

# Texture Mapping: 2D Texturing

# Texture Mapping

Visual complexity on demand

Vary display properties over object

Visible **pixel** maps to **location** on object

**Location** on object

used to **lookup** display **attributes**

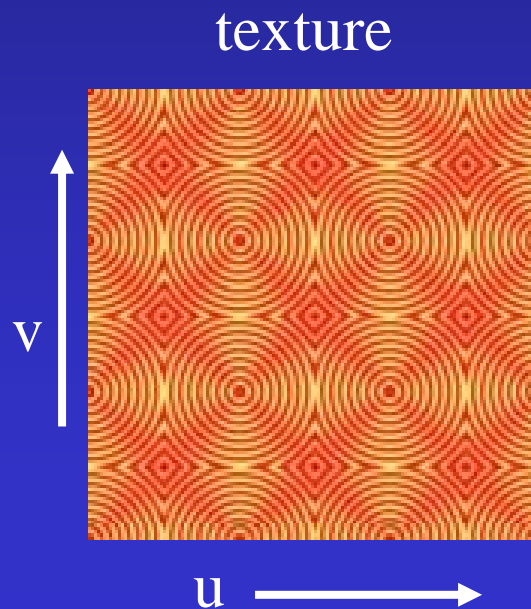
Or

as **function parameters** to generate **attributes**

# 2D Texture Mapping

Usually a 2D rectangular image or function

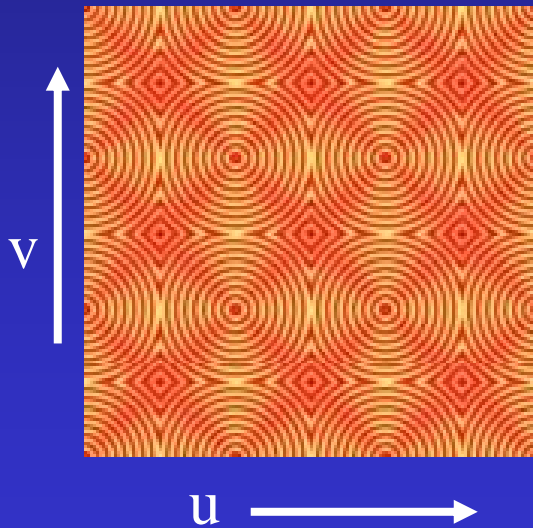
Parameterize using  $(u,v)$  texture coordinates



