



Rasterization: Shading and Visibility

COS 426, Fall 2022

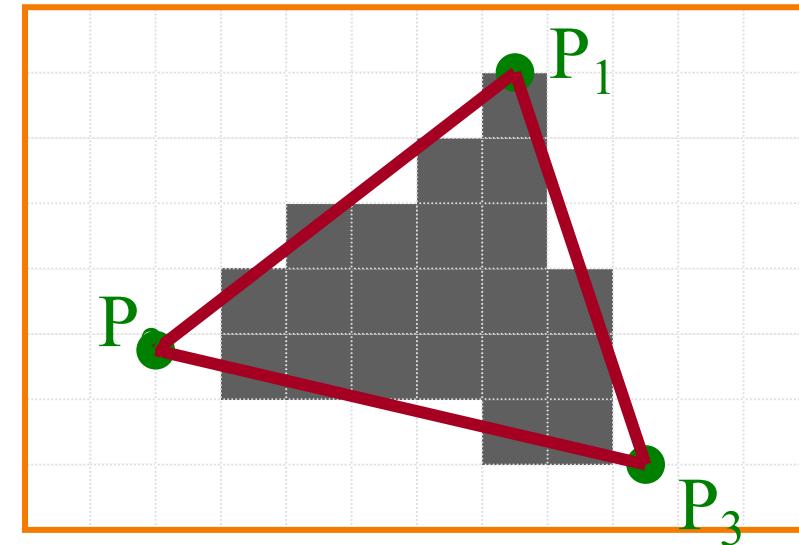
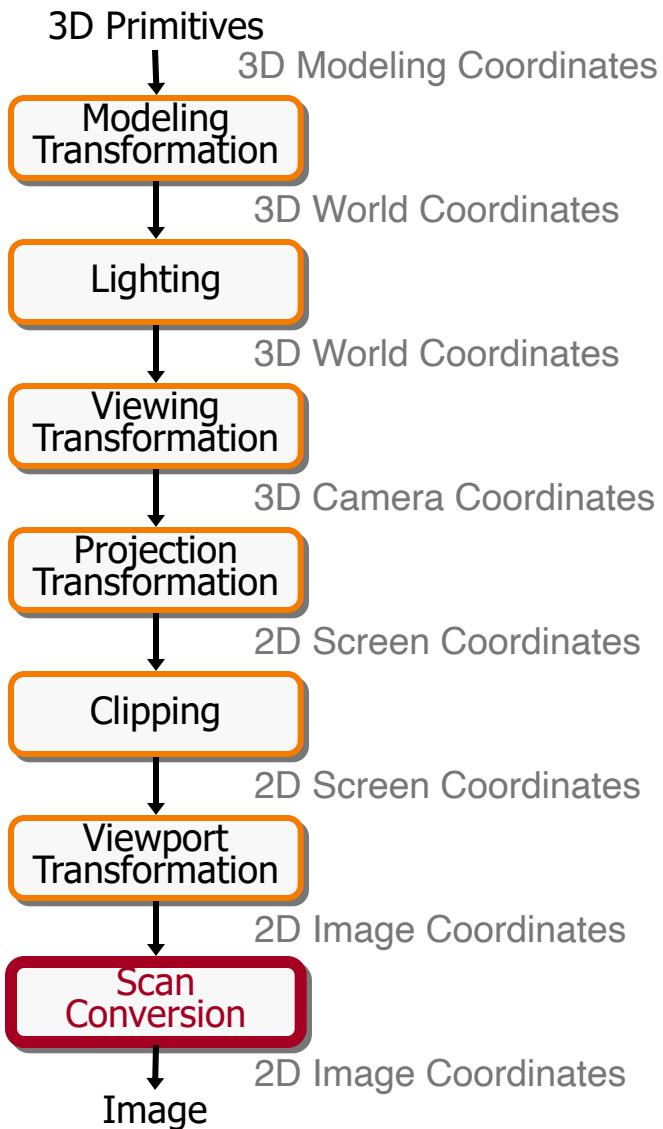


PRINCETON UNIVERSITY



Rasterization Pipeline

(for direct illumination)



Scan Conversion



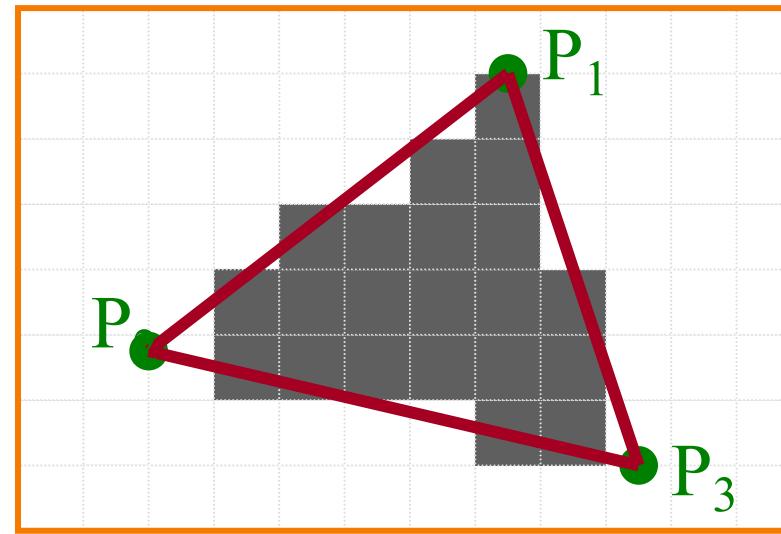
Rasterization

- Scan conversion (last time)
 - Determine which pixels to fill
- Shading
 - Determine a color for each filled pixel
- Texture mapping
 - Describe shading variation within polygon interiors
- Visible surface determination
 - Figure out which surface is front-most at every pixel



Shading

- How do we choose a color for each filled pixel?

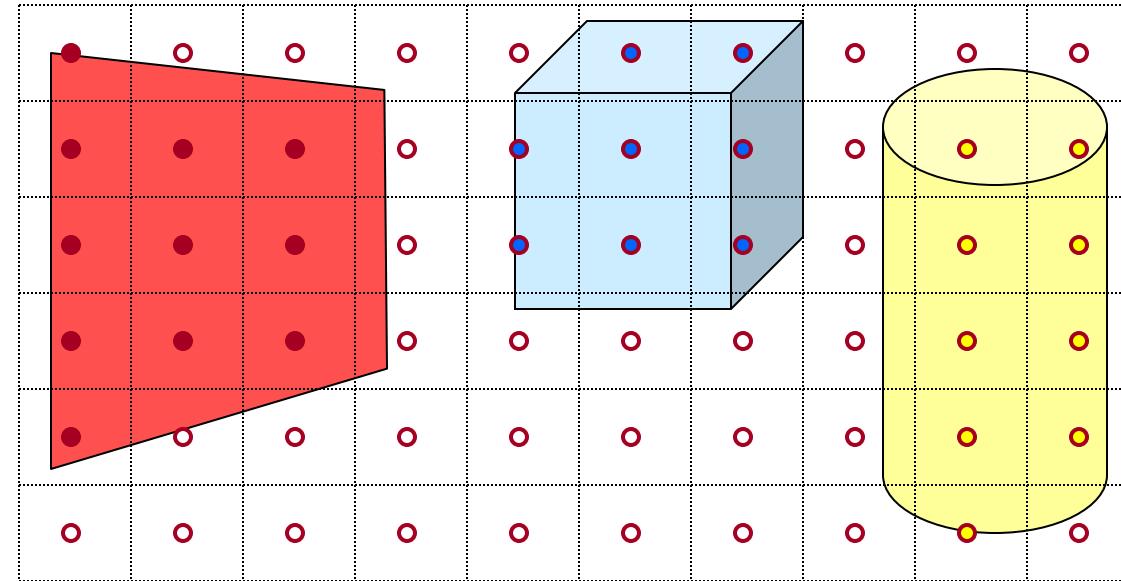


Emphasis on methods that can be implemented in hardware...



Taking Inspiration from Ray Casting

- Simplest shading approach is to perform independent lighting calculation for every pixel



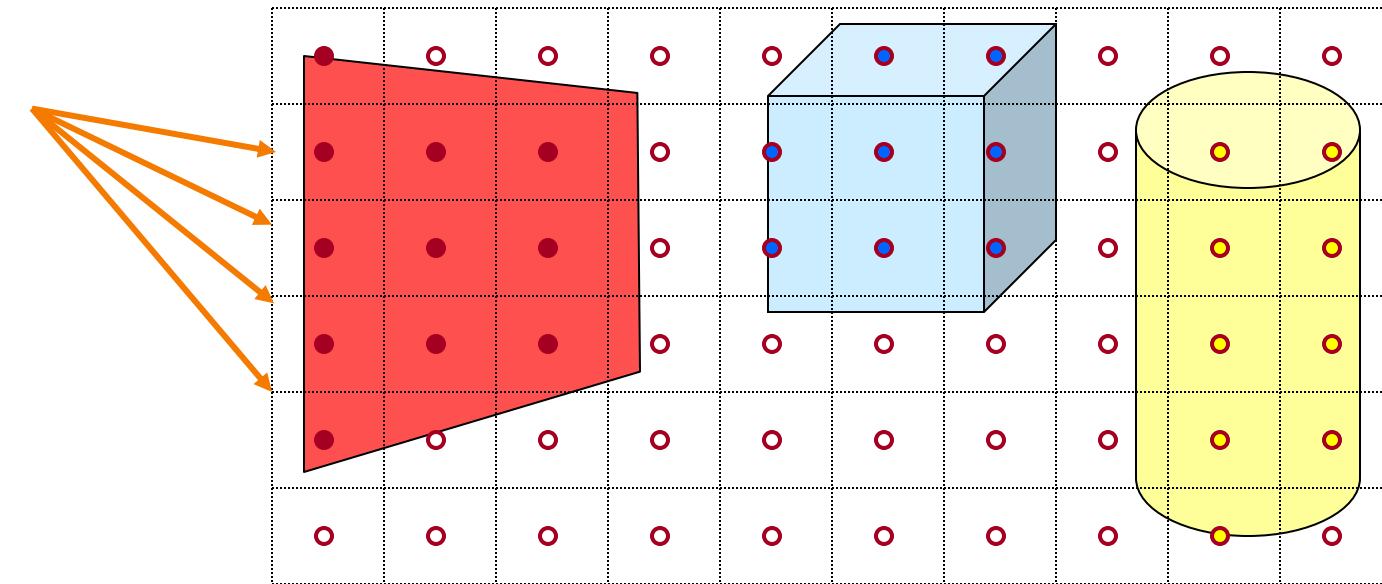
$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$



Polygon Shading

- Increase efficiency by exploiting spatial coherence
 - Illumination calculations for pixels covered by same primitive are related to each other

Similar...



$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$



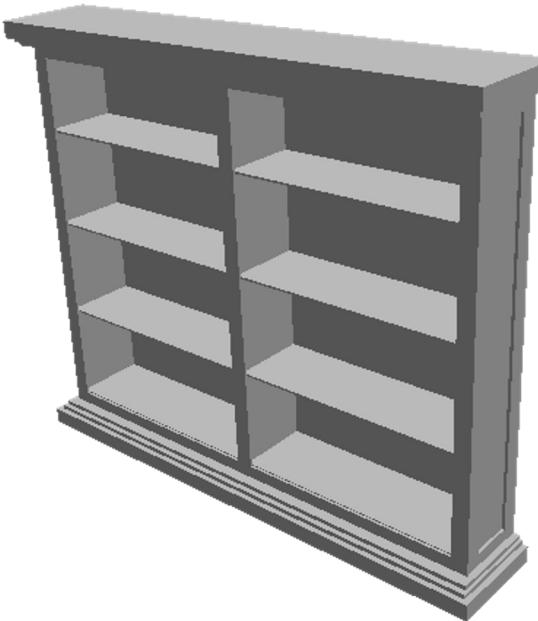
Polygon Shading Algorithms

- **Flat Shading**
- Gouraud Shading
- Phong Shading



Flat Shading

- What if a faceted object is illuminated only by directional light sources and is viewed from infinitely far away

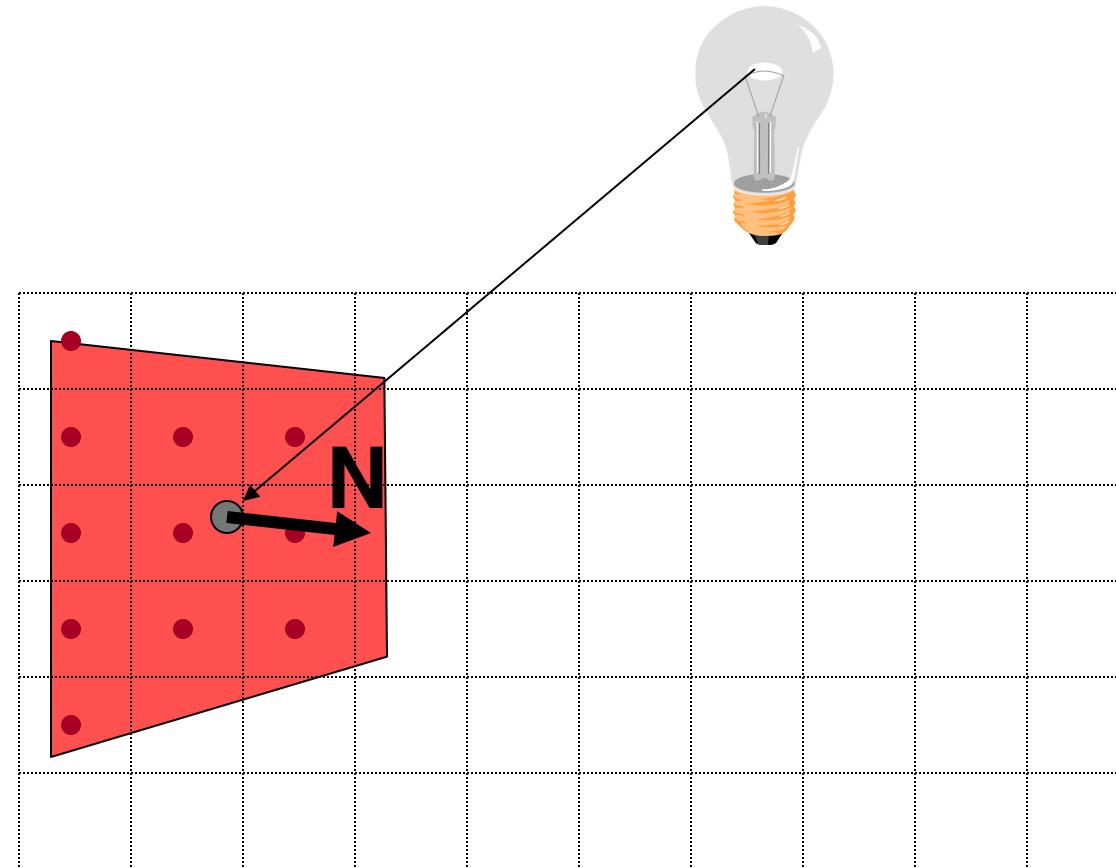


$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$



Flat Shading

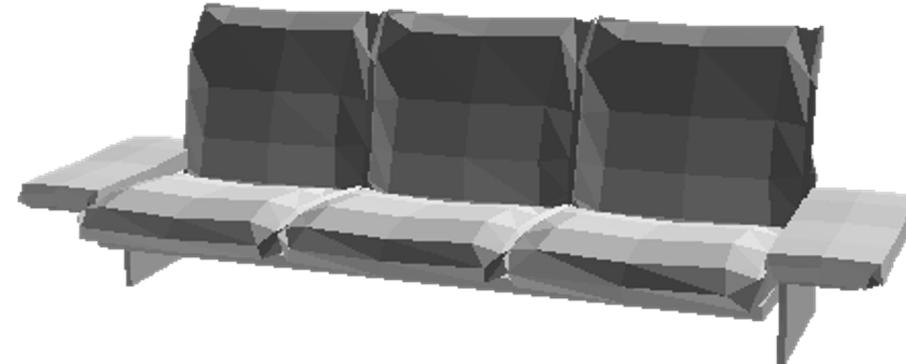
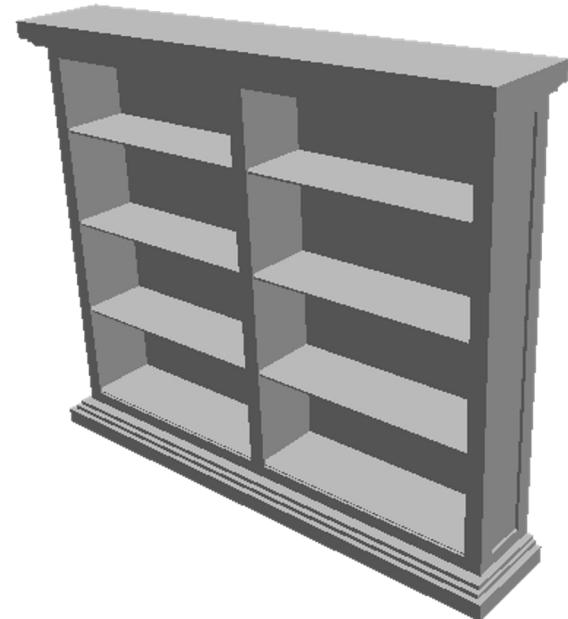
- One illumination calculation per polygon is enough
 - Assign all pixels inside each polygon the same color





Flat Shading

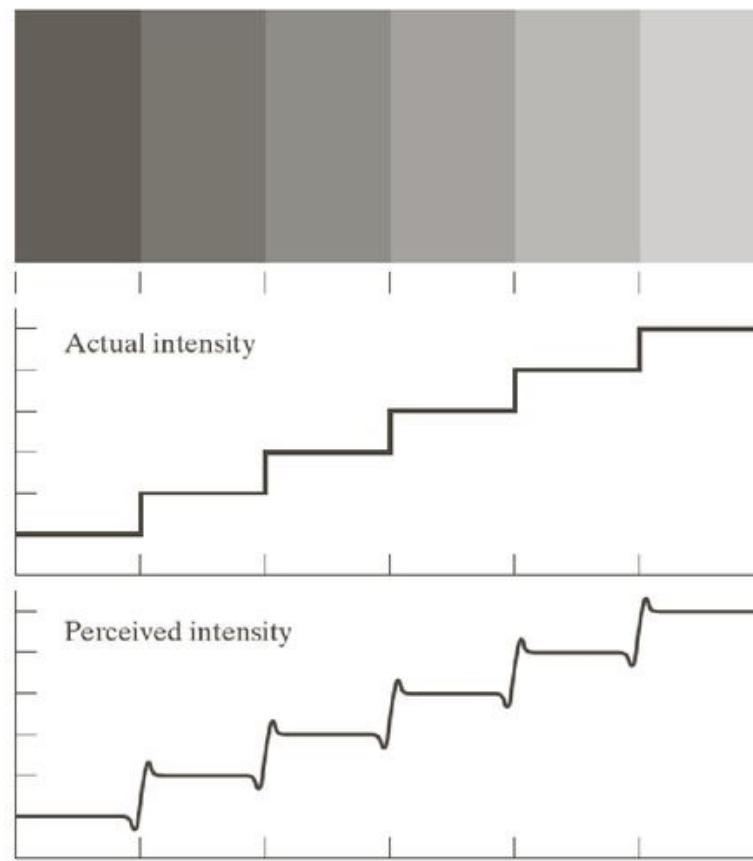
- Objects look like they are composed of polygons
 - OK for polyhedral objects
 - Not so good for smooth surfaces





Mach Band Effect

- Visual system perceives edges between adjacent shades of gray with exaggerated contrast





