W_0)}$$

Perspective-Correct Texturing

- In hardware rendering
  - Must be careful to interpolate texture coordinates correctly