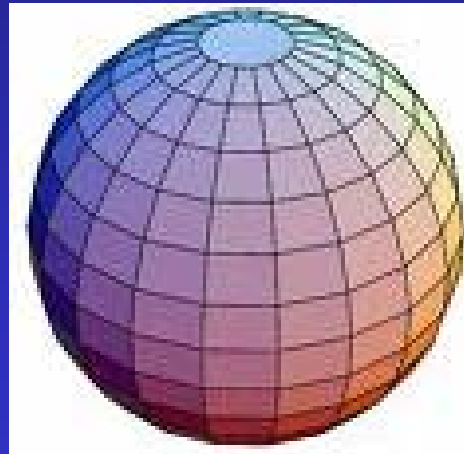
# 2D Texture Mapping

Need to parameterize surface similar to texture

texture

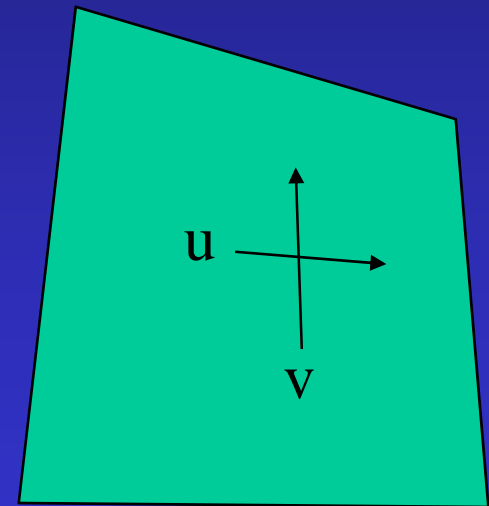


sphere



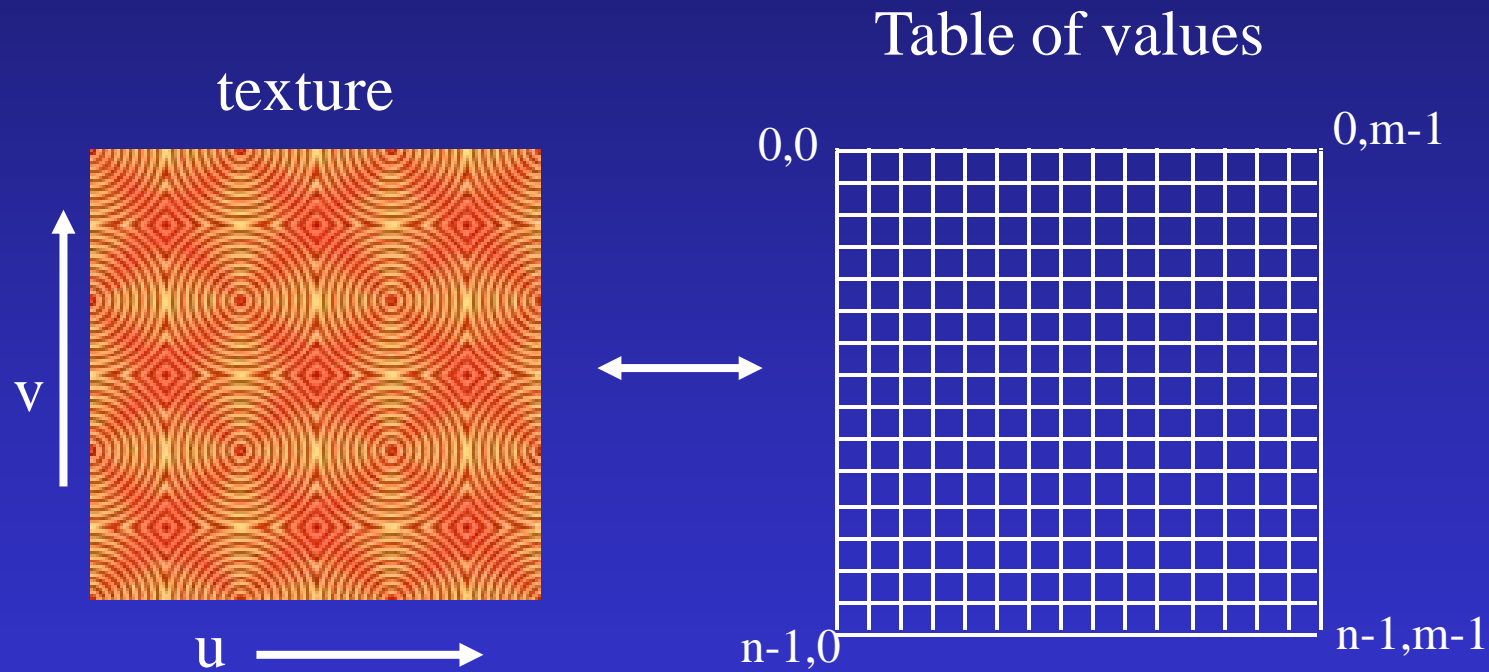
latitude - longitude

quadrilateral