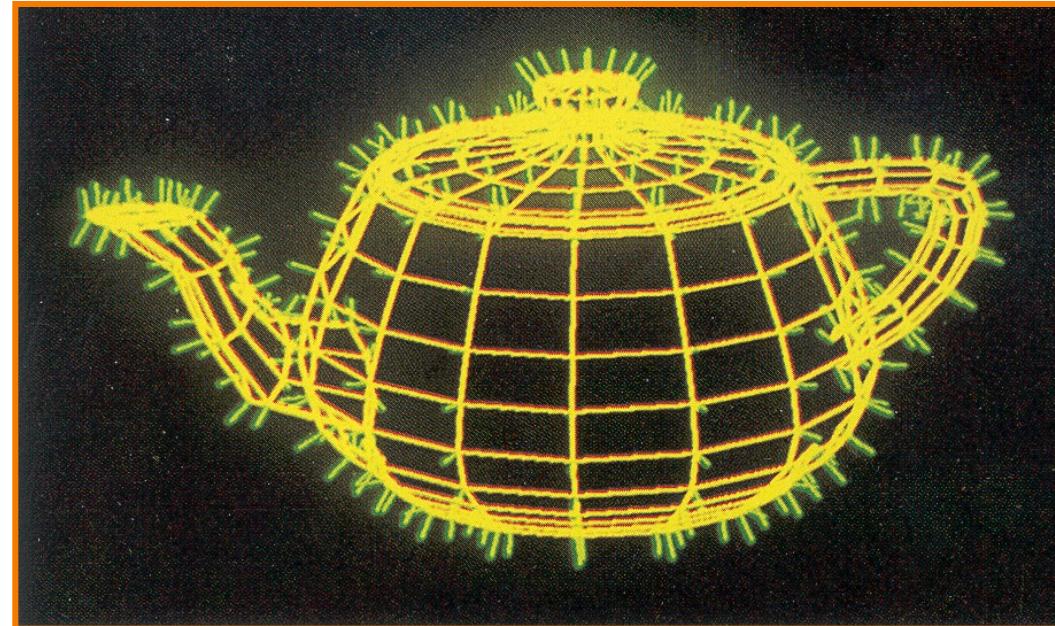
Polygon Shading Algorithms

- Flat Shading
- **Gouraud Shading**
- Phong Shading



Gouraud Shading

- Approximate smooth surface by polygonal mesh with a normal stored at each **vertex**
 - “Shared normals”
 - Calculated as (possibly area-weighted) average of normals of adjacent faces

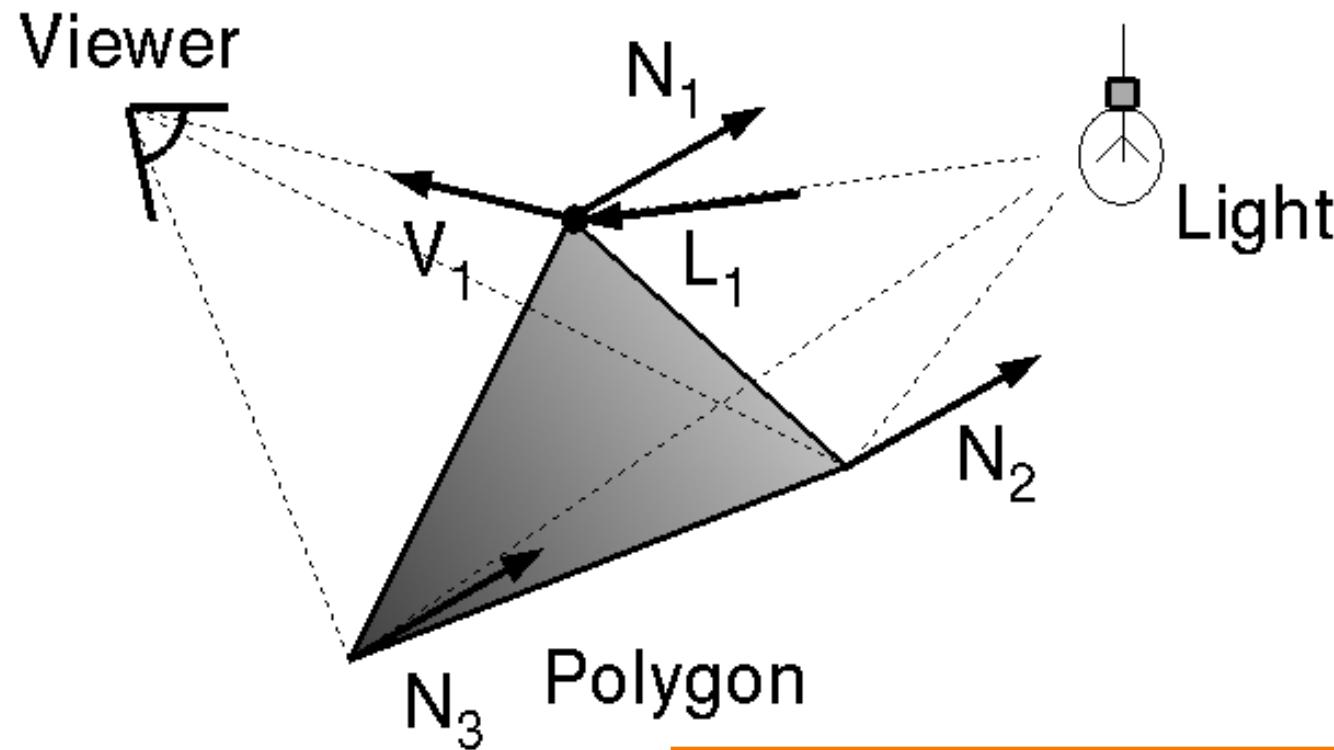


Watt Plate 7



Gouraud Shading

- One lighting calculation per vertex
 - Pixel colors inside polygon interpolated from colors computed at vertices

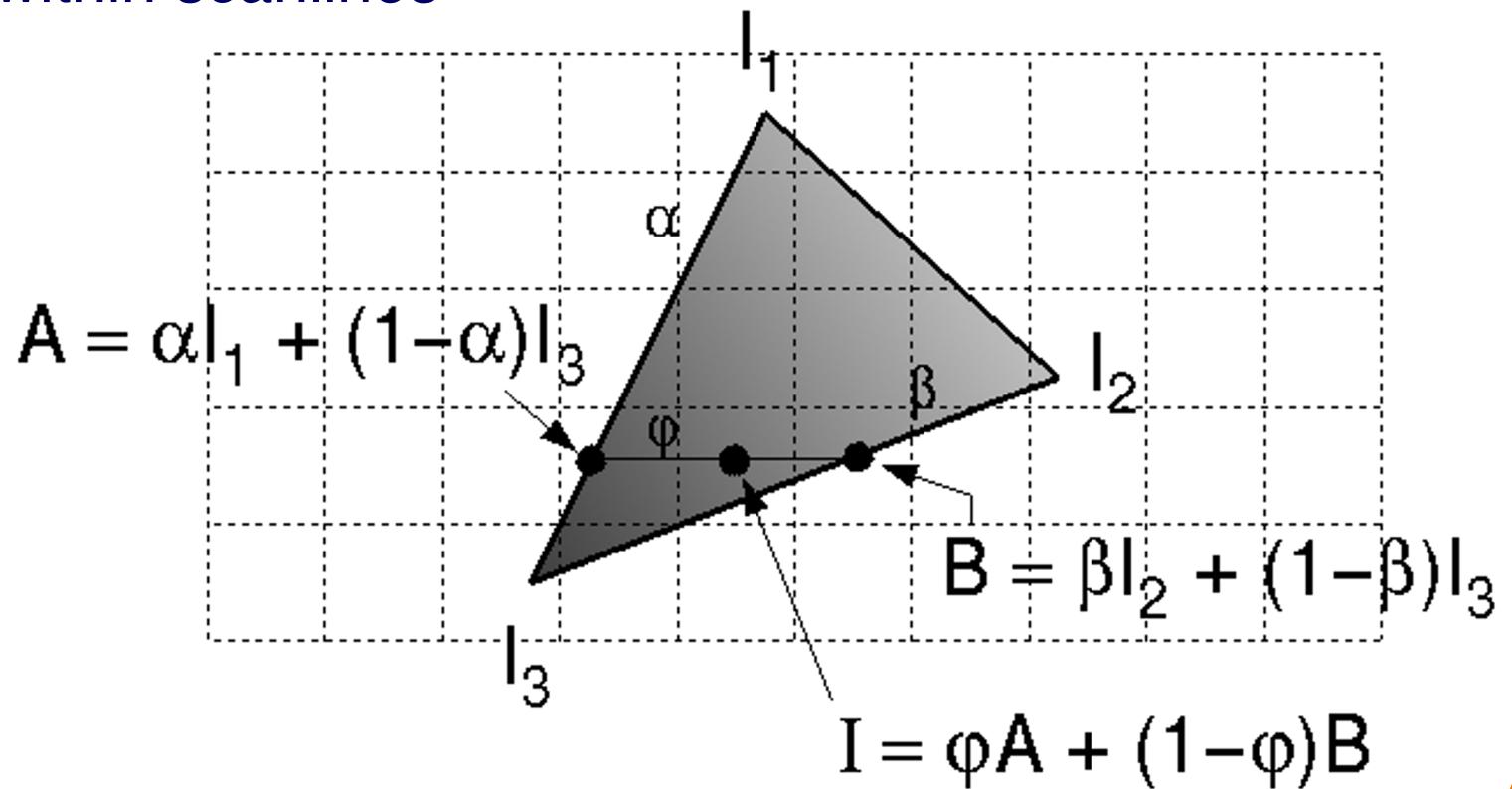


$$I = I_E + K_A I_{AL} + \sum_i (K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i)$$



Gouraud Shading

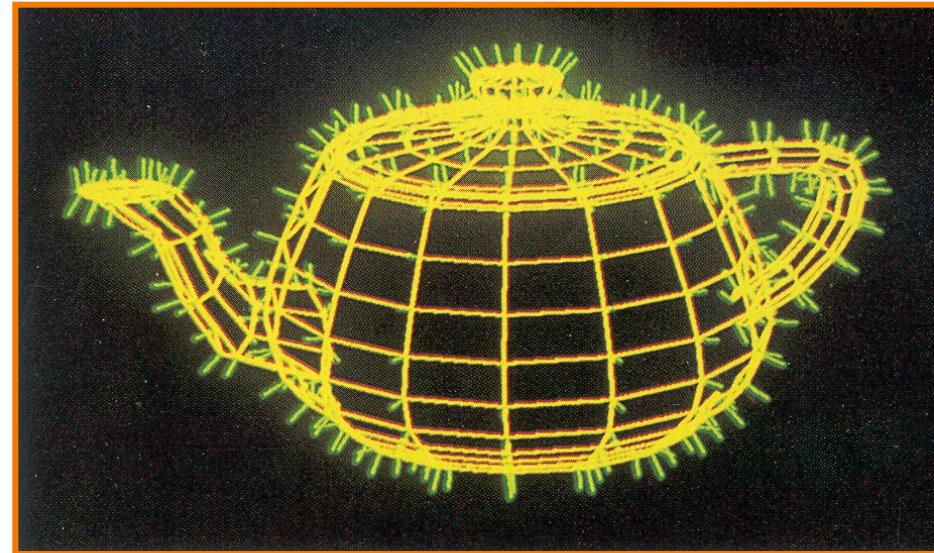
- Bilinear interpolation of colors at vertices
 - Down and across scan lines = barycentric interpolation!
 - Specifically, linearly interpolate at left and right endpoints of each span, then linearly interpolate within scanlines





Gouraud Shading

- Smooth shading over adjacent polygons
 - Curved surfaces
 - Illumination highlights
 - Soft shadows



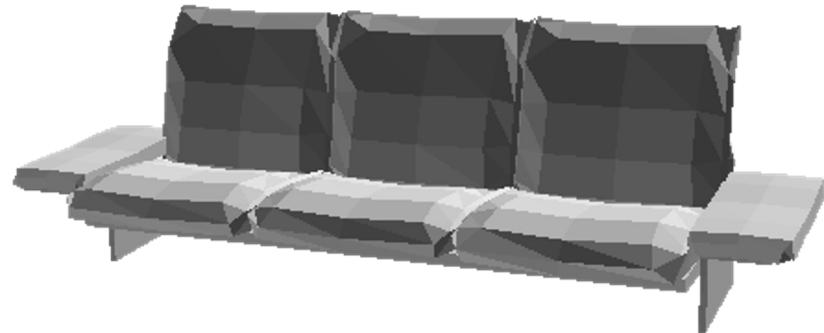
Watt Plate 7

Mesh with shared normals at vertices

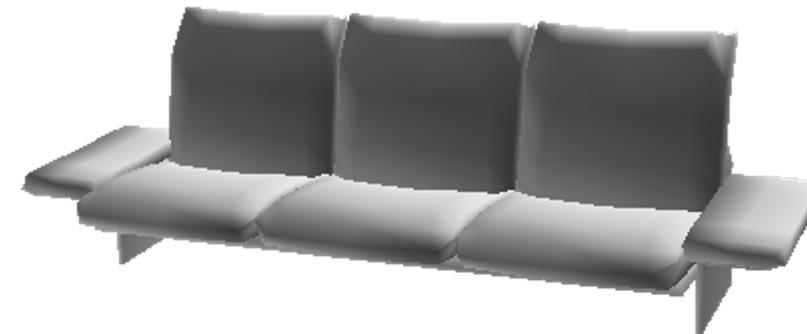


Gouraud Shading

- Produces smoothly shaded polygonal mesh
 - Piecewise linear (!) approximation
 - Need fine mesh to capture subtle lighting effects



Flat Shading



Gouraud Shading

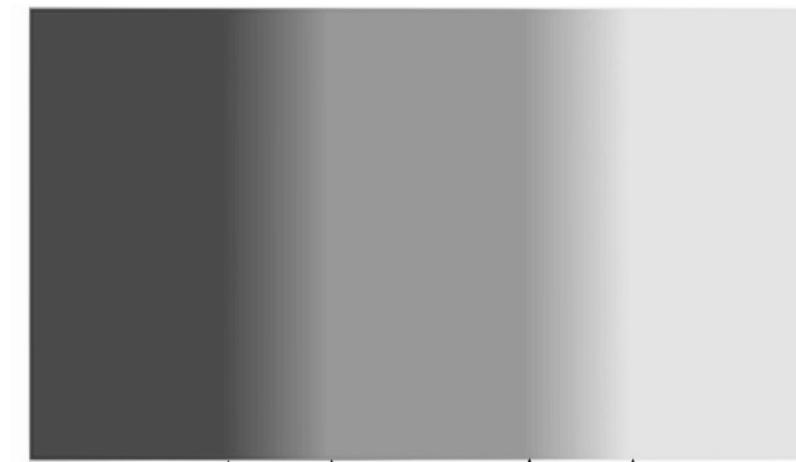


Mach Band Effect

- Mach Band Effect also affects Gouraud Shading for piecewise linear interpolation



Actual Intensity



The thin lines or "bands" along the gradient are illusory.



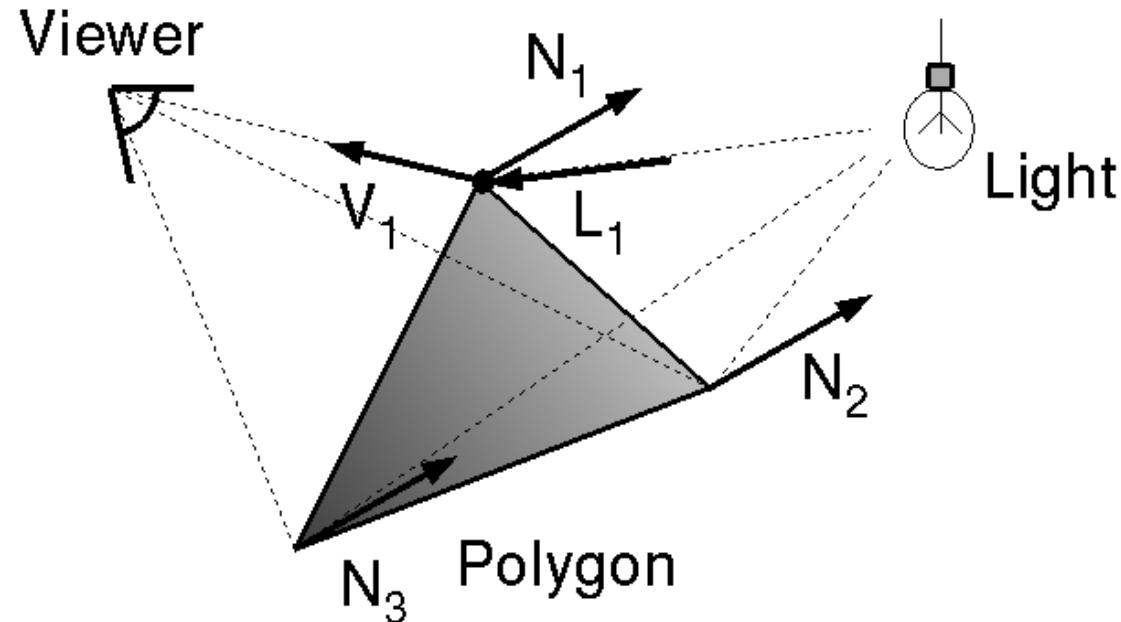
Polygon Shading Algorithms

- Flat Shading
- Gouraud Shading
- **Phong Shading** (\neq Phong reflectance model)



Phong Shading

- What if polygonal mesh is too coarse to capture illumination effects in polygon interiors?

