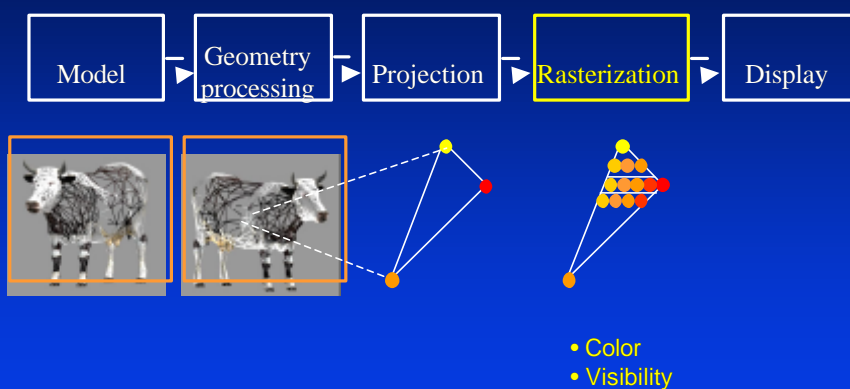


Rasterization

- Goals of rasterization
- Scan conversion and attribute interpolation
- Texture mapping
 - Texture resampling
 - Mip mapping
 - Bilinear versus perspective correct interpolation
- Limitation of *triangle rasterizers*
- Bump Mapping

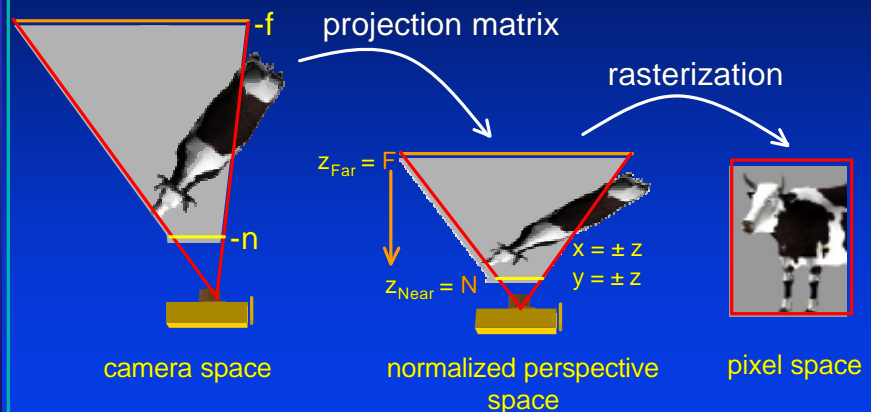
Copyright © Manuel M. Oliveira

Where Are We in the Pipeline?



Copyright © Manuel M. Oliveira

The Path to Images



Copyright © Manuel M. Oliveira

$$N = -an + b \quad F = -af + b$$

Homeomorphic Spaces

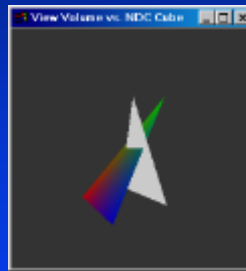
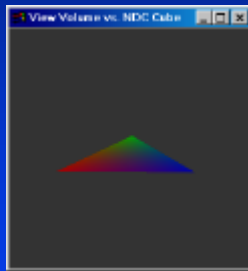
- After multiplication of its vertices by the projection matrix, a triangle is mapped onto a different triangle.
- The view frustum is mapped to a cube after division by w .



Copyright © Manuel M. Oliveira

Goals of Rasterization

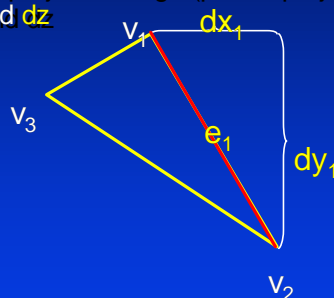
- Fill the convex hull of the projected vertices of a polygon
- Resolve visibility among polygons