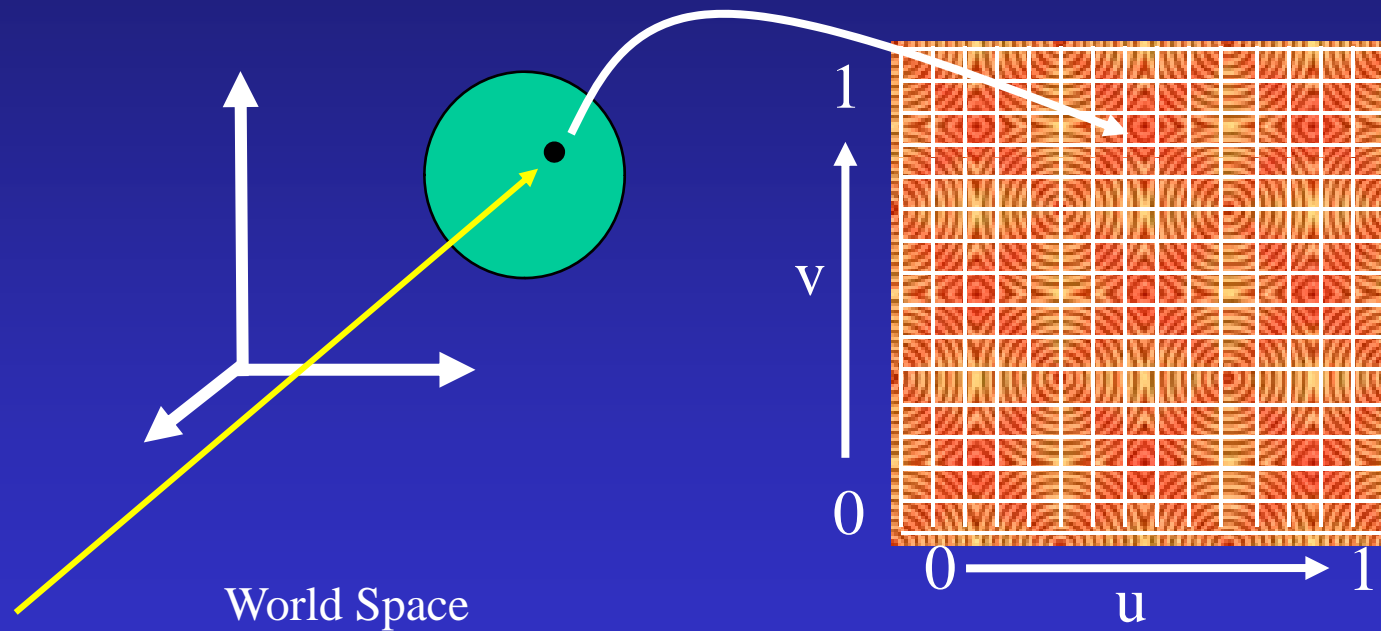
bilinear interpolation

# Texture as table of values



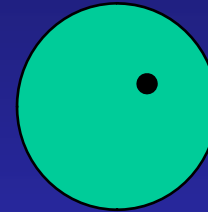
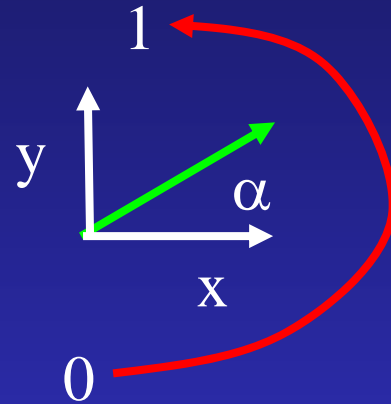
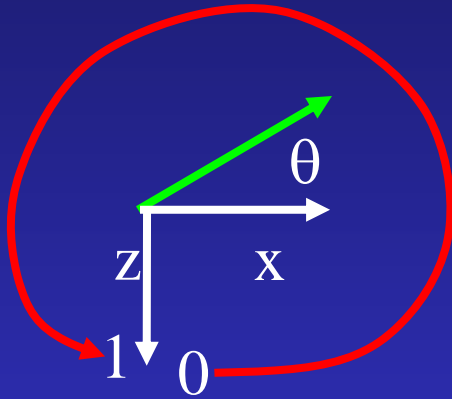
Values at grid intersections