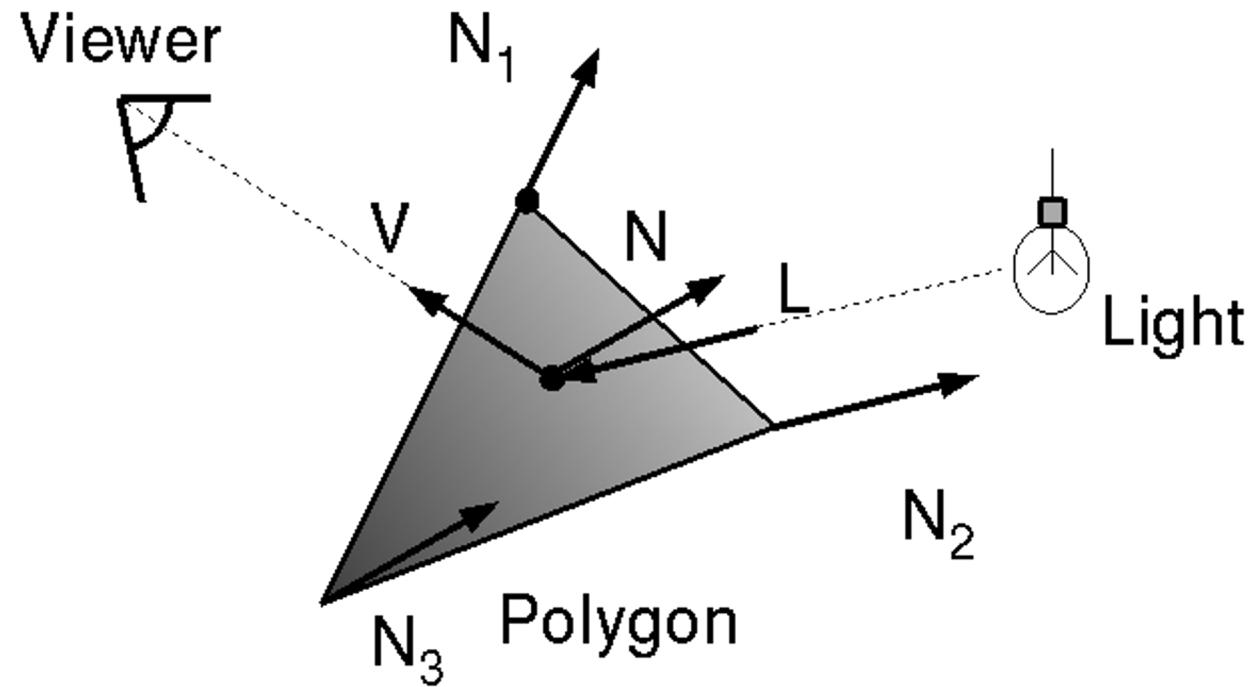


$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$



Phong Shading

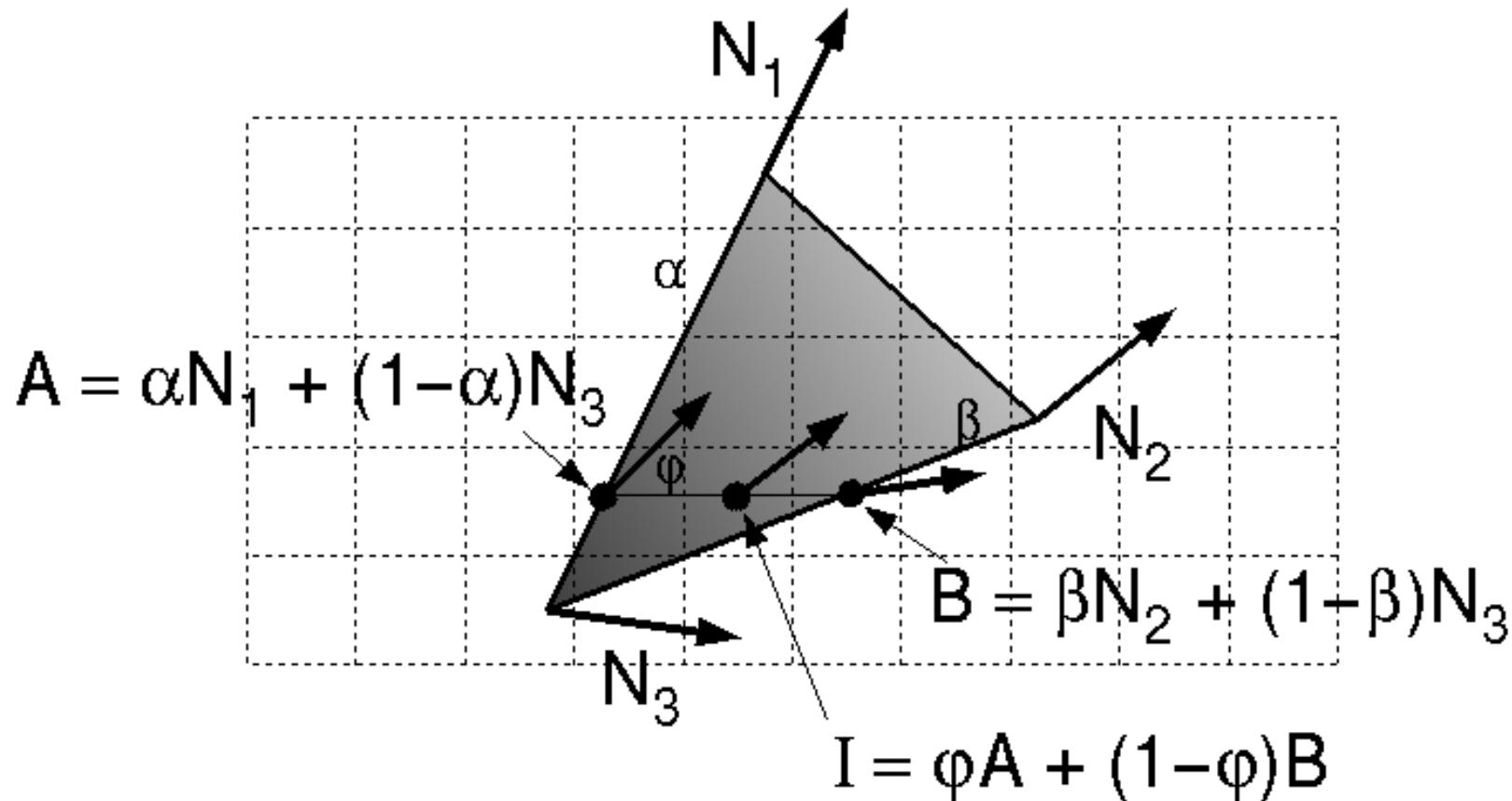
- One lighting calculation per pixel
 - Approximate surface **normals** for points inside polygons by bilinear interpolation of **normals** from vertices
 - Normalize interpolated normal to unit length
 - Finally, do per-pixel lighting calculation using interpolated normal





Phong Shading

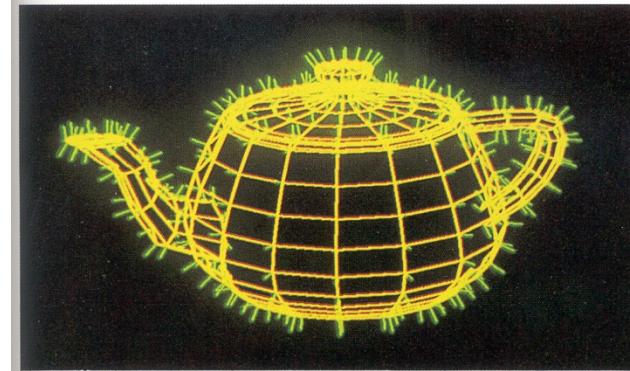
- Bilinear interpolation of surface normals at vertices





Polygon Shading Algorithms

Wireframe



Flat



Gouraud



Phong

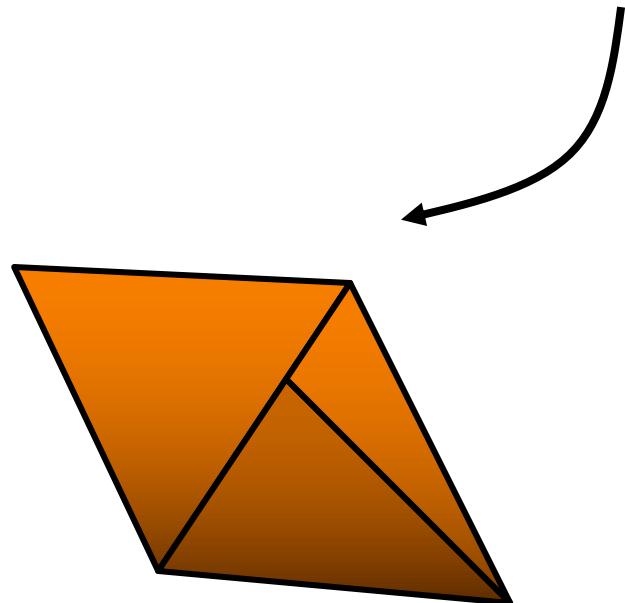
Demo: <https://threejs.org/docs/scenes/material-browser.html#MeshPhongMaterial>

Watt Plate 7

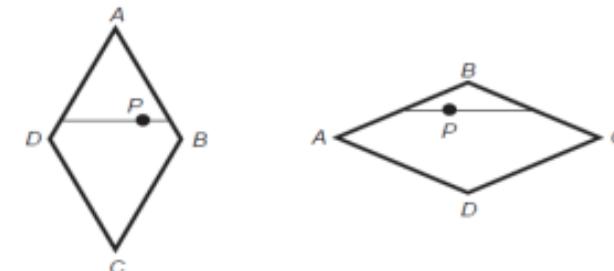


Shading Issues

- Problems with interpolated shading:
 - Polygonal silhouettes still obvious
 - Perspective distortion (due to screen-space interpolation)
 - Problems at T-junctions



The results of interpolated-shading is not independent of the projected polygons position (Foley Figure 14.22).





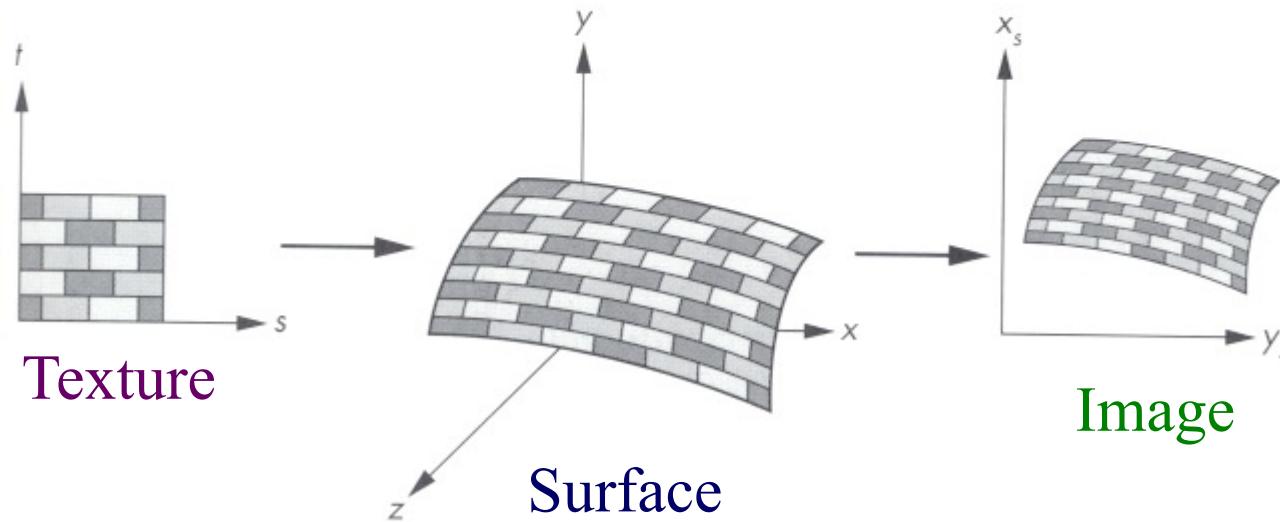
Rasterization

- Scan conversion
 - Determine which pixels to fill
 - Shading
 - Determine a color for each filled pixel
- **Texture mapping**
- Describe shading variation within polygon interiors
- Visible surface determination
 - Figure out which surface is front-most at every pixel



Textures

- Describe **color variation** in interior of 3D polygon
 - When scan converting a polygon, **vary pixel colors** according to values fetched from a texture image

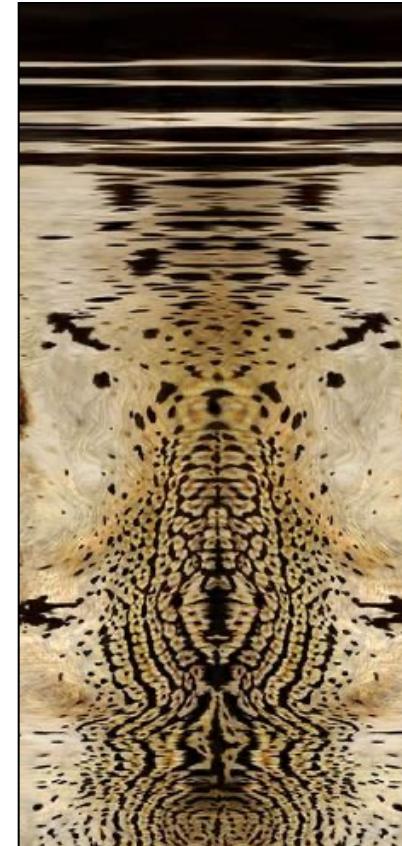


Angel Figure 9.3



Textures

- Add visual detail to surfaces of 3D objects

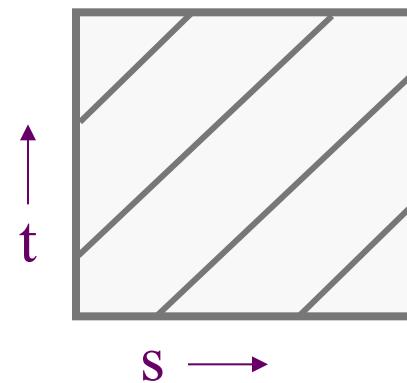


[Daren Horley]

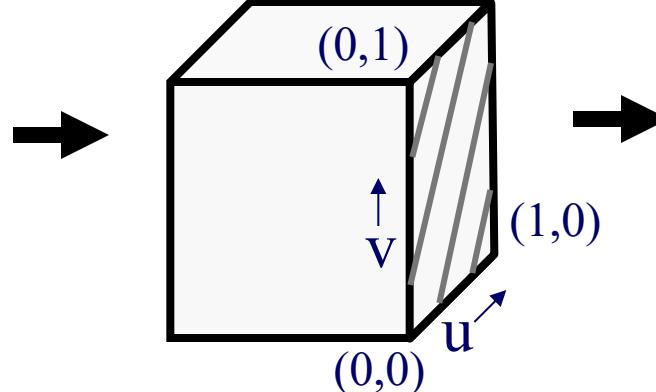


Texture Mapping

- Steps:
 1. Define texture image
 2. Specify mapping from texture to surface
 3. Look up texture values during scan conversion



Texture
Coordinate
System



Modeling
Coordinate
System

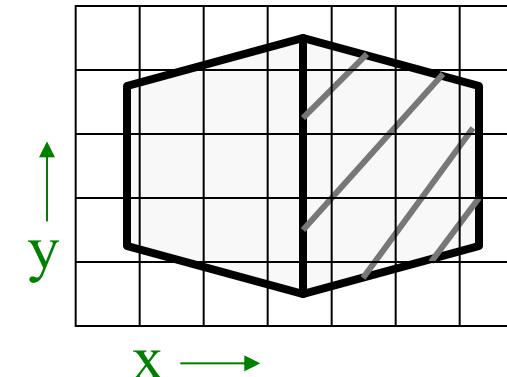
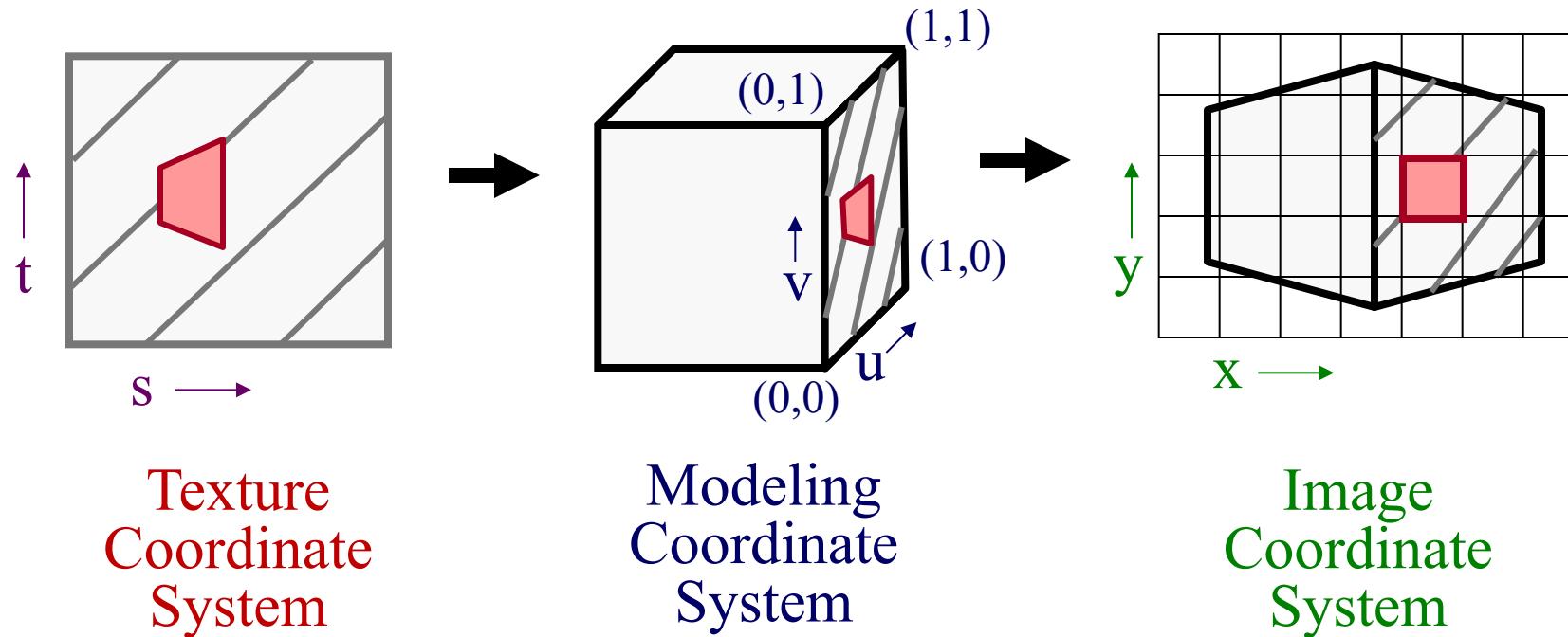


Image
Coordinate
System



Texture Mapping

- When scan converting, map from ...
 - image coordinate system (x,y) to
 - modeling coordinate system (u,v) to
 - texture image (s,t)





Texture Overview

- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering

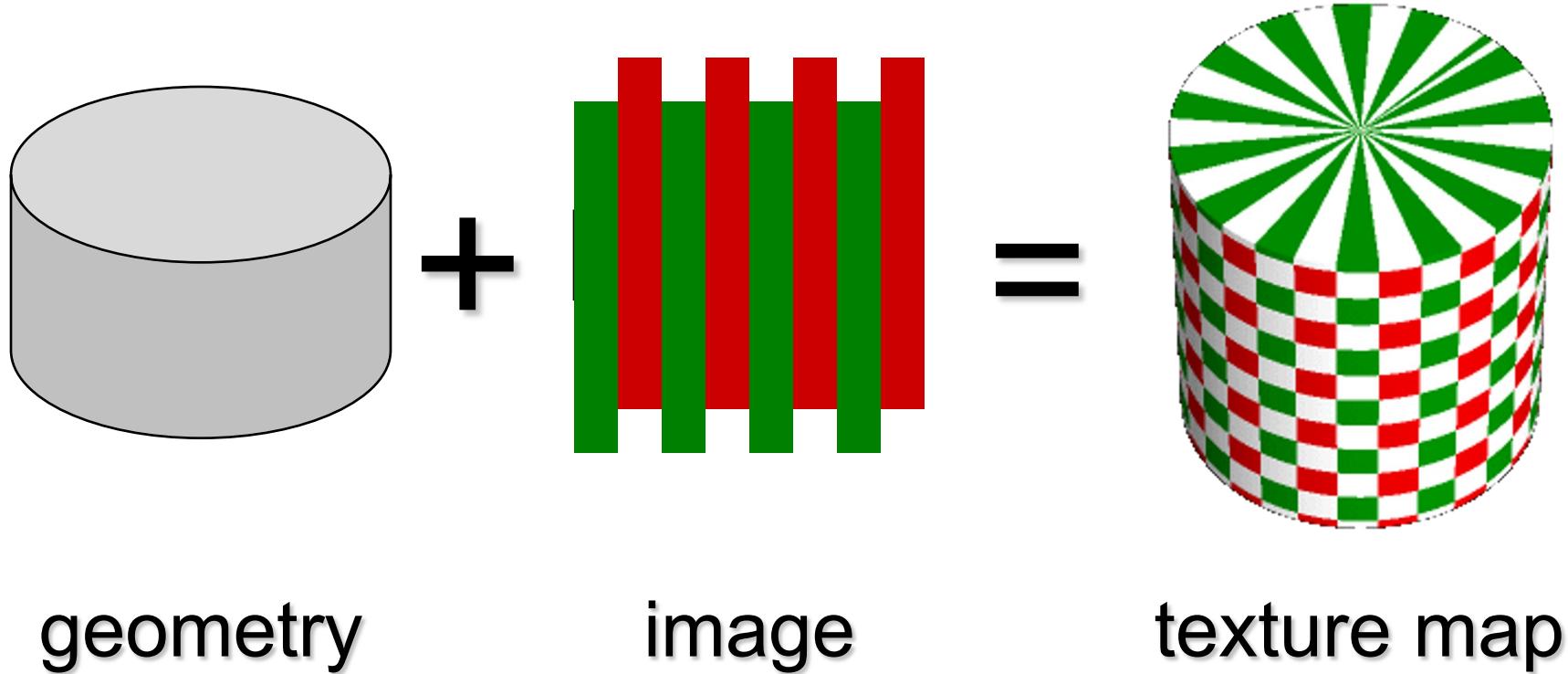


Texture Overview

- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering



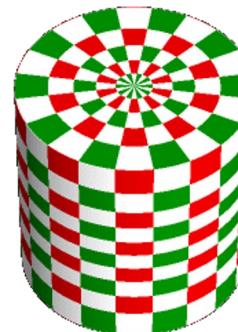
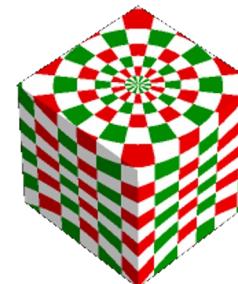
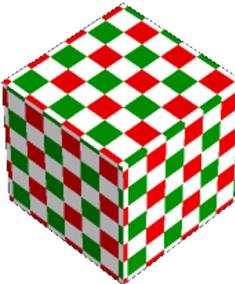
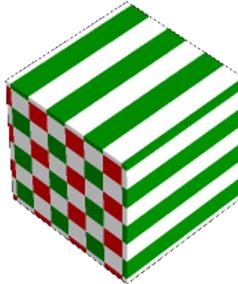
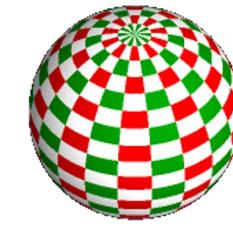
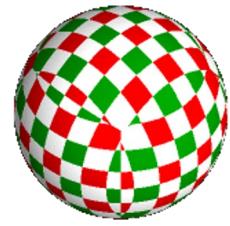
Texture Parameterization



- Q: How do we decide ***where*** on the geometry each color from the image should go?