Copyright © Manuel M. Oliveira

Scan Conversion

- For each triangle
 - For each projected edge (pair of projected vertices) compute dx , dy and dz



$$dx_1 = v_{2x} - v_{1x}$$

$$dy_1 = v_{2y} - v_{1y}$$

$$dz_1 = v_{2z} - v_{1z}$$

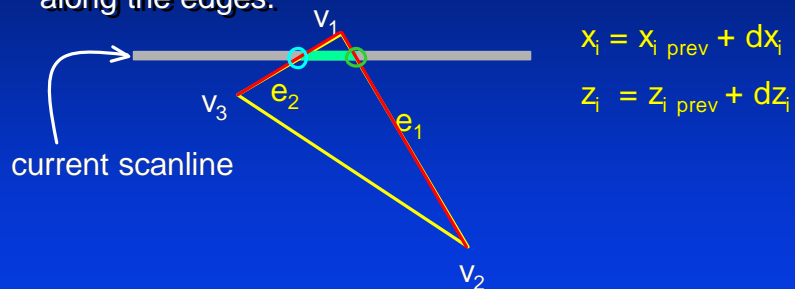
Increments in x and z coordinates per scanline:

$$inc_{ix} = (dx_i/dy_i) \text{ and } inc_{iz} = (dz_i/dy_i)$$

Copyright © Manuel M. Oliveira

Scan Conversion (cont.)

- For each projected edge crossing the current scan line, interpolate vertices' attributes (x, z, color, normals, etc.) along the edges.



Interpolate the resulting values (depth, color, normals, etc.) along the scanline segment.

Copyright © Manuel M. Oliveira

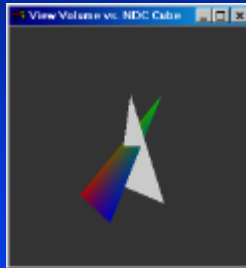
Scan Conversion (cont.)

- Bilinear interpolation ignores perspective distortions
- Perspective projection in homogeneous coordinates preserves depth (z) monotonicity

Copyright © Manuel M. Oliveira

Depth Interpolation

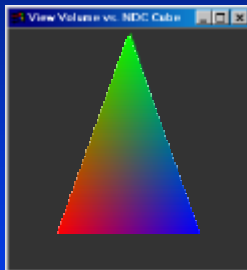
- Bilinear interpolation of depth values is not strictly correct, but produces the desired results (preserves monotonicity)



Copyright © Manuel M. Oliveira

Attribute Interpolation

- For all attributes other than depth, we need to compensate for perspective distortion



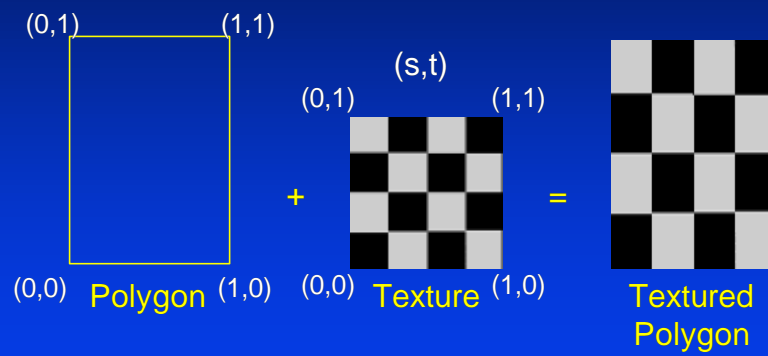
Bilinear Interpolation
Incorrect



Hyperbolic Interpolation
Correct

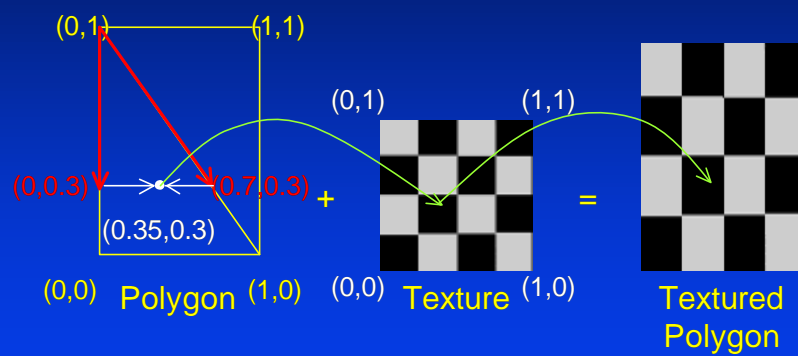
Copyright © Manuel M. Oliveira

Texture Mapping



Copyright © Manuel M. Oliveira

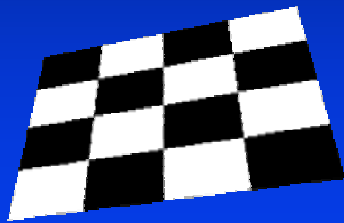
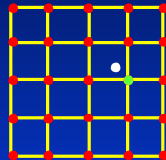
Texture Mapping (Cont.)