# For sphere Texture Map Coordinates



Map  $(x,y,z)$  to  $u,v$  space to table values

# For sphere map sphere surface to (u,v)



$$s = \frac{\tan^{-1}(z/x)}{\pi/2}$$

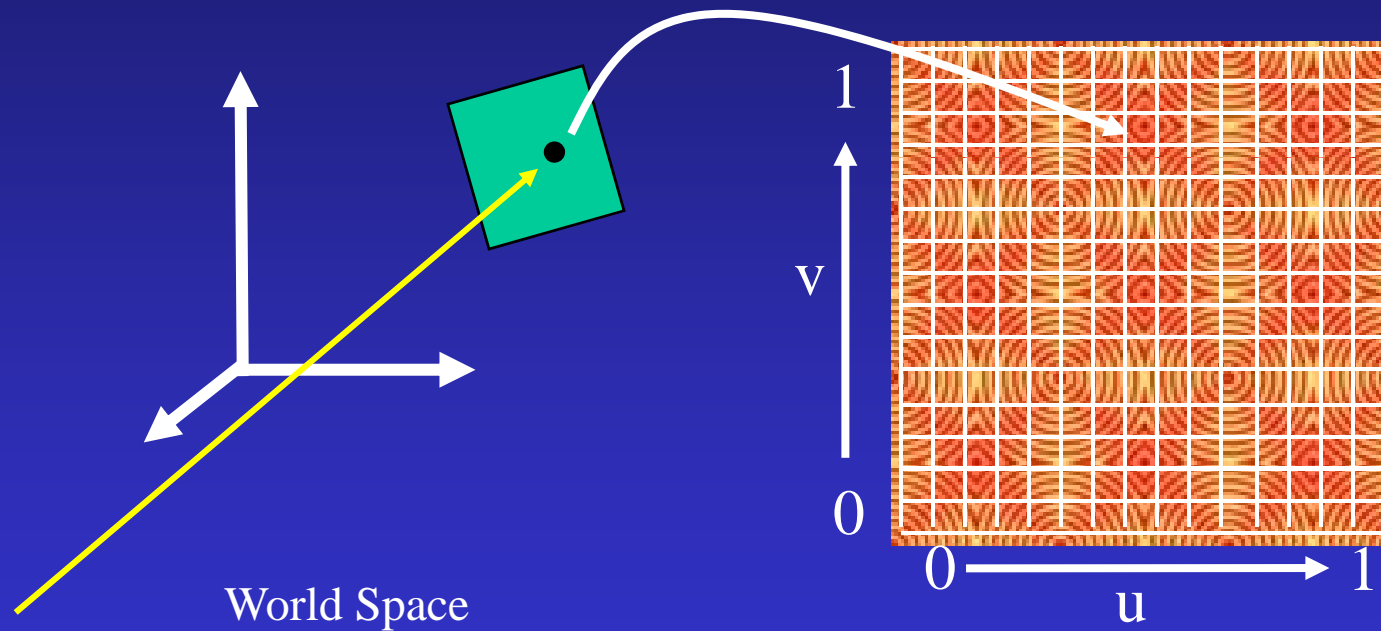
$$\text{if } (x > 0) \{ u = (1+s)/4 \}$$
$$\text{else } \{ u = 1/2 + (1-s)/4 \}$$