Texture Parameterization

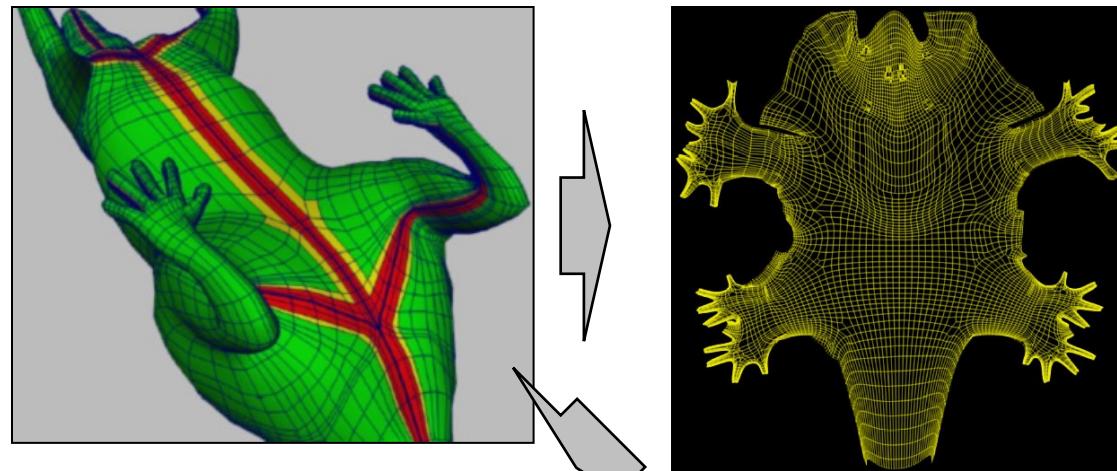


[Paul Bourke]

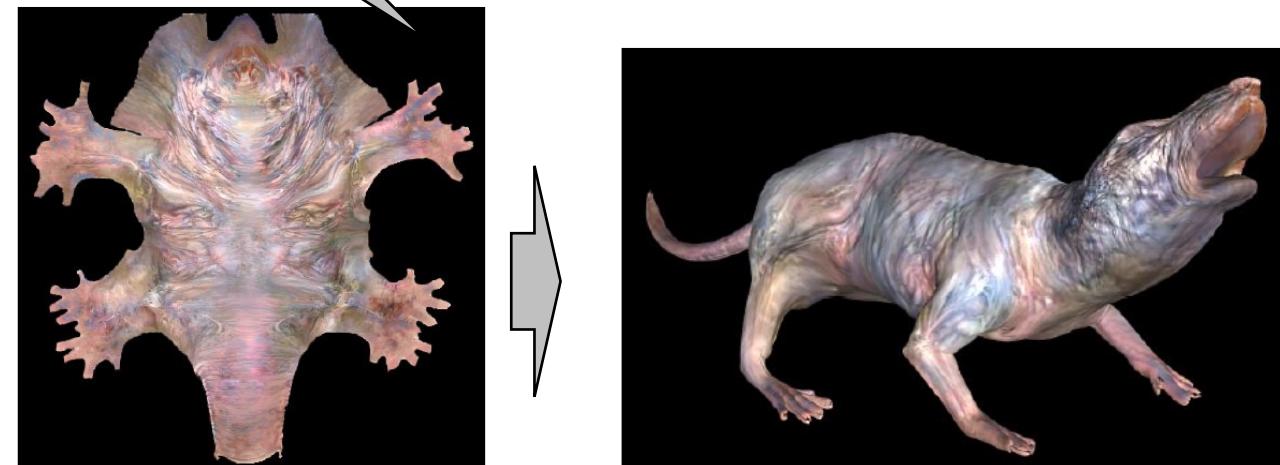


Texture Parameterization

Option1: unfold the surface



[Piponi2000]



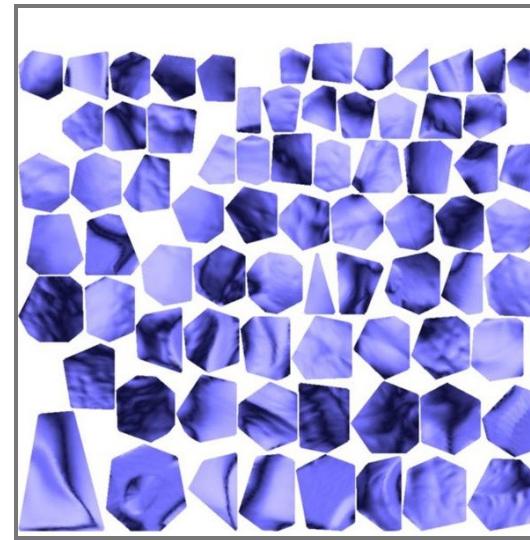


Texture Parameterization

Option2: make an atlas



charts



atlas



surface

[Sander2001]



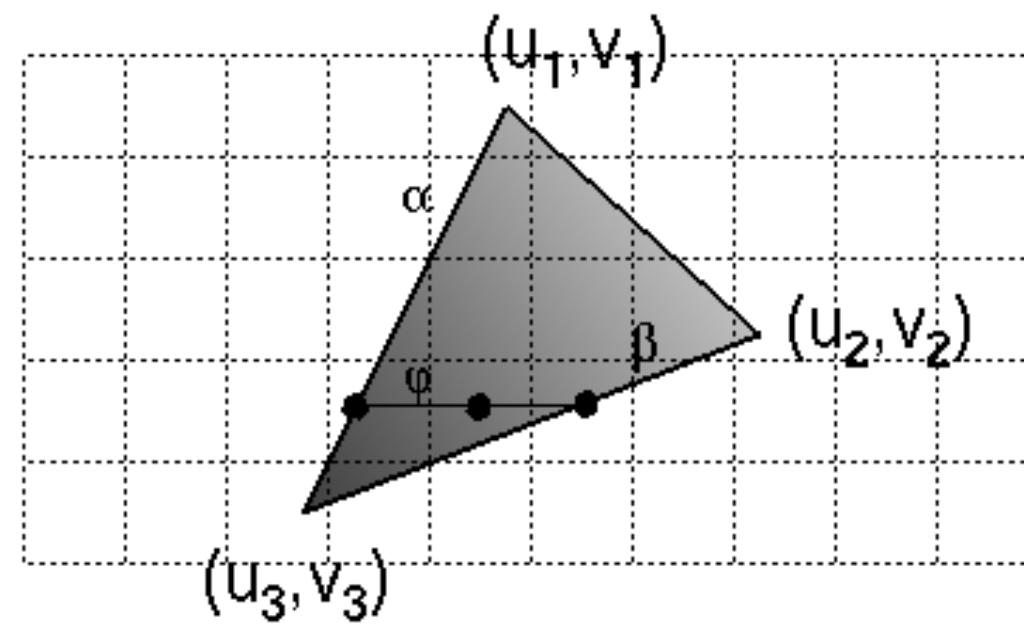
Texture Overview

- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering



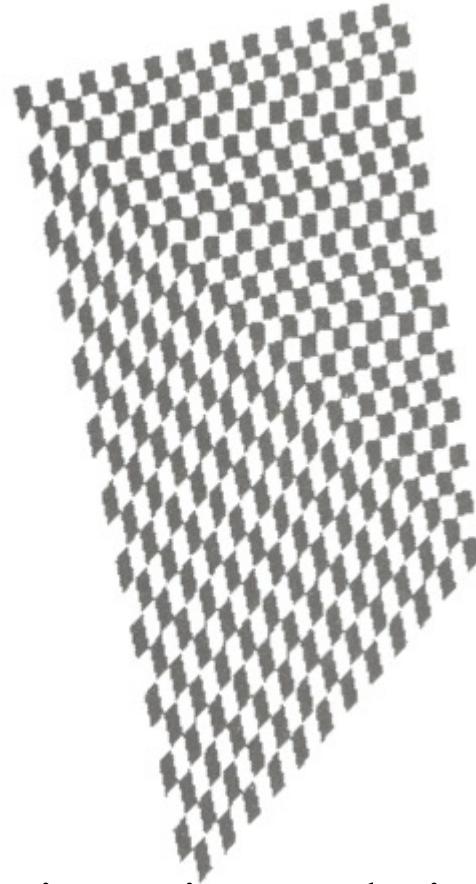
Texture Mapping

- Scan conversion
 - Interpolate texture coordinates down/across scan lines

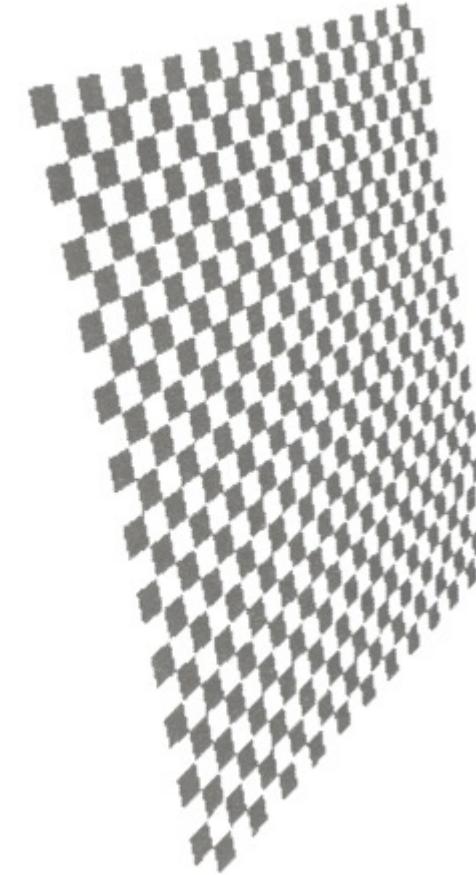




Perspective Divide



Linear interpolation
of texture coordinates

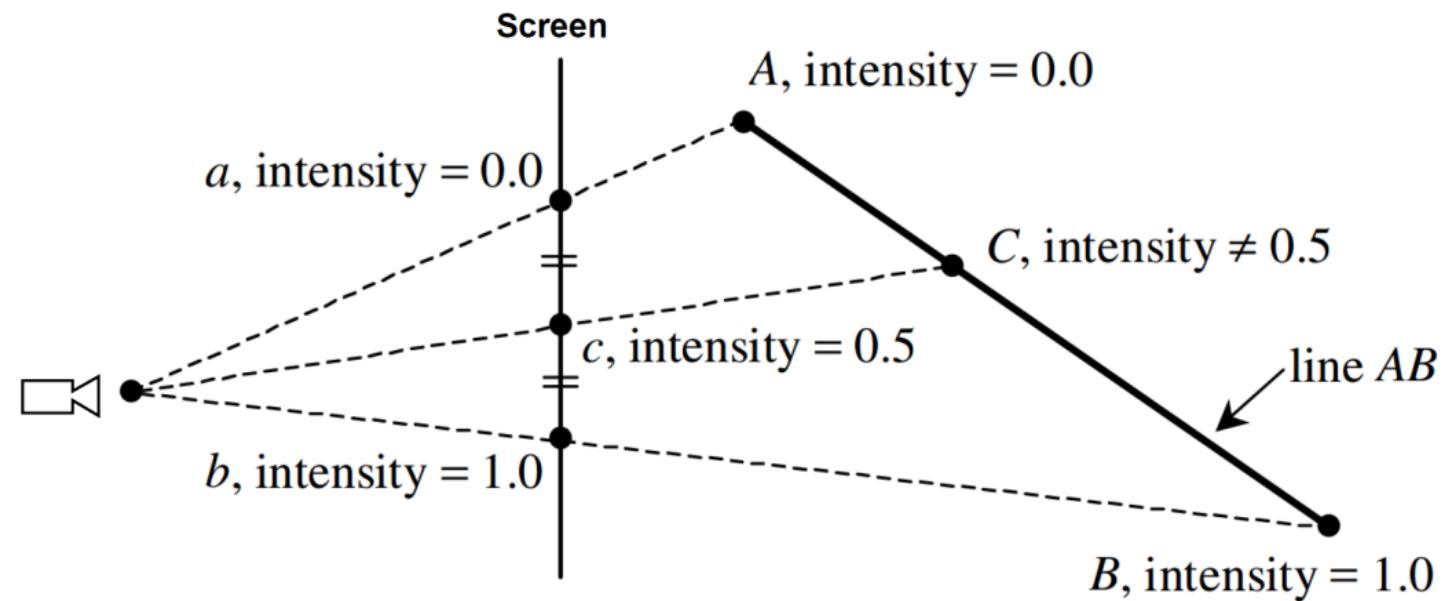


Correct interpolation

Hill Figure 8.42



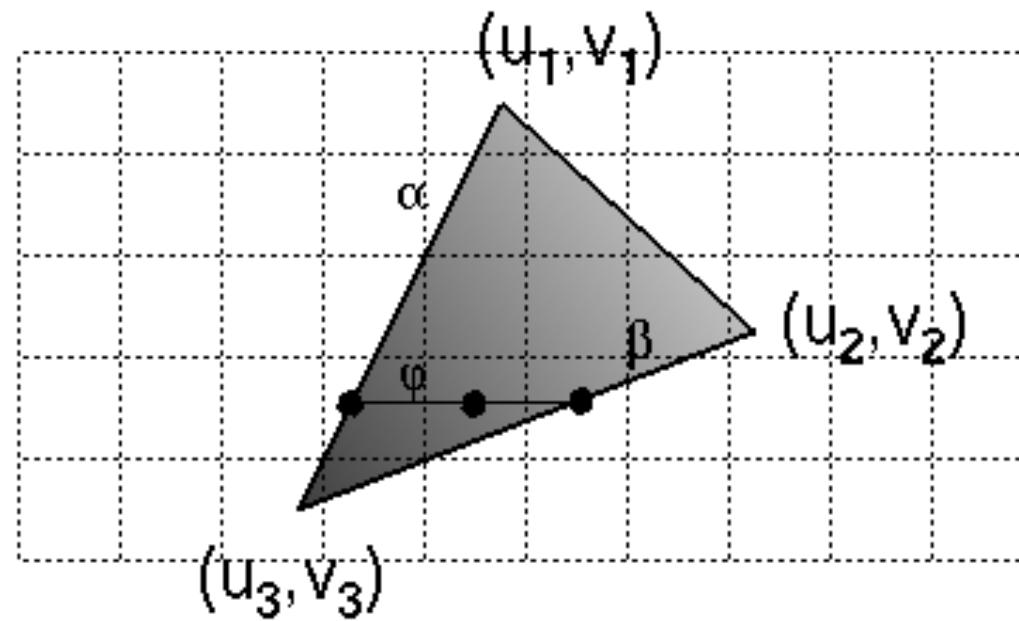
Perspective Divide





Texture Mapping

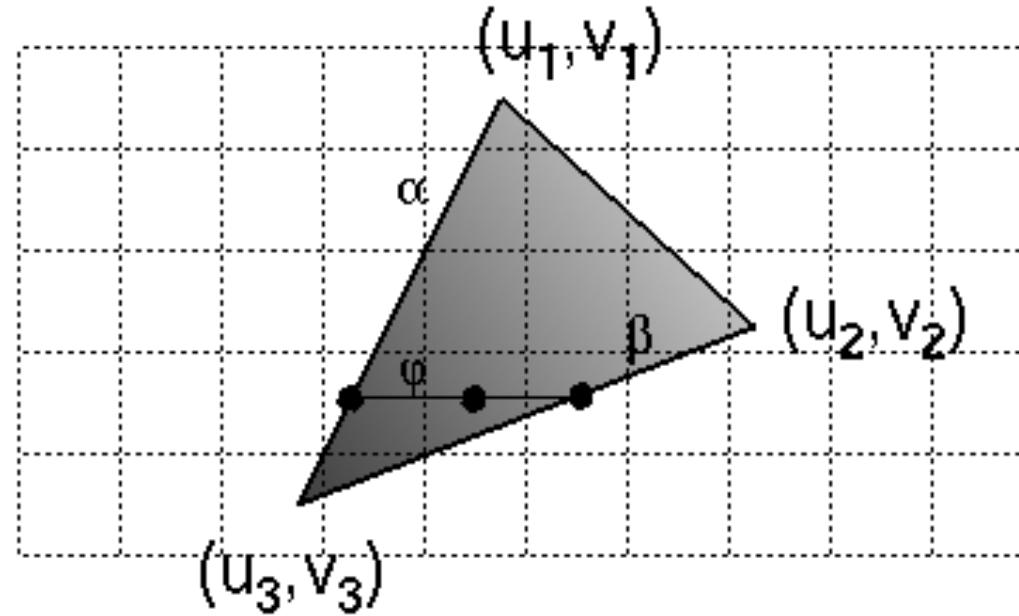
- Scan conversion
 - Interpolate texture coordinates down/across scan lines
 - Distortion due to bilinear interpolation approximation
 - » Cut polygons into smaller ones, or





Texture Mapping

- Scan conversion
 - Interpolate texture coordinates down/across scan lines
 - Distortion due to bilinear interpolation approximation
 - » Cut polygons into smaller ones, or
 - » Perspective divide at each pixel





Perspective Divide

Assume triangle attribute varies linearly across the triangle

Attribute's value at 3D (non-homogeneous) point $P = [x \ y \ z]^T$ is then:

$$f(x, y, z) = ax + by + cz$$

Get 2D homogeneous representation : $[x_{2D-H} \ y_{2D-H} \ w]^T = [x \ y \ z]^T$



Perspective Divide

Assume triangle attribute varies linearly across the triangle

Attribute's value at 3D (non-homogeneous) point $P = [x \ y \ z]^T$ is then:

$$f(x, y, z) = ax + by + cz$$

Get 2D homogeneous representation : $[x_{2D-H} \ y_{2D-H} \ w]^T = [x \ y \ z]^T$

Rewrite attribute equation for f in terms of 2D homogeneous coordinates:

$$f = ax_{2D-H} + by_{2D-H} + cw$$



Perspective Divide

Assume triangle attribute varies linearly across the triangle

Attribute's value at 3D (non-homogeneous) point $P = [x \ y \ z]^T$ is then:

$$f(x, y, z) = ax + by + cz$$

Get 2D homogeneous representation : $[x_{2D-H} \ y_{2D-H} \ w]^T = [x \ y \ z]^T$

Rewrite attribute equation for f in terms of 2D homogeneous coordinates:

$$f = ax_{2D-H} + by_{2D-H} + cw$$

$$\frac{f}{w} = a\frac{x_{2D-H}}{w} + b\frac{y_{2D-H}}{w} + c$$



Perspective Divide

Assume triangle attribute varies linearly across the triangle

Attribute's value at 3D (non-homogeneous) point $P = [x \ y \ z]^T$ is then:

$$f(x, y, z) = ax + by + cz$$

Get 2D homogeneous representation : $[x_{2D-H} \ y_{2D-H} \ w]^T = [x \ y \ z]^T$

Rewrite attribute equation for f in terms of 2D homogeneous coordinates:

$$f = ax_{2D-H} + by_{2D-H} + cw$$

$$\frac{f}{w} = a\frac{x_{2D-H}}{w} + b\frac{y_{2D-H}}{w} + c$$

$$\frac{f}{w} = ax_{2D} + by_{2D} + c$$

Where $[x_{2D} \ y_{2D}]^T$ are projected screen 2D coordinates (after homogeneous divide)



Perspective Divide

Assume triangle attribute varies linearly across the triangle

Attribute's value at 3D (non-homogeneous) point $P = [x \ y \ z]^T$ is then:

$$f(x, y, z) = ax + by + cz$$

Get 2D homogeneous representation : $[x_{2D-H} \ y_{2D-H} \ w]^T = [x \ y \ z]^T$

Rewrite attribute equation for f in terms of 2D homogeneous coordinates:

$$f = ax_{2D-H} + by_{2D-H} + cw$$

$$\frac{f}{w} = a\frac{x_{2D-H}}{w} + b\frac{y_{2D-H}}{w} + c$$

$$\frac{f}{w} = ax_{2D} + by_{2D} + c$$

Where $[x_{2D} \ y_{2D}]^T$ are projected screen 2D coordinates (after homogeneous divide)

So ... $\frac{f}{w}$ is affine function of 2D screen coordinates...