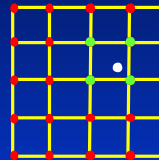
Copyright © Manuel M. Oliveira

Texture Resampling

Nearest Neighbors



Bilinear Filtering



Copyright © Manuel M. Oliveira

Bilinear Filtering

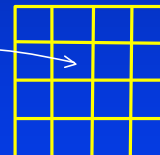
- Good results if the textured object is close to the camera
- Does not work if the textured object is far away
 - Multiple texels map to the same pixel
- Far away objects
 - Need to average the color of an entire region
 - Not practical for interactive applications



Close textured object



Far textured object



Multiple texels map to the same pixel

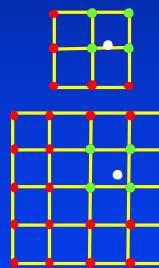
Copyright © Manuel M. Oliveira

Mip Mapping

Mip Map Pyramid



Trilinear Filtering

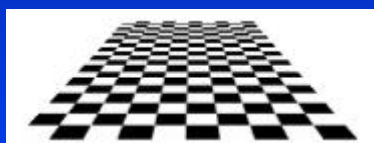


Copyright © Manuel M. Oliveira

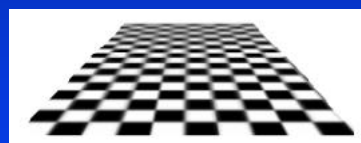
Mip Mapping (cont.)

- Mip mapping is very effective on reducing aliasing, specially in animated sequences
- Reduces aliasing by blurring the images

Bilinear Filtering



Trilinear Filtering



Copyright © Manuel M. Oliveira

Mip Mapping Level Estimation

- Based on the pixel projected area in texture space.
Let $s = s(x,y)$ and $t = t(x,y)$

- Derivative-based method** [Heckbert 83]

- Consider a pixel as a unit square in screen space

$$l = \log_2 \left(\max \left(\sqrt{\left(\frac{\partial s}{\partial x} \right)^2 + \left(\frac{\partial t}{\partial x} \right)^2}, \sqrt{\left(\frac{\partial s}{\partial y} \right)^2 + \left(\frac{\partial t}{\partial y} \right)^2} \right) \right)$$

- Area Estimation**

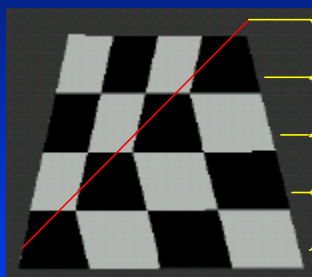
- Compute the area of the quadrilateral in texture space that the current pixel maps to. Then, compute the mip map level as

$$l = 0.5 * \log_2(\text{area})$$

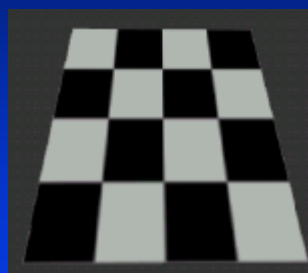
Copyright © Manuel M. Oliveira

Texture Coordinates Interpolation

- Bilinear interpolation of texture coordinates (not the same as bilinear texture resampling) produces incorrect results



Bilinear Interpolation
(Nearest Neighbor resampling)
Incorrect



Hyperbolic Interpolation
(Bilinear Filtering)
Correct

Copyright © Manuel M. Oliveira

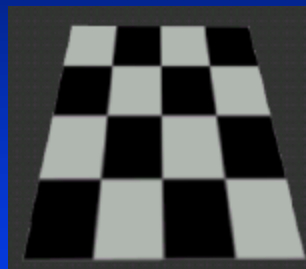
Perspective Correct Interpolation

[Blinn92]

- 1 Construct an array of values for each vertex of the polygon $[X_p, Y_p, Z_p, W_p, X_{\text{ccs}}, \dots, \text{Normal}_x, \dots, \text{Color}_r, \dots, u, \dots, 1]$, where subscript p represents the coordinates of the vertex after multiplication by the Projection Matrix.
- 2 Perform clipping.
- 3 Perform the perspective division to all elements of the vector (divide by W_p). The last term becomes $1/W_p$.
- 4 "Interpolate all values linearly down polygon edges and across scanlines internal to the polygon".
- 5 At each pixel, divide the resulting values by the interpolated $1/W_p$.