$$t = \frac{\tan^{-1}(y/x)}{\pi/2}$$

$$v = \frac{t+1}{2}$$

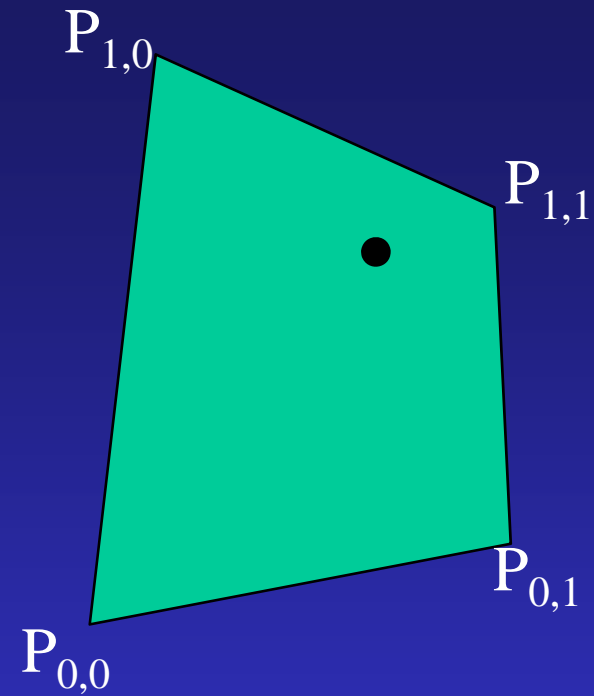
BUT -  
Has a seam  
& distorts

# For quadrilateral Texture Map Coordinates



Map  $(x,y,z)$  to  $u,v$  space to table values

# World space point to u,v space



$$P_{u,0} = P_{0,0} + u(P_{1,0} - P_{0,0})$$

$$P_{u,1} = P_{0,1} + u(P_{1,1} - P_{0,1})$$

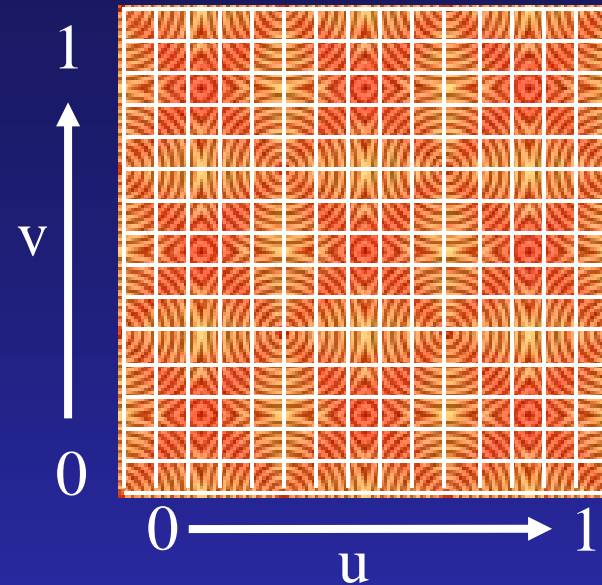
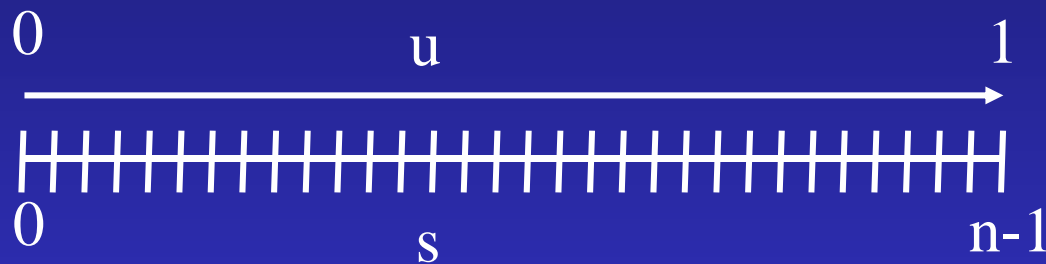
$$P_{u,v} = P_{u,0} + v(P_{u,1} - P_{u,0})$$

$$P_{u,v} = P_{0,0} + u(P_{1,0} - P_{0,0}) + v(P_{0,1} + u(P_{1,1} - P_{0,1}) - P_{0,0} + u(P_{1,0} - P_{0,0}))$$

$$P_{u,v} = P_{0,0} + u(P_{1,0} - P_{0,0}) + v(P_{0,1} - P_{0,0}) + uv(P_{1,1} - P_{0,1} + P_{1,0} - P_{0,0})$$

$$u = \frac{P_{u,v} - P_{0,0} - v(P_{0,1} - P_{0,0})}{(P_{1,0} - P_{0,0}) + v(P_{1,1} - P_{0,1} + P_{1,0} - P_{0,0})}$$