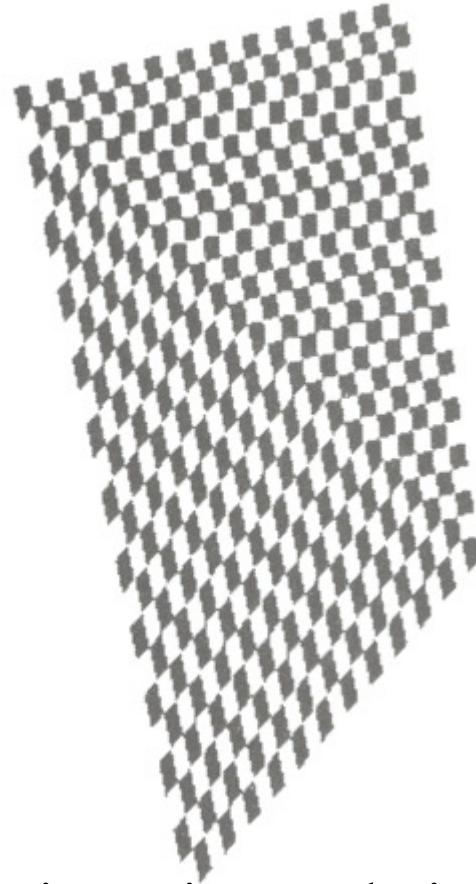


Perspective Divide

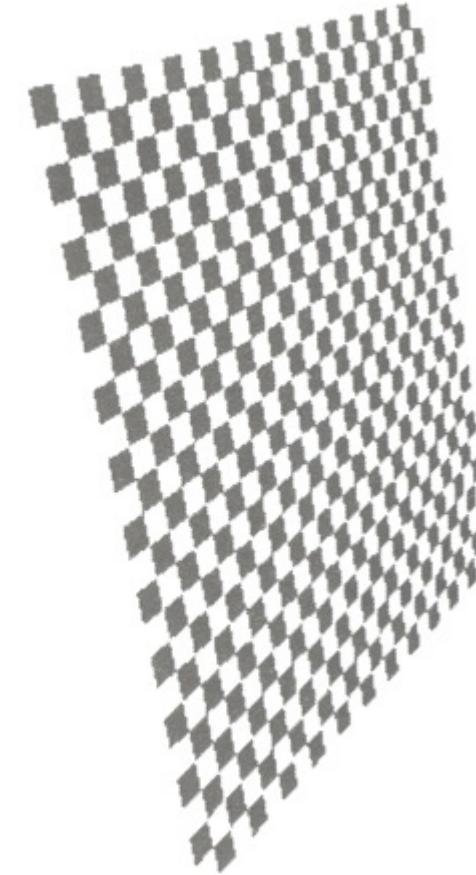
- Compute at each vertex after perspective transformation:
 - “Numerators” s/w , t/w
 - “Denominator” $1/w$
- Linearly interpolate s/w , and t/w and $1/w$ across the polygon
- At each pixel:
 - Perform perspective division of interpolated texture coordinates (s/w , t/w) by interpolated $1/w$ (i.e., numerator over denominator) to get (s, t)



Perspective Divide



Linear interpolation
of texture coordinates



Correct interpolation

Hill Figure 8.42



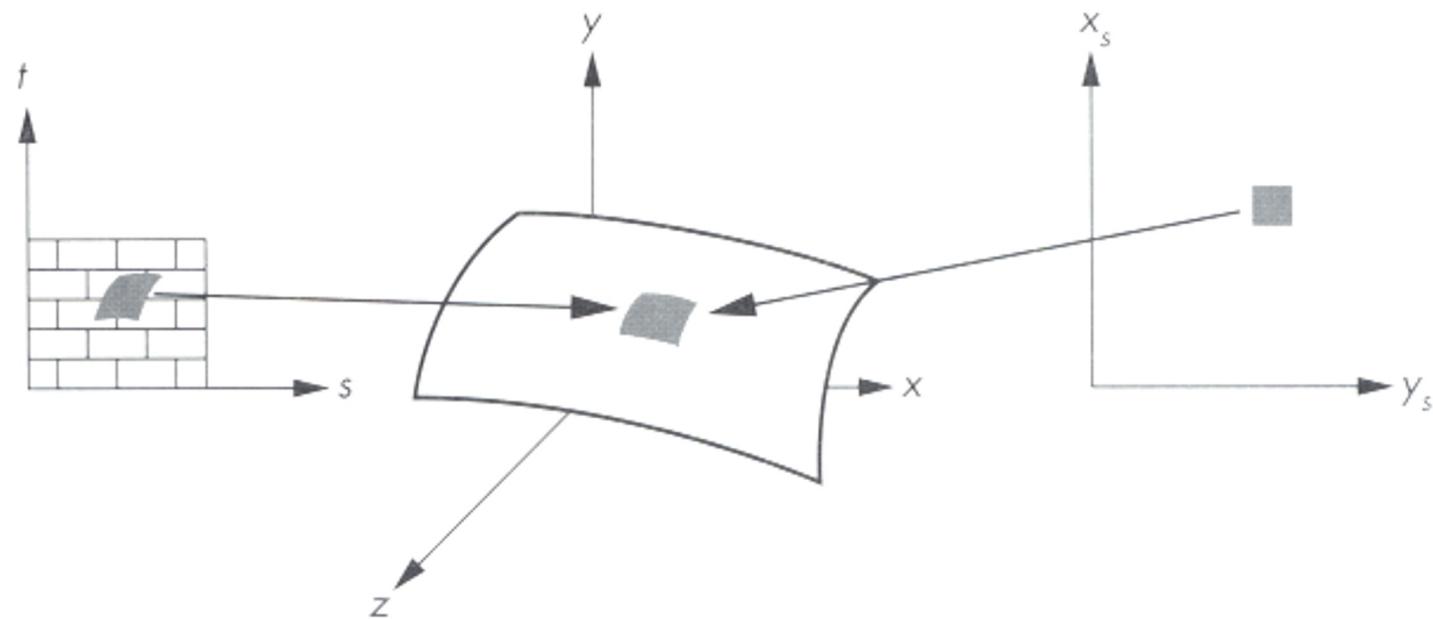
Texture Overview

- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering



Texture Filtering

- Must **sample** texture to determine color at each pixel in image

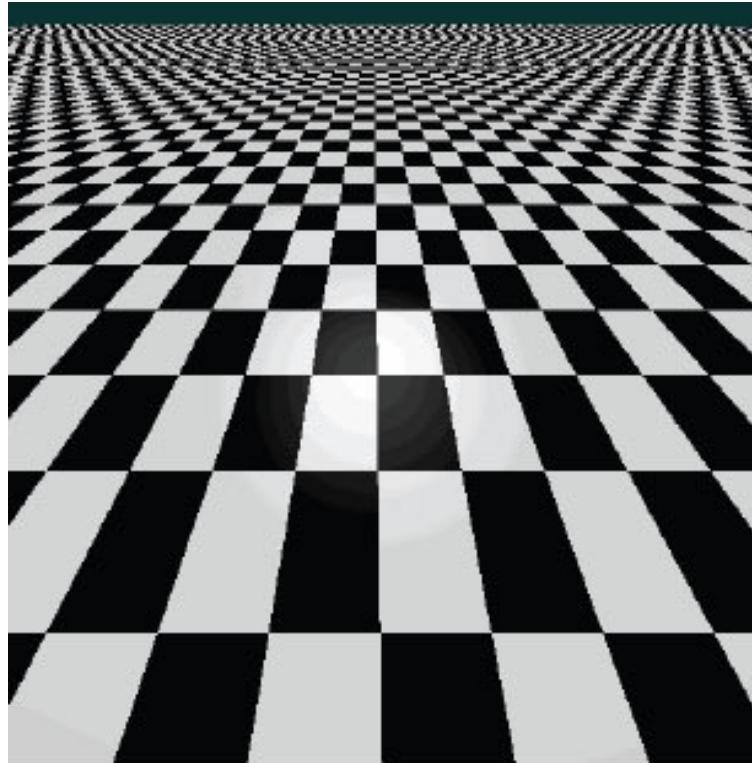


Angel Figure 9.4

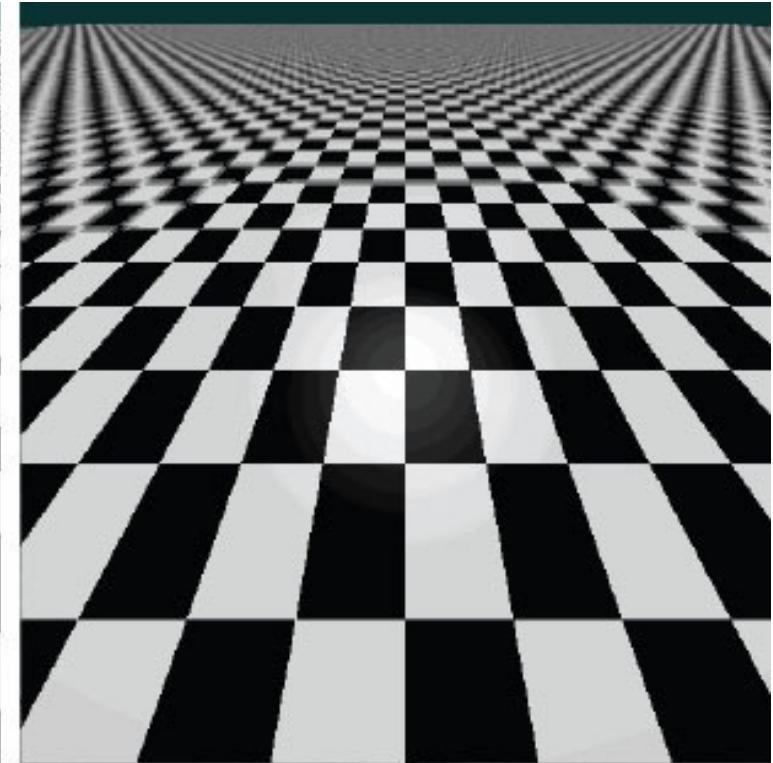


Texture Filtering

- Aliasing is a problem



Point sampling

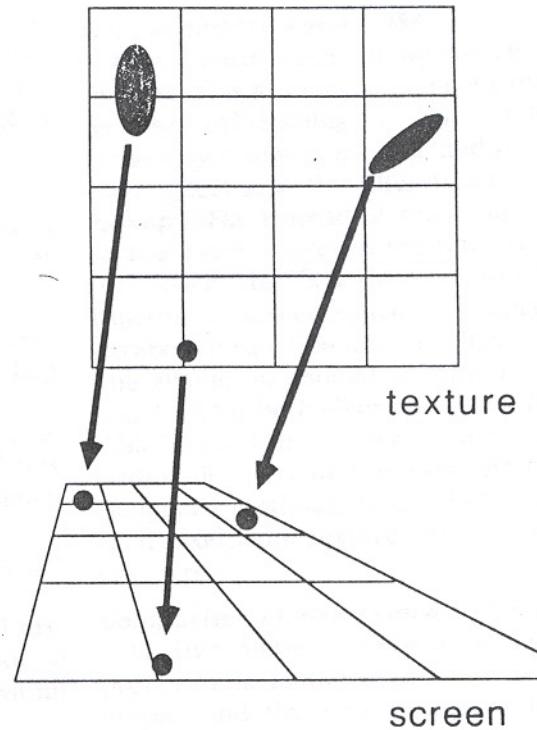


Area filtering



Texture Filtering

- Ideally, use elliptically shaped convolution filters

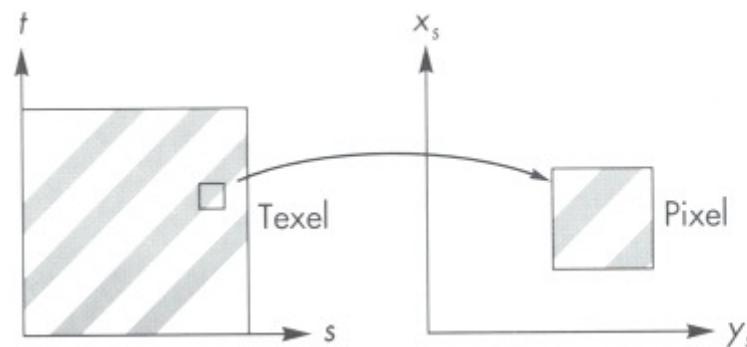


In practice, *use rectangles or squares*

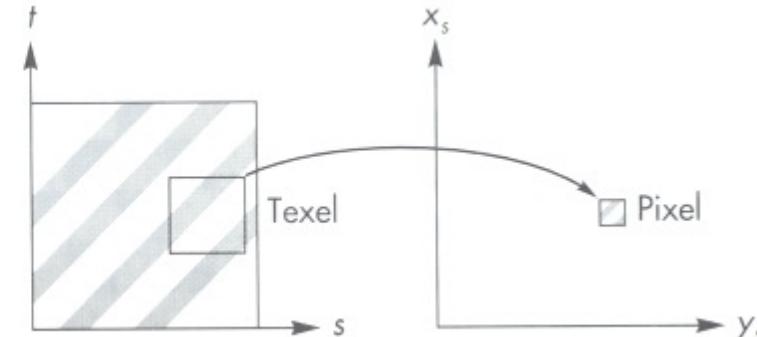


Texture Filtering

- Size of filter depends on projective warp
 - Compute prefiltered images to avoid run-time cost
 - » Mipmaps
 - » Summed area tables



Magnification



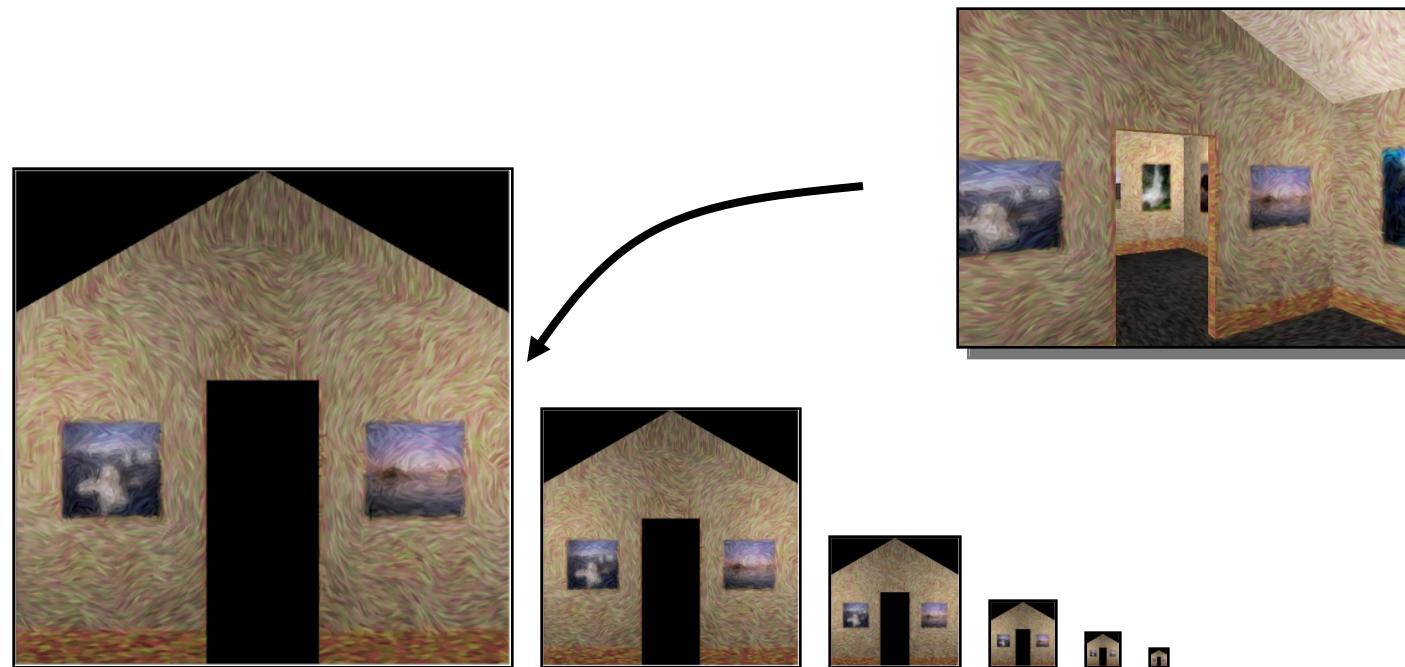
Minification

Angel Figure 9.14



Mipmaps

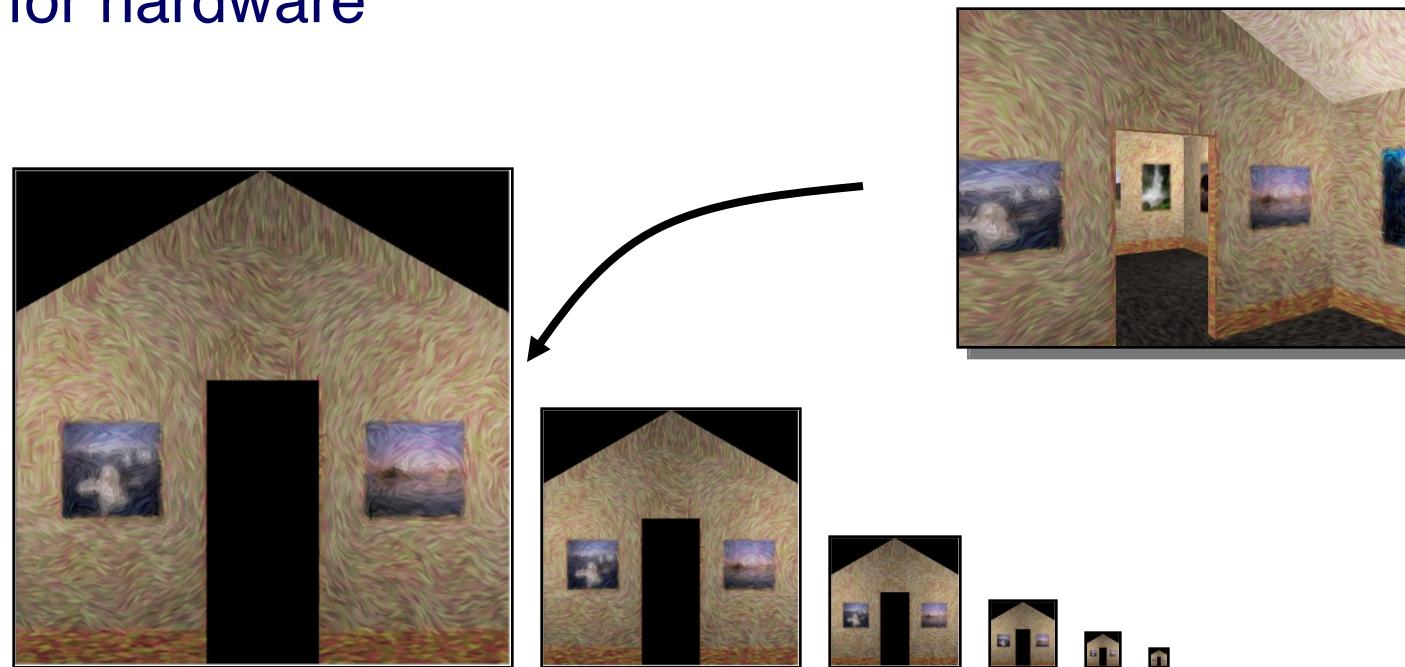
- Keep textures prefiltered at multiple resolutions
 - Usually powers of 2