Copyright © Manuel M. Oliveira

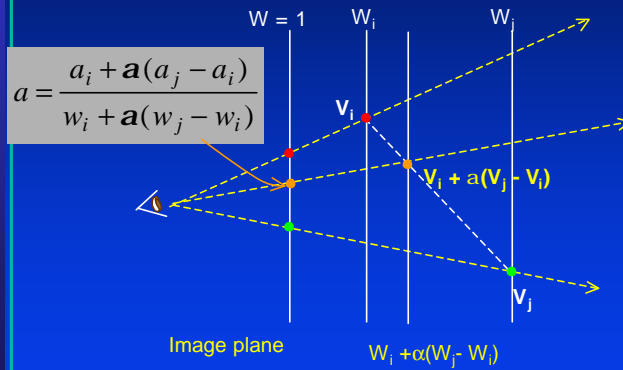
Perspective-Correct Interpolation



Copyright © Manuel M. Oliveira

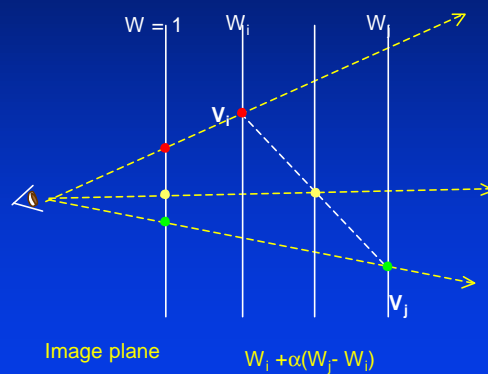
Understanding the Problem (2-D)

- Geometric Interpretation



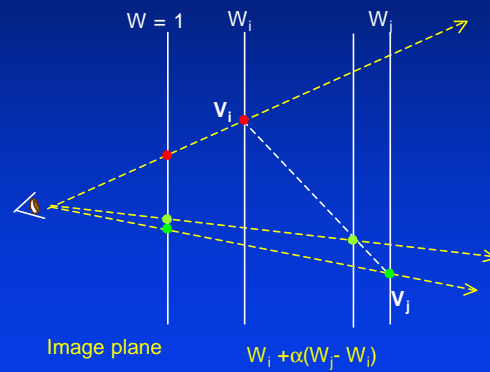
Copyright © Manuel M. Oliveira

Understanding the Problem (2-D)



Copyright © Manuel M. Oliveira

Understanding the Problem (2-D)



Copyright © Manuel M. Oliveira

Solution

- Interpolate using the attribute values as if in Camera/Eye space
- Notation
 - **E**: Camera/Eye Space. Point represented as $(E_x, E_y, E_z, 1)^T$
 - **H**: Homogeneous Space. Point represented as $(H_x, H_y, H_z, H_w)^T$
 - **M**: 4x4 Projection & Viewport Transformation Matrix. $H = ME$
 - **P**: Pixel Space. Obtained with the perspective divide after clipping. $P = H / H_w$. Point represented as $(P_x, P_y, P_z, 1)^T$

Copyright © Manuel M. Oliveira

Relationship Between the E and P Spaces

$$H = ME$$

and

$$P = \frac{H}{H_w}$$

Thus

$$P = \left(\frac{1}{H_w} \right) ME$$

$$\tilde{E} = \begin{bmatrix} \frac{E_x}{H_w} & \frac{E_y}{H_w} & \frac{E_z}{H_w} & \frac{1}{H_w} \end{bmatrix}^T = \begin{bmatrix} \tilde{E}_x & \tilde{E}_y & \tilde{E}_z & \tilde{E}_w \end{bmatrix}^T$$

Copyright © Manuel M. Oliveira

Obtaining the Right Vector

- Vertices attributes as defined in Eye Space

$$\begin{bmatrix} E_x, E_y, E_z, N_x, \dots, C_{red}, \dots, u, v, 1 \end{bmatrix}^T$$

- After multiplying M by E

$$\begin{bmatrix} H_x, H_y, H_z, H_w, E_x, E_y, E_z, N_x, \dots, C_{red}, \dots, u, v, 1 \end{bmatrix}^T$$