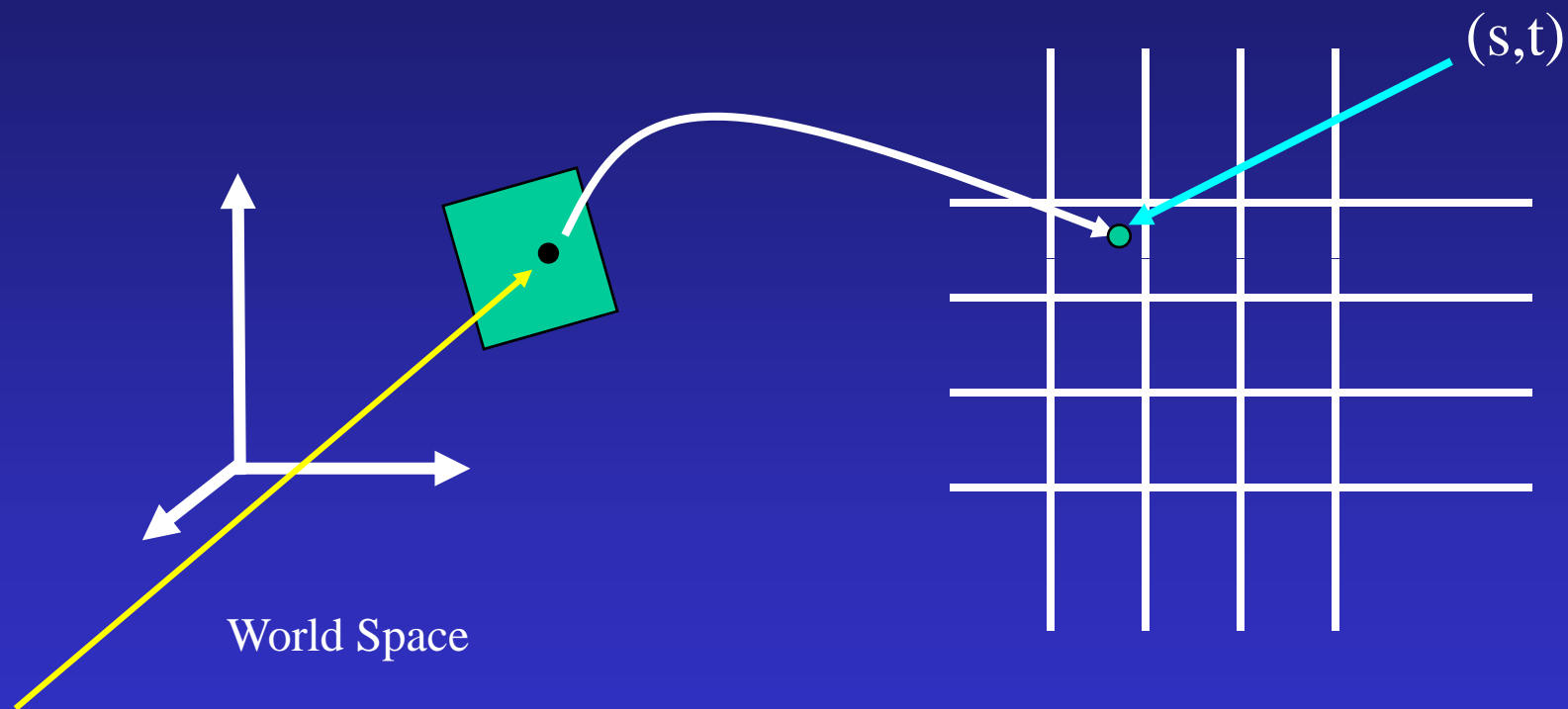
# u,v space to table indice space



$$s = u(n - 1)$$

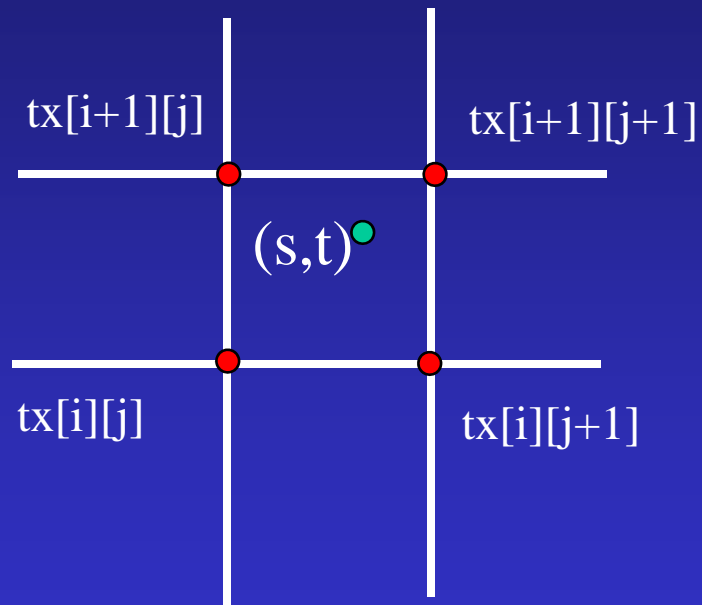
$$t = m - 1 - v(m - 1)$$

# A closer look



Values only at the intesections  
What value to use at non-intersection point?