Mipmaps

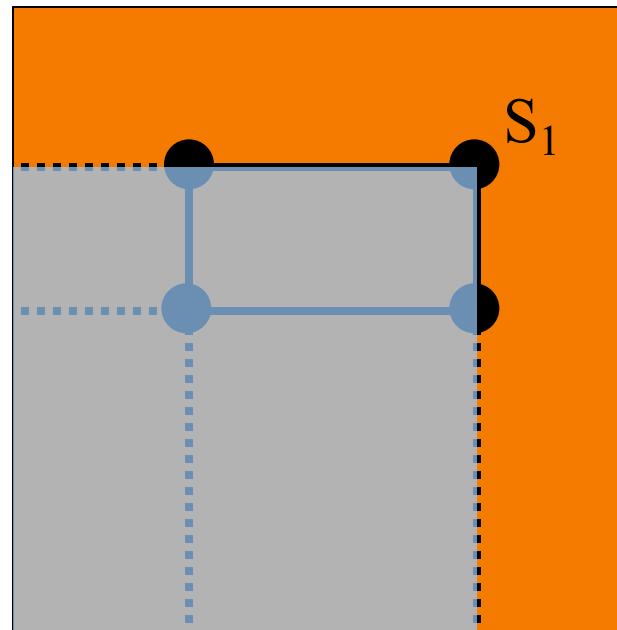
- Keep textures prefiltered at multiple resolutions
 - Usually powers of 2
 - For each pixel, linearly interpolate between two closest levels (i.e., trilinear filtering)
 - Fast, easy for hardware





Summed-area tables

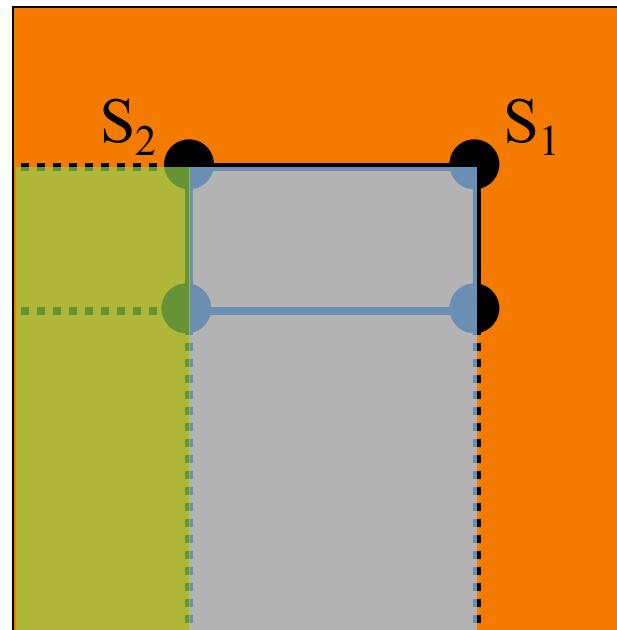
- At each texel keep sum of all values down & left
 - To compute sum of all values within a rectangle, simply combine four entries: S_1





Summed-area tables

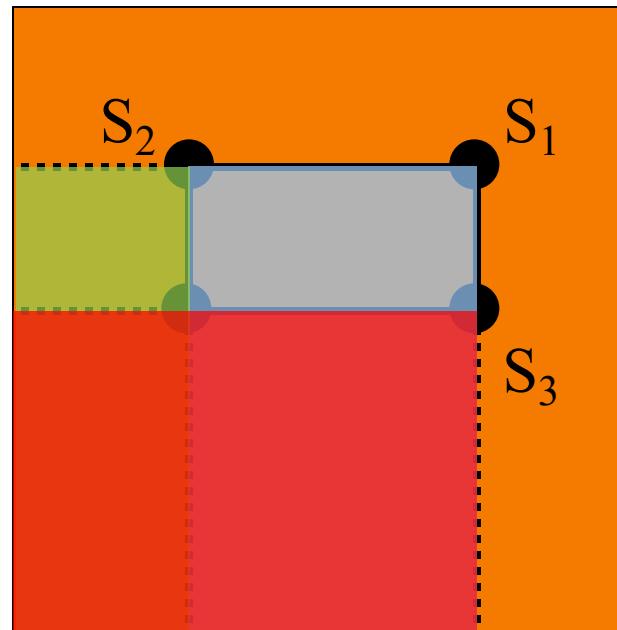
- At each texel keep sum of all values down & left
 - To compute sum of all values within a rectangle, simply combine four entries: $S_1 - S_2$





Summed-area tables

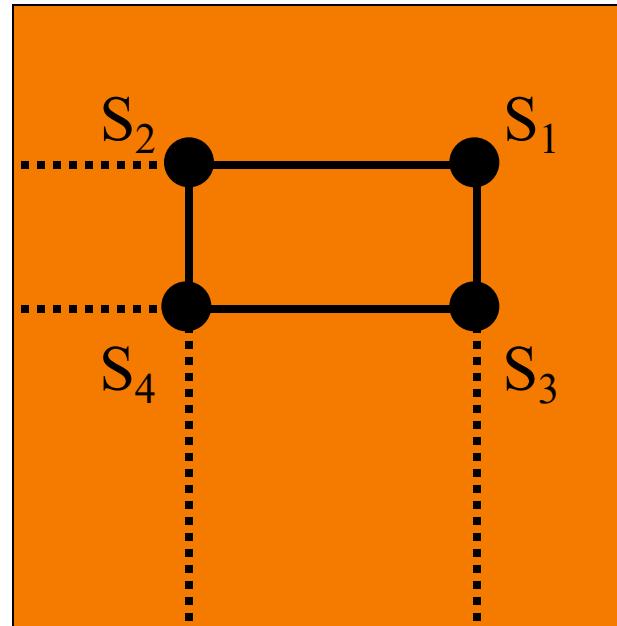
- At each texel keep sum of all values down & left
 - To compute sum of all values within a rectangle, simply combine four entries: $S_1 - S_2 - S_3 + S_4$





Summed-area tables

- At each texel keep sum of all values down & left
 - To compute sum of all values within a rectangle, simply combine four entries: $S_1 - S_2 - S_3 + S_4$
 - Better ability to capture oblique projections, but still not perfect



- (Mipmaps are more common.)



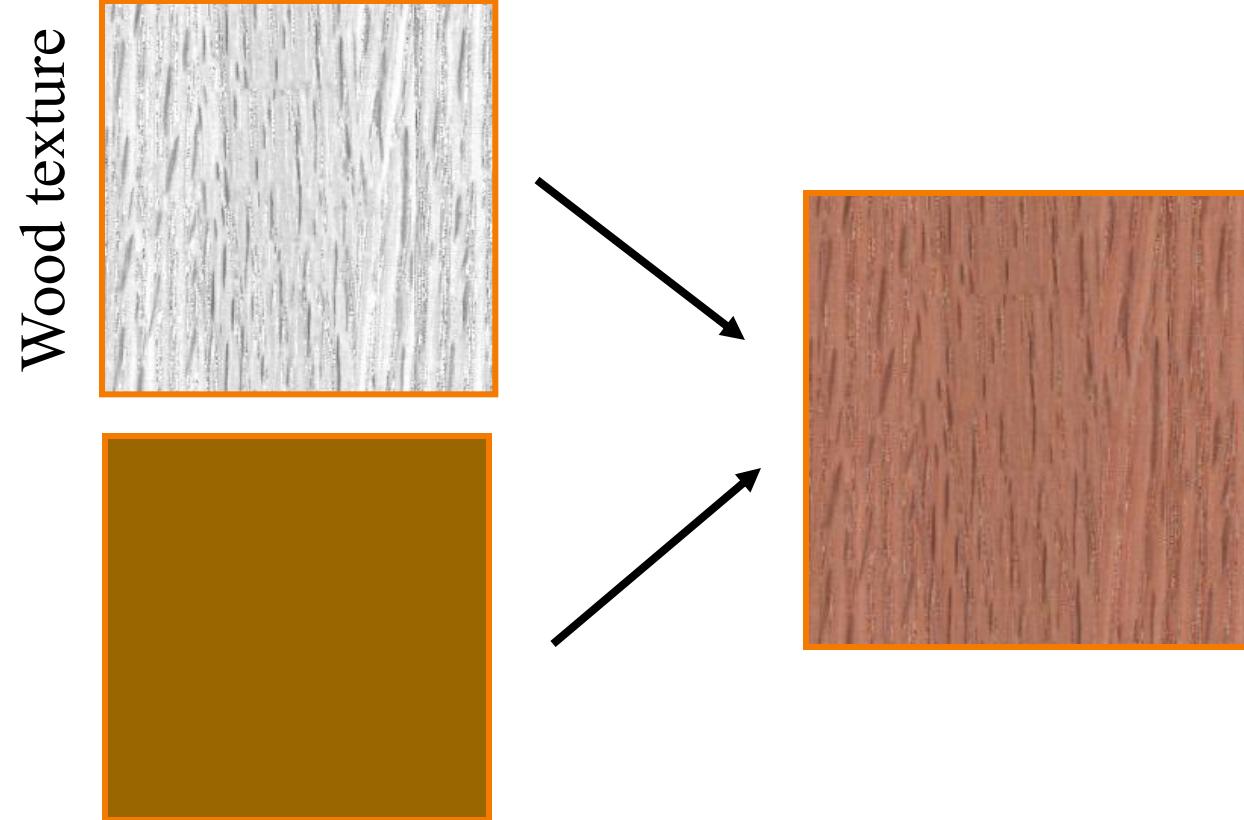
Texture Overview

- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering



Modulation textures

- Texture values scale result of lighting calculation



$$I = T(s, t) \left(I_E + K_A I_A + \sum_L \left(K_D (N \cdot L) + K_S (V \cdot R)^n \right) S_L I_L + K_T I_T + K_S I_S \right)$$



Illumination Mapping

- Map texture values to surface material parameter
 - K_A
 - K_D
 - K_S
 - K_T
 - n



Texture
value

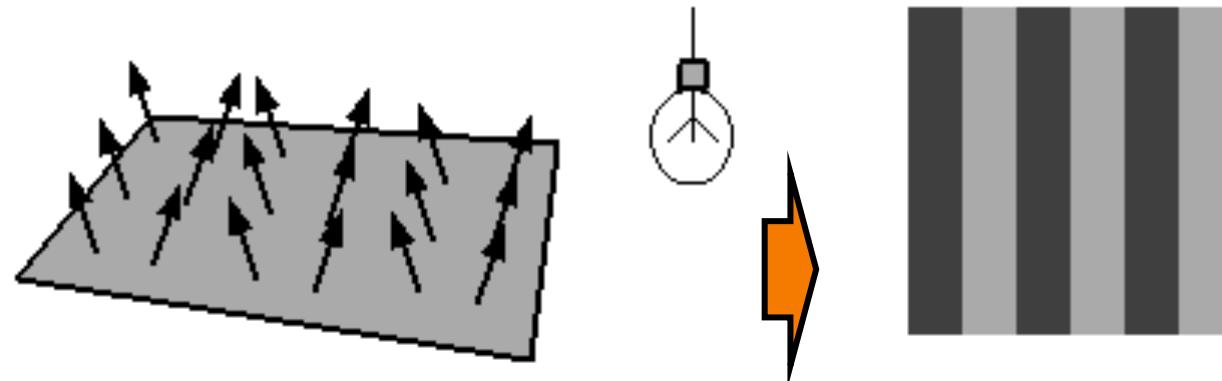


$$I = I_E + K_A I_A + \sum_L \left(K_D(s, t)(N \cdot L) + K_S(V \cdot R)^n \right) S_L I_L + K_T I_T + K_S I_S$$



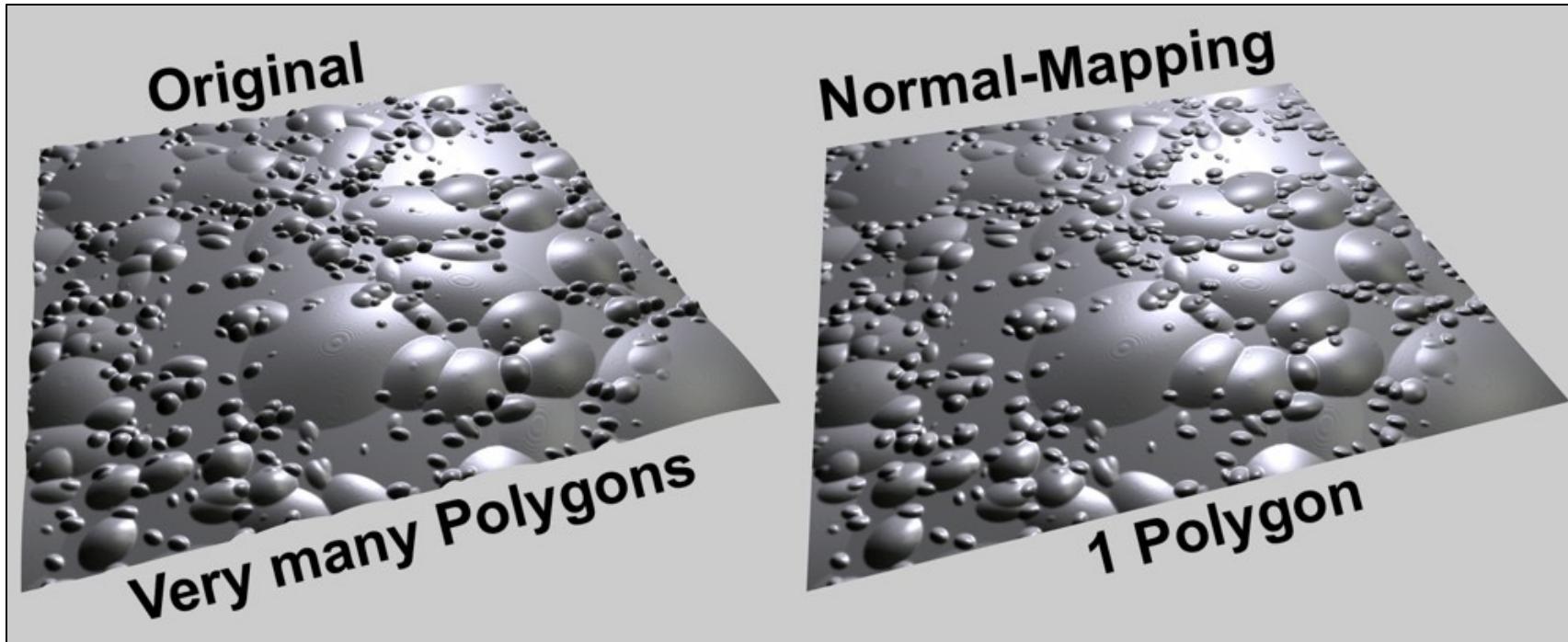
Bump/Normal Mapping

- Texture values determine or perturb surface normals:
 - Encode normals in RGB ($R \rightarrow N_x$, $G \rightarrow N_y$, $B \rightarrow N_z$, $0..255 \rightarrow -1..1$)
 - Or encode normal **offsets** in RGB
 - Or use gradient of grayscale image as normal offset (“bump mapping”)