- After Clipping and Perspective divide

$$\begin{bmatrix} P_x, P_y, P_z, 1, \tilde{E}_x, \tilde{E}_y, \tilde{E}_z, \tilde{N}_x, \dots, \tilde{C}_{red}, \dots, \tilde{u}, \tilde{v}, \frac{1}{H_w} \end{bmatrix}^T$$

Copyright © Manuel M. Oliveira

Obtaining the Right Interpolation

- For each scan converted pixel, linearly interpolate the values of the “tilded” attributes and divide the result by the interpolated $1/H_w$

$$\left[P_x, P_y, P_z, 1, \tilde{E}_x, \tilde{E}_y, \tilde{E}_z, \tilde{N}_x, \dots, \tilde{C}_{red}, \dots, \tilde{u}, \tilde{v}, \frac{1}{H_w} \right]^T$$

Note that dividing by $1/H_w$ is equivalent to multiplying by H_w , which will bring the attribute values back to Eye space.

Copyright © Manuel M. Oliveira

Does It Really Work?

- Interpolation and projection in Homogeneous Space (we know this works!):

$$l = \frac{a_i + a(b_i - a_i)}{a_w + a(b_w - a_w)} = \frac{a_i(1 - a) + ab_i}{a_w(1 - a) + ab_w}$$

- Interpolation and projection using interpolated $1/H_w$

$$l' = \frac{\frac{a_i}{a_w} + a \left(\frac{b_i}{b_w} - \frac{a_i}{a_w} \right)}{\frac{1}{a_w} + a \left(\frac{1}{b_w} - \frac{1}{a_w} \right)} = \frac{a_i b_w (1 - a) + a a_w b_i}{b_w (1 - a) + a a_w}$$

Copyright © Manuel M. Oliveira

But They Still Look Different ...

- Interpolation and projection in Homogeneous Space:

$$l = \frac{a_i(1-a) + ab_i}{a_w(1-a) + ab_w}$$

- We can rewrite l' as

$$l' = \frac{a_i b_w (1-a) + a a_w b_i}{b_w (1-a) + a a_w} = \frac{\frac{a_i}{a_w} (1-a) + a \left(\frac{b_i}{b_w} \right)}{\frac{1}{a_w} (1-a) + a \left(\frac{1}{b_w} \right)}$$

Copyright © Manuel M. Oliveira

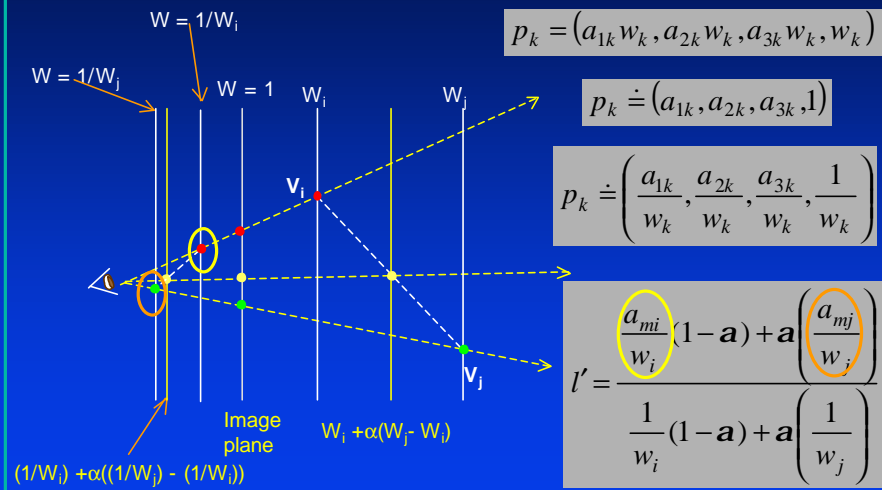
In Homogeneous Space ...

$$[\tilde{E}_x, \tilde{E}_y, \tilde{E}_z, \tilde{E}_w]^T = \left[\frac{E_x}{H_w}, \frac{E_y}{H_w}, \frac{E_z}{H_w}, \frac{1}{H_w} \right]^T$$

- This provides the geometric intuition we need to understand what is going on: \tilde{E} is on the homogeneous plane $w = 1/H_w$!!!
- Such a plane is behind the image plane for $\|H_w\| > 1$

Copyright © Manuel M. Oliveira

Geometrically...



Copyright © Manuel M. Oliveira