# Closer still



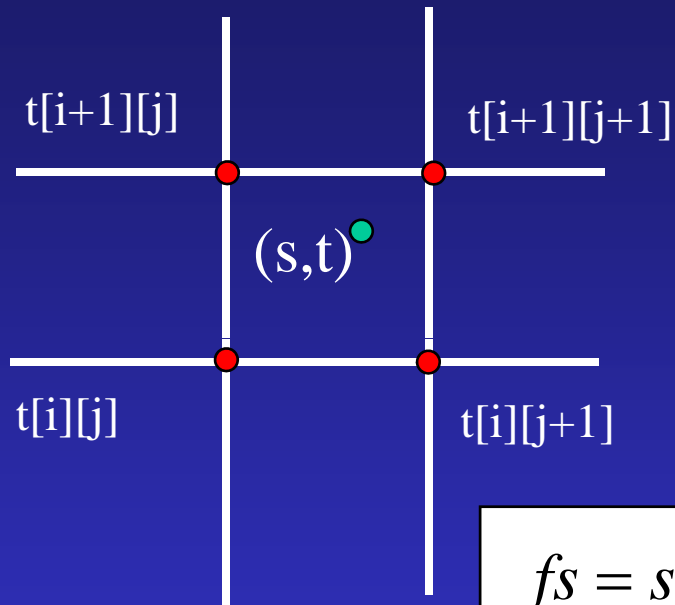
Use closest value?

$$i = \lfloor s + 0.5 \rfloor$$

$$j = \lfloor t + 0.5 \rfloor$$

$$txst = tx[i][j]$$

# Closer still



Interpolate 4 closest?