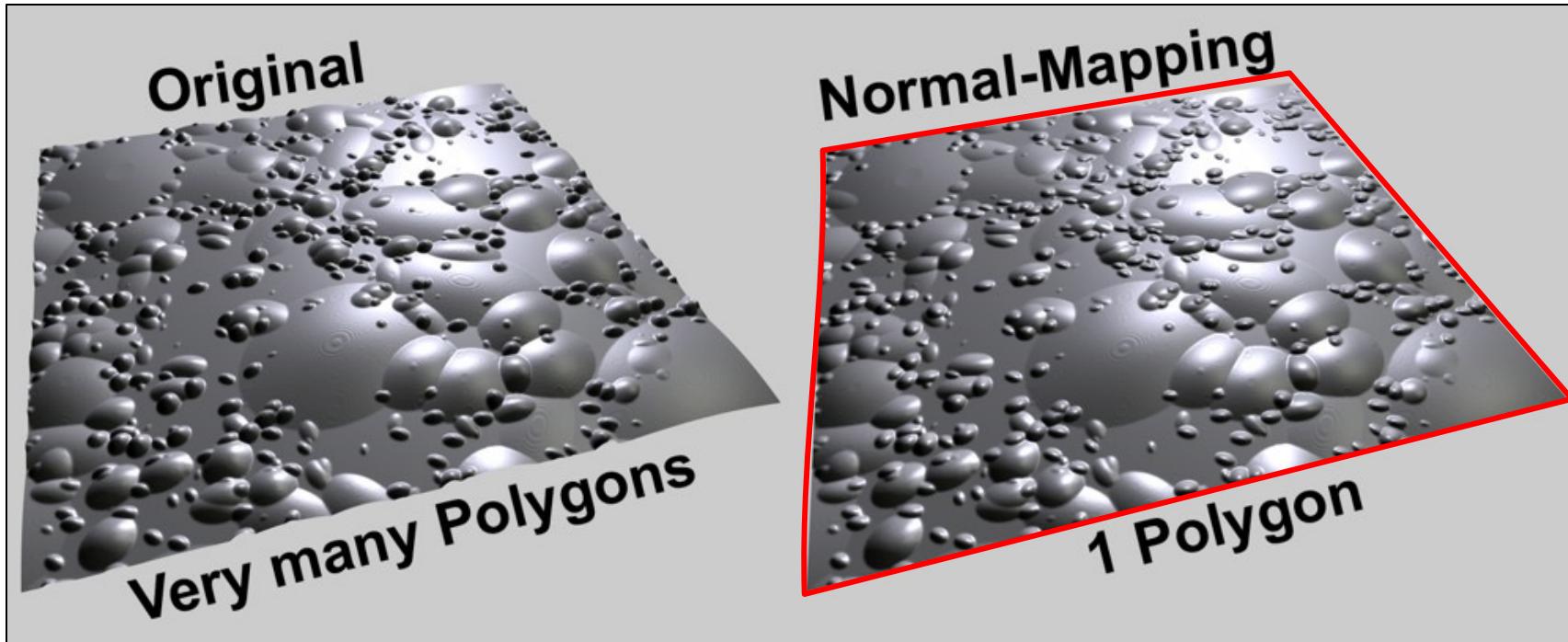


Normal Mapping





Normal Mapping



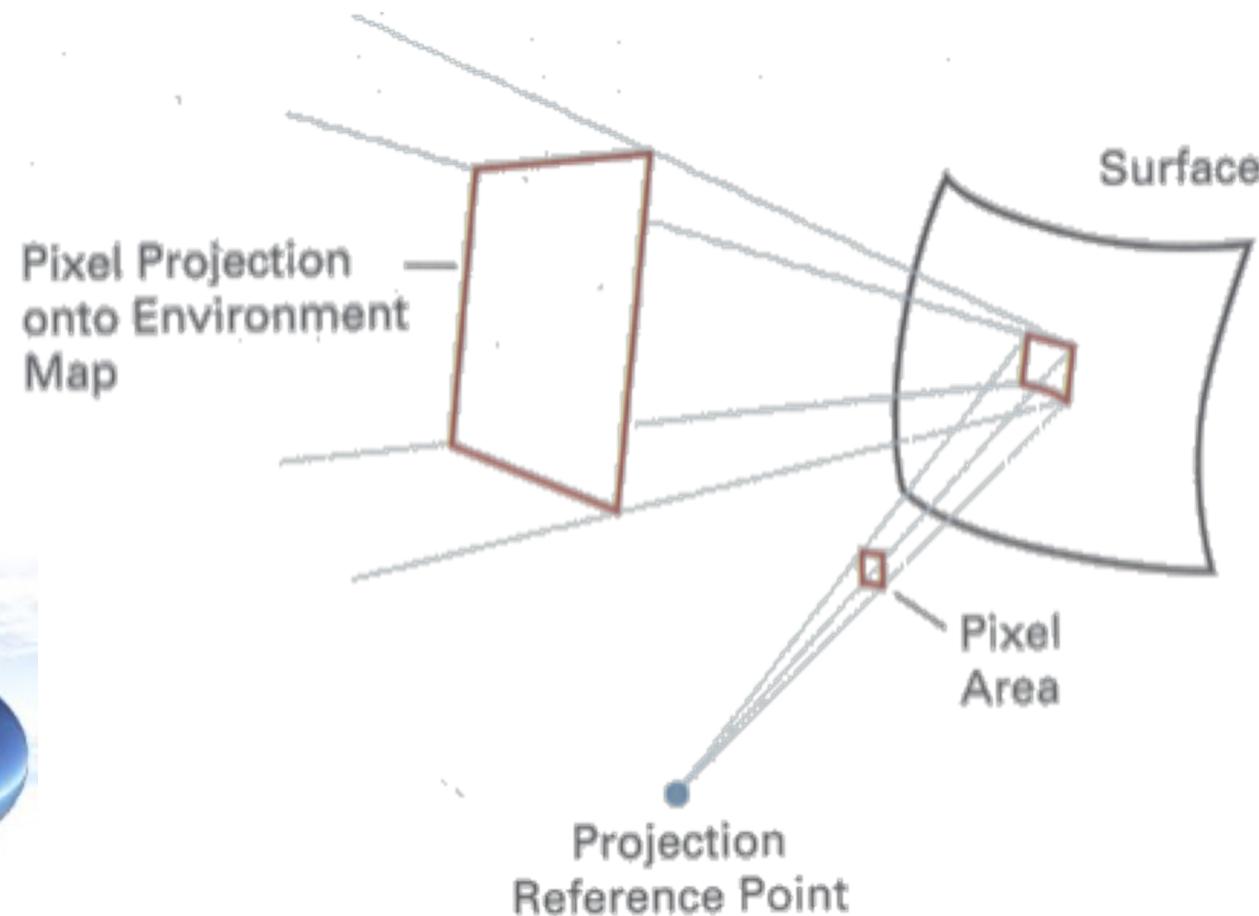


Environment Mapping

- Texture values are reflected off surface patch



Gamer3D/Wikipedia

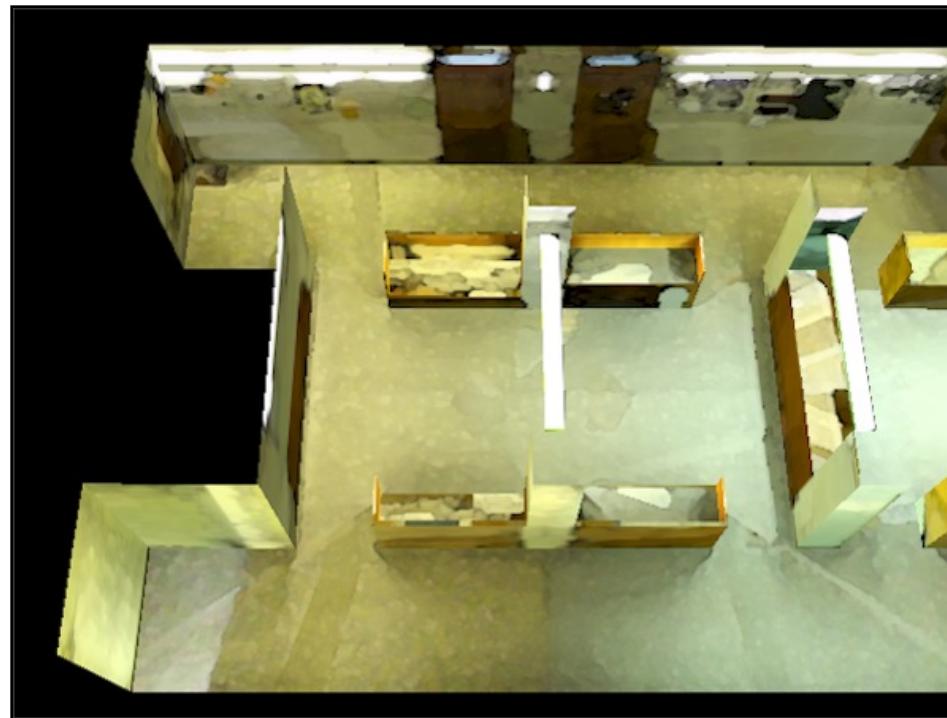


H&B Figure 14.93



Image-Based Rendering

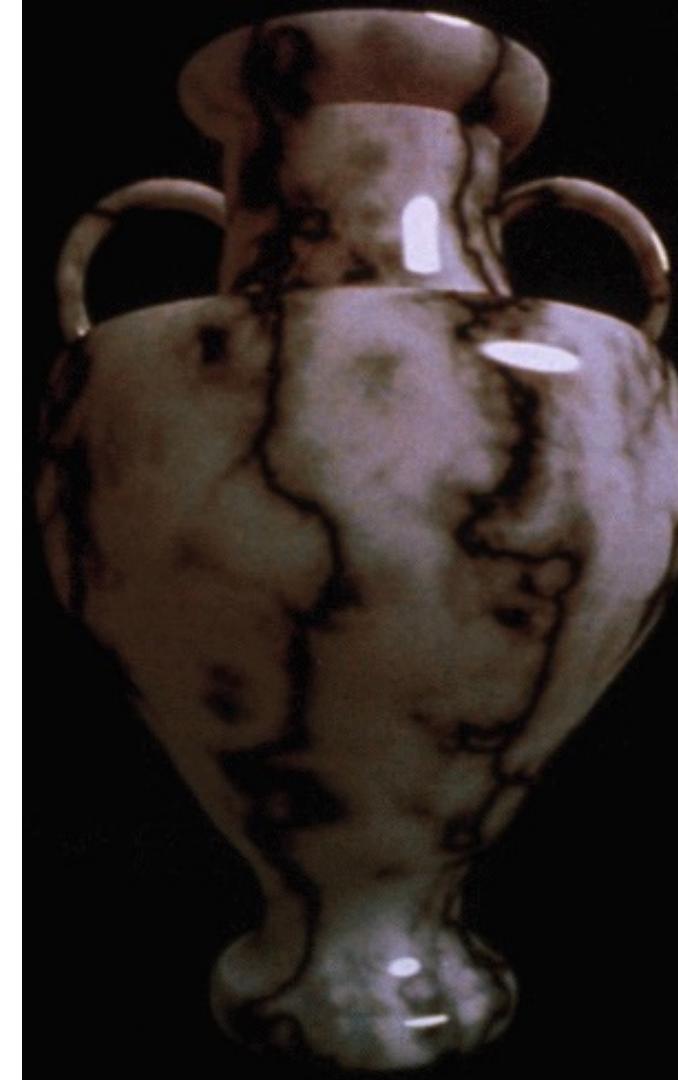
- Map photographic textures to provide details for coarsely detailed polygonal model





Solid textures

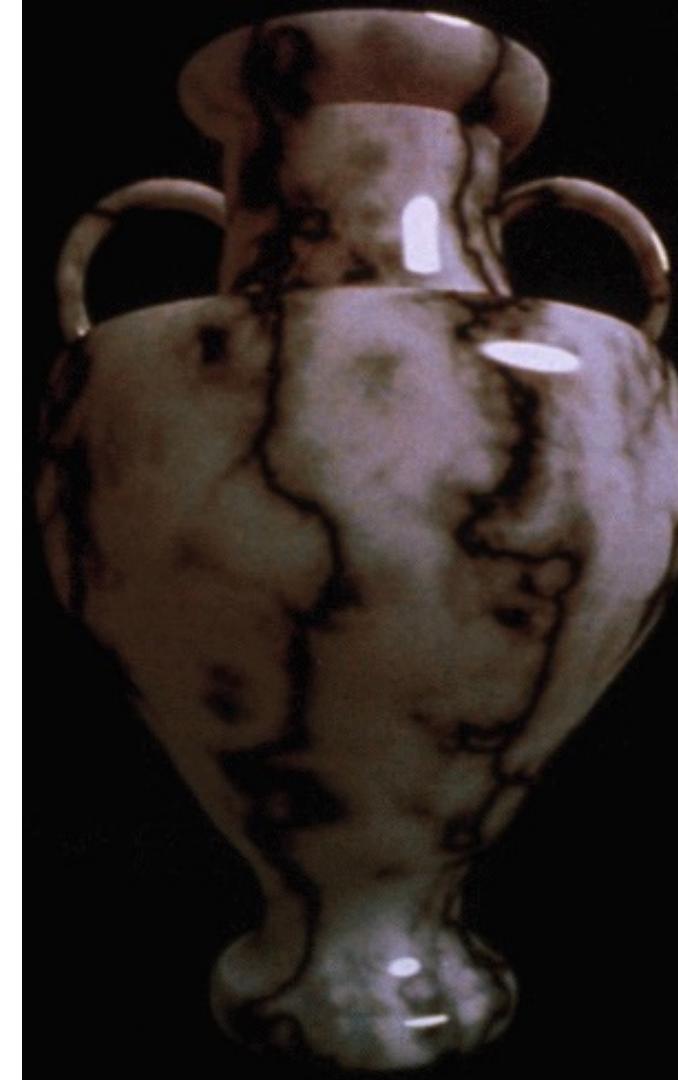
- Texture values indexed by 3D location (x,y,z)
 - Expensive storage, or





Solid textures

- Texture values indexed by 3D location (x,y,z)
 - Expensive storage, or
 - Compute on the fly,
e.g. Perlin noise →





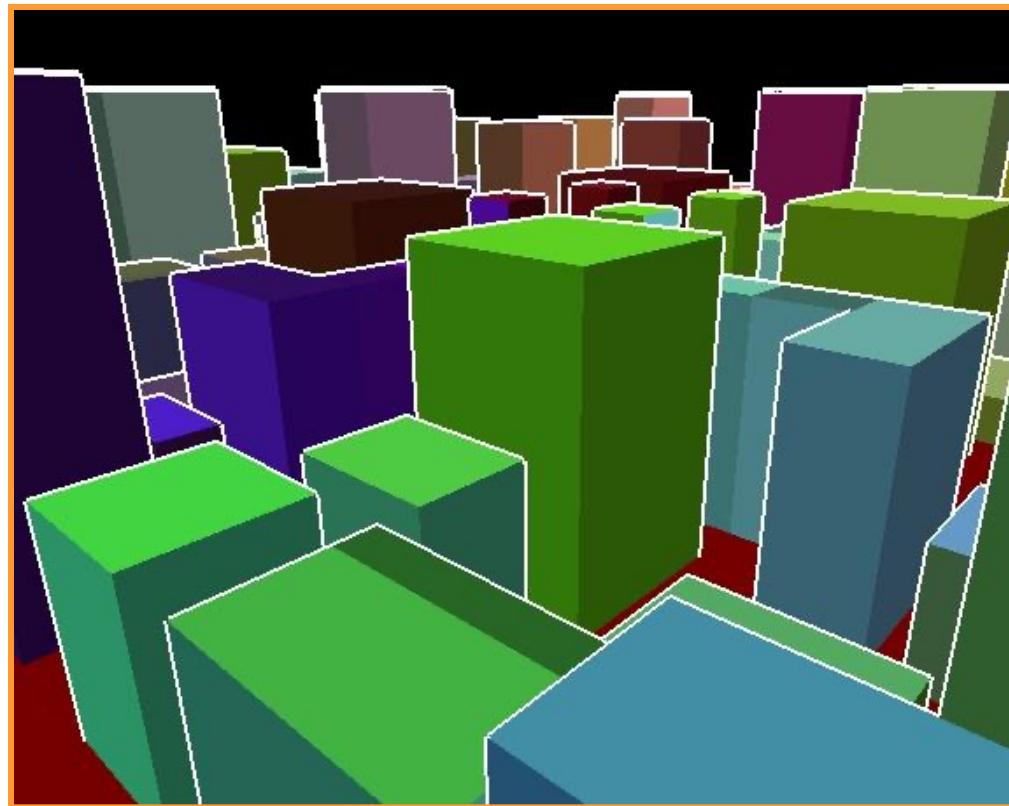
Rasterization

- Scan conversion
 - Determine which pixels to fill
 - Shading
 - Determine a color for each filled pixel
 - Texture mapping
 - Describe shading variation within polygon interiors
- **Visible surface determination**
- Figure out which surface is front-most at every pixel



Visible Surface Determination

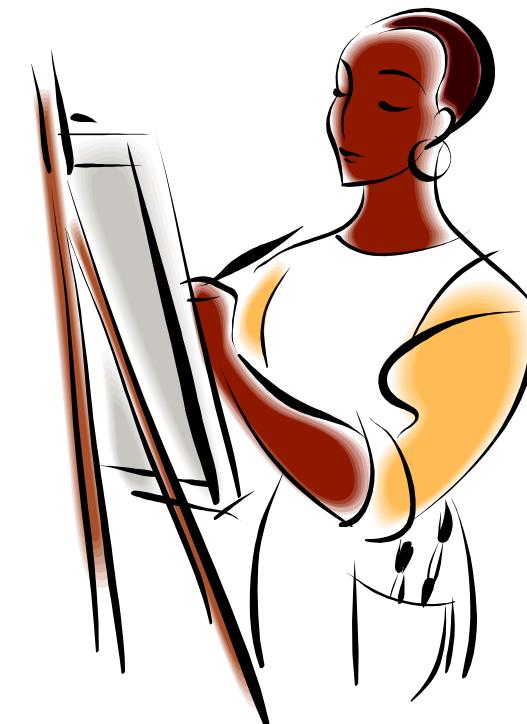
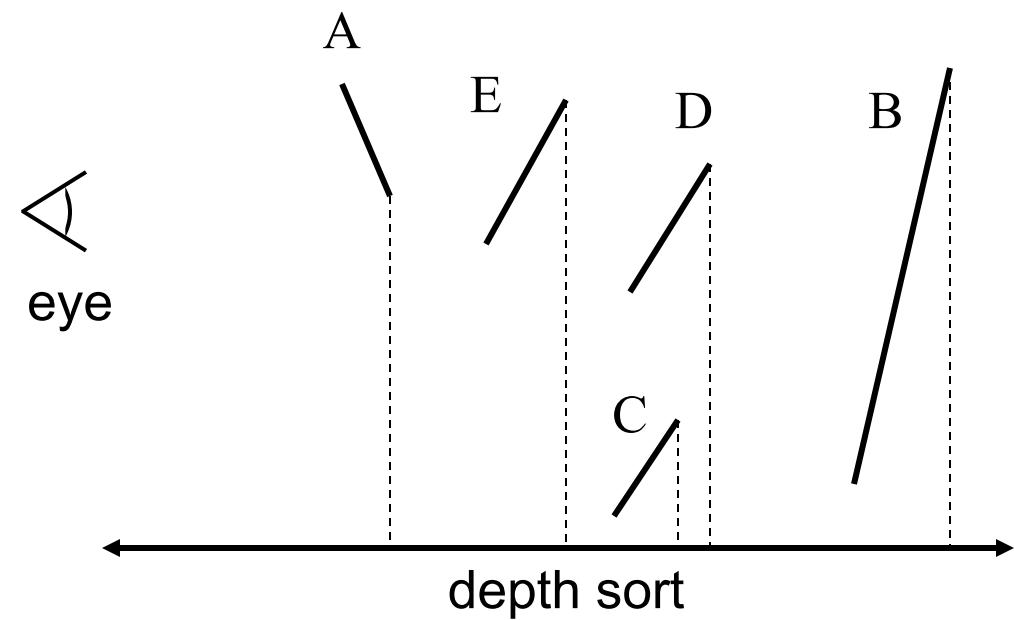
- Make sure only front-most surface contributes to color at every pixel





Depth sort

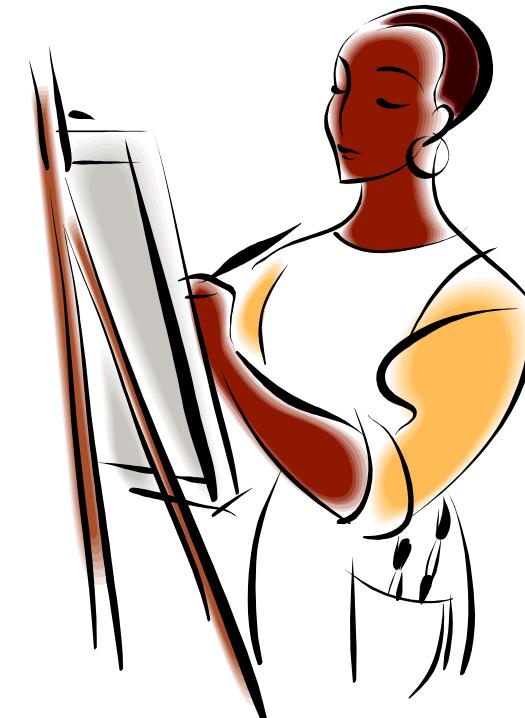
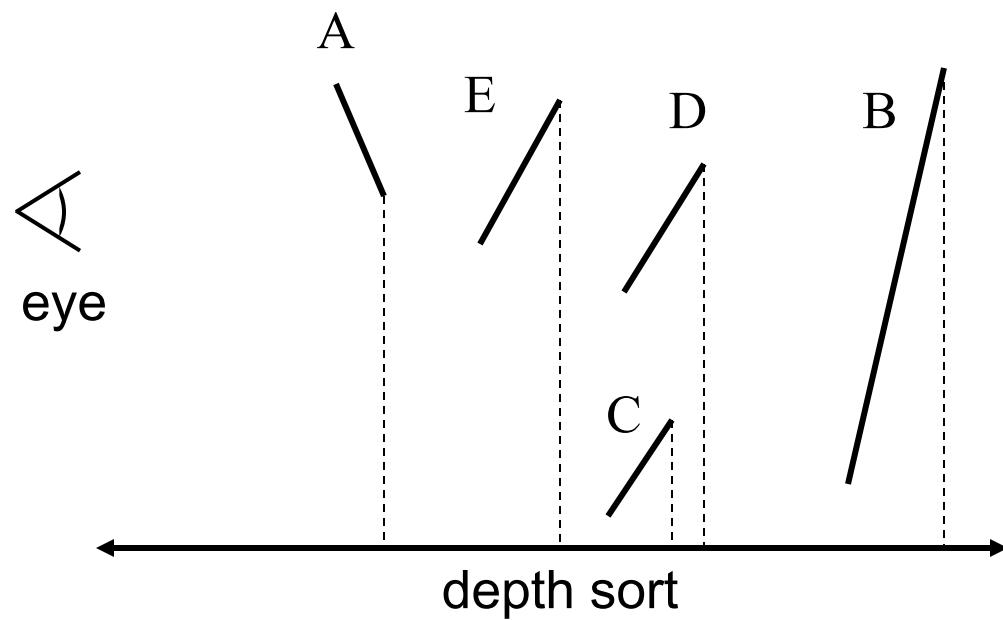
- “Painter’s algorithm”
 - First sort surfaces in order of decreasing **maximum depth**





