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 as table of values

Table of values

Values at grid intersections

For sphere
Texture Map Coordinates

Map (x,yz) to u,v space to table values

For sphere
map sphere surface to (u,v)

x

y

α

BUT - Has a seam & distorts

\[
\begin{align*}
\theta &= \frac{\min(\alpha/\pi/2, \pi/2)}{\pi/2} \\
\if(\theta > 0) u &= (1 + \theta)/4 \\
\else u &= 1/2 + (1 - \theta)/4 \fi
\end{align*}
\]

\[
\begin{align*}
v &= \frac{x + 1}{2} \\
\end{align*}
\]

For quadrilateral
Texture Map Coordinates

Map (x,yz) to u,v space to table values

\[
\begin{align*}
u &= \frac{\min(\alpha/\pi/2, \pi/2)}{\pi/2} \\
\if(\theta > 0) u &= (1 + \theta)/4 \\
\else u &= 1/2 + (1 - \theta)/4 \fi
\end{align*}
\]

\[
\begin{align*}
v &= \frac{x + 1}{2} \\
\end{align*}
\]
World space point to $u,v$ space

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

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

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

$u,v$ space to table indice space

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

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

A closer look

Values only at the intersections

What value to use at non-intersection point?

Closer still

Use closest value?

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

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

$$txst = tx[i][j]$$
Interpolate 4 closest?

\[
i = \lfloor s \rfloor \\
j = \lfloor t \rfloor
\]

\[
f s = s - \lfloor s \rfloor \\
f t = t - \lfloor t \rfloor
\]

\[
ts1 = ts[i][j] + fs(ts[i+1][j] - ts[i][j])
\]

\[
ts2 = ts[i][j+1] + fs(ts[i+1][j+1] - ts[i][j+1])
\]

\[
tst = ts1 + ft(ts2 - ts1)
\]

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

(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