$$i = \lfloor s \rfloor$$

$$j = \lfloor t \rfloor$$

$$fs = s - \lfloor s \rfloor$$

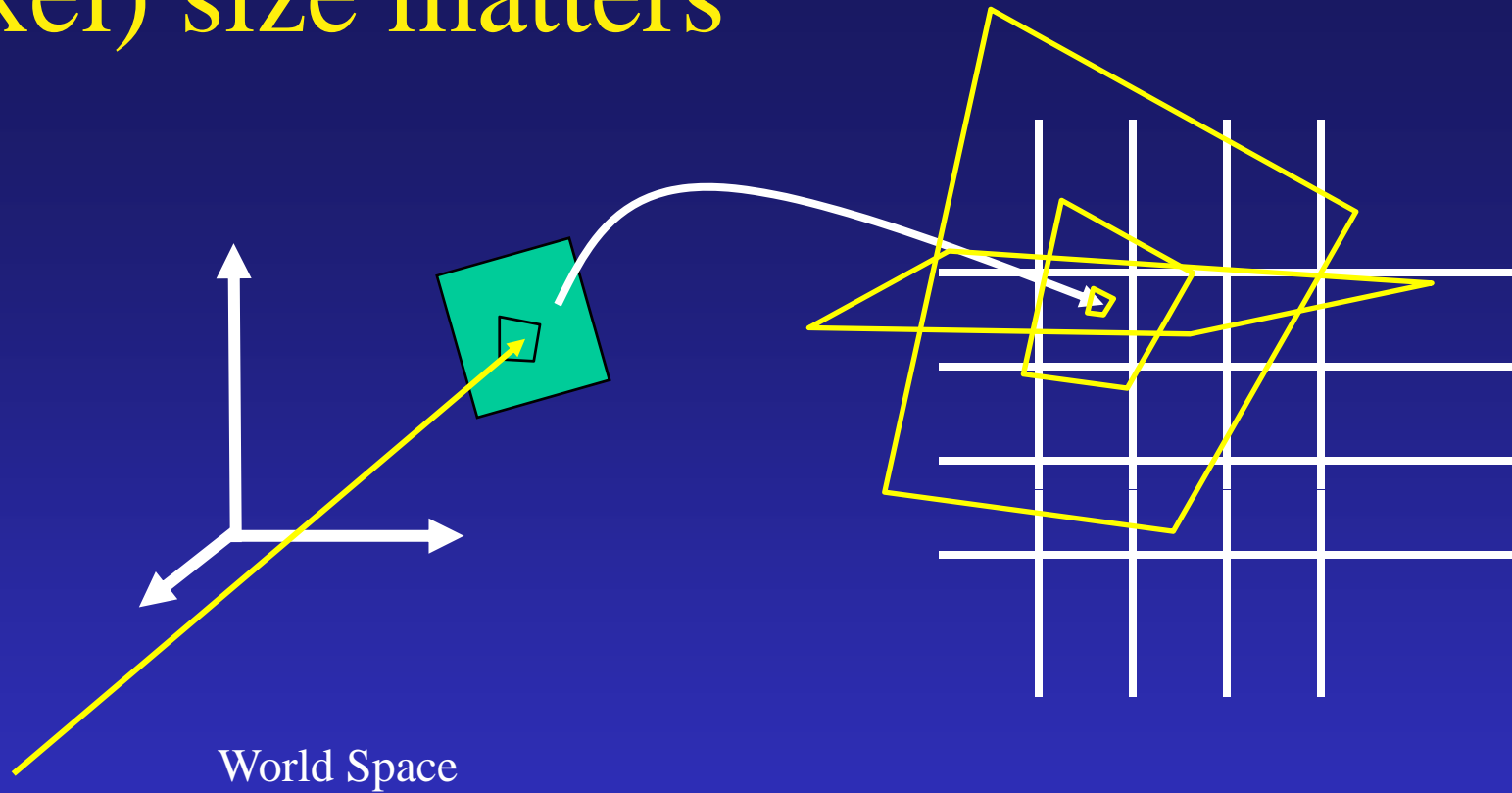
$$ft = t - \lfloor t \rfloor$$

$$ts1 = tx[i][j] + fs(tx[i+1][j] - tx[i][j])$$

$$ts2 = tx[i][j+1] + fs(tx[i+1][j+1] - tx[i][j+1])$$

$$txst = ts1 + ft(ts2 - ts1)$$

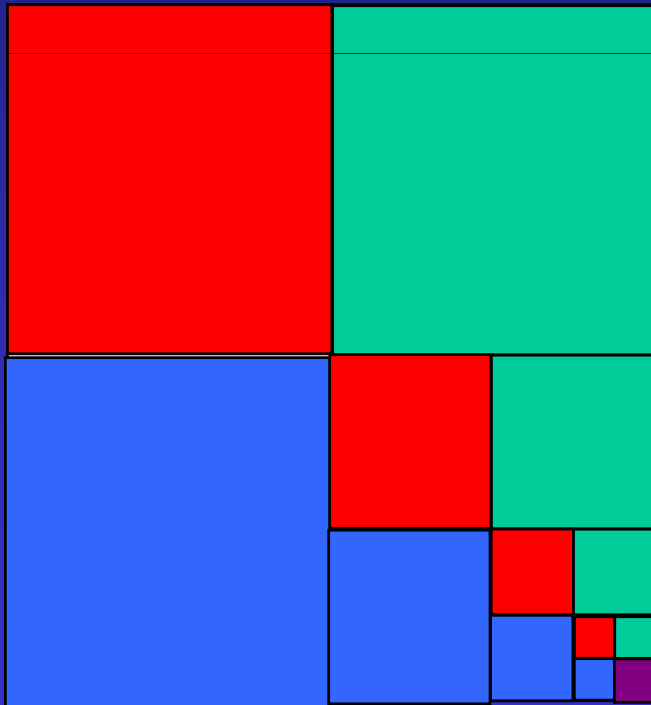
# (Pixel) size matters



Can't just use pixel center and expect good results in all cases - need to consider how entire pixel maps into texture space

# One solution: Mip-mapping

Pre-filter texture, reducing resolution  
(increase size of grid relative to pixel size)



Successive table of values  
(r,g,b) at reduced resolution

Down to single pixel

Index into highest resolution  
one in which bilinear  
interpolation makes sense