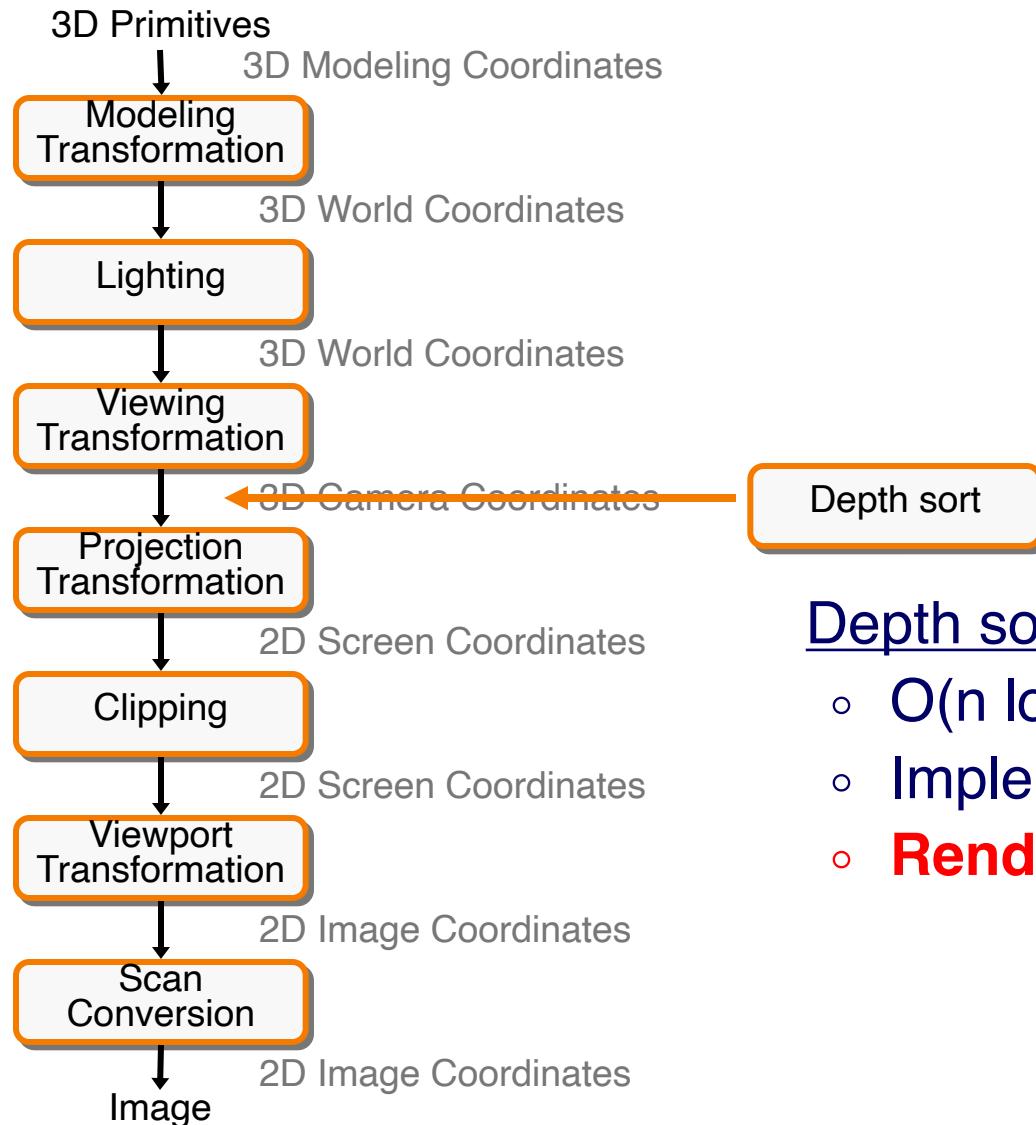
Depth sort

- “Painter’s algorithm”
 - First sort surfaces in order of decreasing **maximum depth**
 - Scan convert surfaces in **back-to-front** order, **always overwriting pixels**





3D Rasterization Pipeline



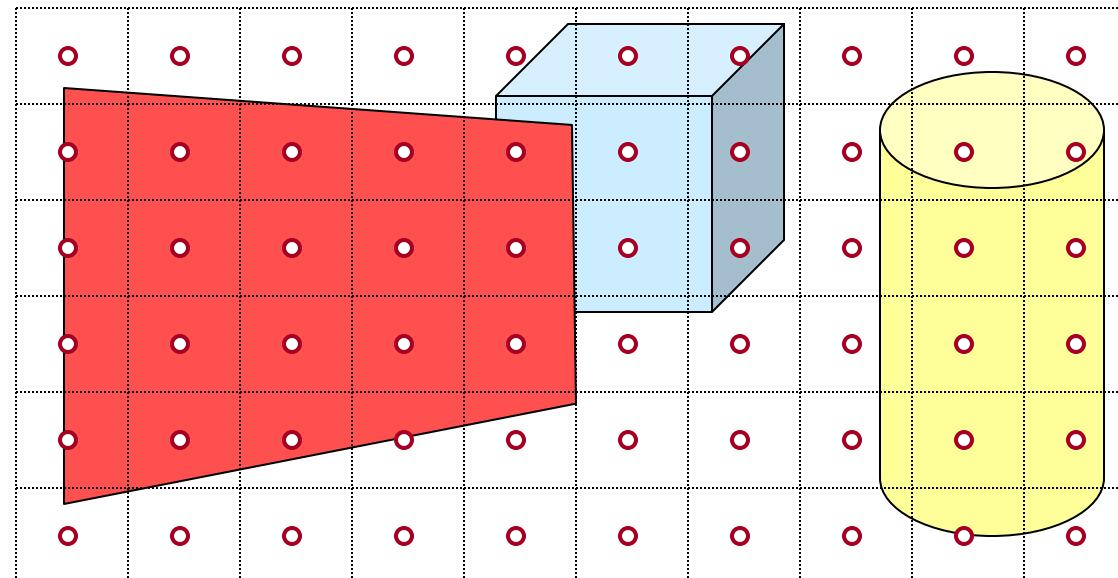
Depth sort comments

- $O(n \log n)$
- Implemented in software
- **Render pixels of every polygon**



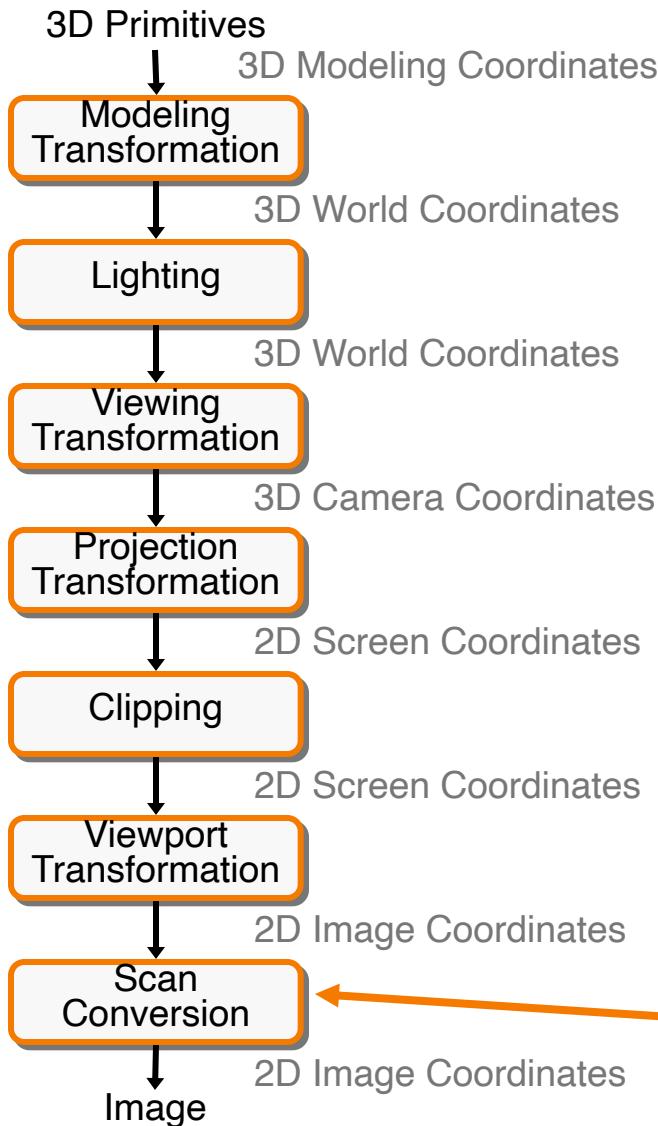
Z-Buffer

- Maintain color & depth of closest object per pixel
 - Framebuffer now RGBAz – initialize z to far plane
 - Update only pixels with depth **closer** than currently in z-buffer
 - Depths are interpolated for in-primitive pixels from vertices, just like colors



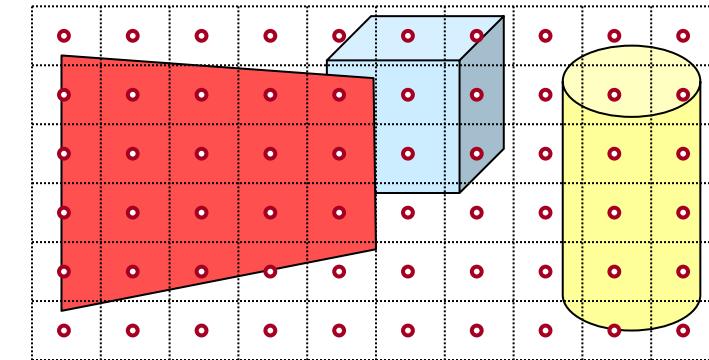


Z-Buffer



Z-buffer comments

- + Polygons rasterized in any order
- + Process one polygon at a time
- + Suitable for hardware pipeline
- Requires extra memory for z-buffer
 - o Commonly in hardware





Hidden Surface Removal Algorithms

36 • I. E. Sutherland, R. F. Sproull, and R. A. Schumacker

A Characterization of Ten Hidden-Surface Algorithms

37

Only z-buffer and ray tracing

OPAQUE-OBJECT ALGORITHMS																																																																																																																								
COMPARISON ALGORITHMS																																																																																																																								
OBJECT SPACE				(partly each)				IMAGE SPACE																																																																																																																
edges edges				edges volumes				DEPTH PRIORITY ALGORITHMS																																																																																																																
edges edges				a priori priority				area sampling																																																																																																																
edges edges				dynamically, computed priority				point sampling																																																																																																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">RESTRICTIONS</th><th>APPEL 1967</th><th>GALIMBERTI, et al 1969</th><th>LOUTREL 1967</th><th>ROBERTS 1963</th><th>SCHUMACKER, et al 1969</th><th>NEWELL, et al 1972</th><th>WARNOCK 1968</th><th>MATEIN 1970</th><th>RONNEY, et al 1967</th><th>BOURNIGHT 1969</th></tr> </thead> <tbody> <tr> <td>COHERENCE</td><td>Promote visibility to all edges at vertex</td><td>Promote visibility to all edges at vertex</td><td>Promote visibility to all edges at vertex</td><td></td><td>Frame coherence in depth</td><td>None used</td><td>None used</td><td>None</td><td>TP, CF, NP</td><td></td></tr> <tr> <td>SORTING</td><td>Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t</td><td>Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t</td><td>Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t</td><td>Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t</td><td>Intra-Cluster Priority (1) Faces - max of max points (2) Dot product with visibility (3) Cull (4) List of edges, E_s (5) l, E_t</td><td>Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Ordered table (5) l, F_r</td><td>Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Ordered table (5) l, F_r</td><td>Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Table of lists (5) l, E_r</td><td>Sort (Opt.) (1) Faces, min Y (2) Comparison (3) Bucket (4) Table of lists (5) l, E_r</td><td>Y Sort (1) Edges, Min Y (2) Comparison (3) Bucket (4) Table of lists (5) l, E_r</td></tr> <tr> <td>Method</td><td>Contour Edge Cull (1) Edges separating front & back faces (2) Dot product with normals & topology (3) Cull (4) List, E_c (5) l, E_t</td><td>Contour Edge Cull (1) Edges separating front & back faces (2) Dot product with normals & topology (3) Cull (4) List, E_c (5) l, E_t</td><td>(Omitted)</td><td>(Omitted)</td><td>Inter-Cluster Priority (1) Clusters (2) Dot product with separating planes (3) Prefix scan (4) Quadtree (5) l, E_t</td><td>X Merge (1) Clusters (2) Dot product with separating planes (3) Prefix scan (4) Quadtree (5) l, E_t</td><td>X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) E_r, S_k</td><td>X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) n, S_k</td><td>X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) $E_r, 2S_k$ (edges)</td><td>X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) $E_r, 2S_k$ (edges)</td></tr> <tr> <td>Type</td><td>(3)</td><td>(3)</td><td>(3)</td><td>(3)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>Result structure</td><td>(4)</td><td>(4)</td><td>(4)</td><td>(4)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>Number per frame, number of objects</td><td>(5)</td><td>(5)</td><td>(5)</td><td>(5)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>(merge)</td><td>Initial Visibility (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r</td><td>Initial Visibility (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r</td><td>Visibility (1) Ray to vertex against all faces (2) Betweenness (3) Edges (4) Exhaustive search (5) Quantitative visibility of vertex (6) Objects, F_r</td><td>Edge/Volume Test (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r</td><td>Cull (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r</td><td>Y Sort (1) Face segment by Y range (2) Dot product with normal (3) Cull (4) Smaller ordered table (5) l, F_r</td><td>Depth Search (1) Segments, x left (2) 4-corner compare (3) Bucket (4) None (5) l, F_r</td><td>Depth Search (1) Segments, x left (2) 4-corner compare (3) Bucket (4) Answer/failure (5) l, F_r</td><td>Priority Search (1) Edges, value (2) Comparison (3) Bubble (4) Active segment list (5) n, m</td><td>Y Sort (1) Edges, X value (2) Comparison (3) Bucket (4) 1-way linked list (5) $N, 2S_k$ (edges)</td></tr> <tr> <td>User interface</td><td>Intersection search with mask</td><td>Intersection search with mask</td><td>Intersection search with mask</td><td>Intersection search with mask</td><td>Y Cull (1) Faces by Y extent (2) Mini-max on Y axis (3) Cull (unordered) (4) X intercepts of relevant segments (5) E_s, E_c</td><td>X Merge (1) Segments, Y intercept (2) Comparison (3) Cull ordered list (4) X intercepts of relevant segments (5) S_r, S_v</td><td>IV Sort (Opt.) (1) Segments, overlap with sample span (2) Double comparison and comparison (3) Cull ordered list (4) Ordered merge (5) S_r, S_v</td><td>IV Sort (Opt.) (1) Segments, overlap with sample span (2) Double comparison and comparison (3) Cull ordered list (4) Ordered merge (5) S_r, S_v</td><td>Priority Search (1) Segments, priority (2) Logic network (3) Logic network (4) Visible segment (5) nm, S_k</td><td>Y Sort (1) Edges, X value (2) Comparison (3) Bucket (4) 1-way linked list (5) $N, 2S_k$ (edges)</td></tr> <tr> <td>Search with mask</td><td>Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)</td><td>Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X_r/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)</td><td>Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X_r/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)</td><td>Y Sort (1) Segments, Z (2) Counters (3) Hardware (4) Segments at this X (5) nm, S_k</td><td>Z Search (1) Segments, Z (2) Depth by logarithmic search (3) Search (unordered) (4) Visible segment (5) n^2S_k, B_c</td><td>Span Cull (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search (unordered) (4) Visible segment (5) n^2S_k, B_c</td><td>Span Cull (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search (unordered) (4) Visible segment (5) n^2S_k, B_c</td><td>Z Search (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search of unsorted active list (4) Visible segment (5) n^2S_k, B_c</td><td>Z Search (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search of unsorted active list (4) Visible segment (5) n^2S_k, B_c</td></tr> <tr> <td>Priority Search</td><td>(7)</td><td>(7)</td><td>(7)</td><td>(7)</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>	RESTRICTIONS	APPEL 1967	GALIMBERTI, et al 1969	LOUTREL 1967	ROBERTS 1963	SCHUMACKER, et al 1969	NEWELL, et al 1972	WARNOCK 1968	MATEIN 1970	RONNEY, et al 1967	BOURNIGHT 1969	COHERENCE	Promote visibility to all edges at vertex	Promote visibility to all edges at vertex	Promote visibility to all edges at vertex		Frame coherence in depth	None used	None used	None	TP, CF, NP		SORTING	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Intra-Cluster Priority (1) Faces - max of max points (2) Dot product with visibility (3) Cull (4) List of edges, E_s (5) l, E_t	Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Ordered table (5) l, F_r	Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Ordered table (5) l, F_r	Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Table of lists (5) l, E_r	Sort (Opt.) (1) Faces, min Y (2) Comparison (3) Bucket (4) Table of lists (5) l, E_r	Y Sort (1) Edges, Min Y (2) Comparison (3) Bucket (4) Table of lists (5) l, E_r	Method	Contour Edge Cull (1) Edges separating front & back faces (2) Dot product with normals & topology (3) Cull (4) List, E_c (5) l, E_t	Contour Edge Cull (1) Edges separating front & back faces (2) Dot product with normals & topology (3) Cull (4) List, E_c (5) l, E_t	(Omitted)	(Omitted)	Inter-Cluster Priority (1) Clusters (2) Dot product with separating planes (3) Prefix scan (4) Quadtree (5) l, E_t	X Merge (1) Clusters (2) Dot product with separating planes (3) Prefix scan (4) Quadtree (5) l, E_t	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) E_r, S_k	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) n, S_k	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) $E_r, 2S_k$ (edges)	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) $E_r, 2S_k$ (edges)	Type	(3)	(3)	(3)	(3)							Result structure	(4)	(4)	(4)	(4)							Number per frame, number of objects	(5)	(5)	(5)	(5)							(merge)	Initial Visibility (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Initial Visibility (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Visibility (1) Ray to vertex against all faces (2) Betweenness (3) Edges (4) Exhaustive search (5) Quantitative visibility of vertex (6) Objects, F_r	Edge/Volume Test (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Cull (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Y Sort (1) Face segment by Y range (2) Dot product with normal (3) Cull (4) Smaller ordered table (5) l, F_r	Depth Search (1) Segments, x left (2) 4-corner compare (3) Bucket (4) None (5) l, F_r	Depth Search (1) Segments, x left (2) 4-corner compare (3) Bucket (4) Answer/failure (5) l, F_r	Priority Search (1) Edges, value (2) Comparison (3) Bubble (4) Active segment list (5) n, m	Y Sort (1) Edges, X value (2) Comparison (3) Bucket (4) 1-way linked list (5) $N, 2S_k$ (edges)	User interface	Intersection search with mask	Y Cull (1) Faces by Y extent (2) Mini-max on Y axis (3) Cull (unordered) (4) X intercepts of relevant segments (5) E_s, E_c	X Merge (1) Segments, Y intercept (2) Comparison (3) Cull ordered list (4) X intercepts of relevant segments (5) S_r, S_v	IV Sort (Opt.) (1) Segments, overlap with sample span (2) Double comparison and comparison (3) Cull ordered list (4) Ordered merge (5) S_r, S_v	IV Sort (Opt.) (1) Segments, overlap with sample span (2) Double comparison and comparison (3) Cull ordered list (4) Ordered merge (5) S_r, S_v	Priority Search (1) Segments, priority (2) Logic network (3) Logic network (4) Visible segment (5) nm, S_k	Y Sort (1) Edges, X value (2) Comparison (3) Bucket (4) 1-way linked list (5) $N, 2S_k$ (edges)	Search with mask	Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)	Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X_r/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)	Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X_r/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)	Y Sort (1) Segments, Z (2) Counters (3) Hardware (4) Segments at this X (5) nm, S_k	Z Search (1) Segments, Z (2) Depth by logarithmic search (3) Search (unordered) (4) Visible segment (5) n^2S_k , B_c	Span Cull (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search (unordered) (4) Visible segment (5) n^2S_k , B_c	Span Cull (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search (unordered) (4) Visible segment (5) n^2S_k , B_c	Z Search (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search of unsorted active list (4) Visible segment (5) n^2S_k , B_c	Z Search (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search of unsorted active list (4) Visible segment (5) n^2S_k , B_c	Priority Search	(7)	(7)	(7)	(7)									
RESTRICTIONS	APPEL 1967	GALIMBERTI, et al 1969	LOUTREL 1967	ROBERTS 1963	SCHUMACKER, et al 1969	NEWELL, et al 1972	WARNOCK 1968	MATEIN 1970	RONNEY, et al 1967	BOURNIGHT 1969																																																																																																														
COHERENCE	Promote visibility to all edges at vertex	Promote visibility to all edges at vertex	Promote visibility to all edges at vertex		Frame coherence in depth	None used	None used	None	TP, CF, NP																																																																																																															
SORTING	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Back Edge Cull (1) Edges separating back-facing planes (2) Dot product with normals & topology (3) Cull (4) List of edges, E_s (5) l, E_t	Intra-Cluster Priority (1) Faces - max of max points (2) Dot product with visibility (3) Cull (4) List of edges, E_s (5) l, E_t	Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Ordered table (5) l, F_r	Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Ordered table (5) l, F_r	Sort (Opt.) (1) Faces, max Z (2) Dot product of max points (3) Log (4) Table of lists (5) l, E_r	Sort (Opt.) (1) Faces, min Y (2) Comparison (3) Bucket (4) Table of lists (5) l, E_r	Y Sort (1) Edges, Min Y (2) Comparison (3) Bucket (4) Table of lists (5) l, E_r																																																																																																														
Method	Contour Edge Cull (1) Edges separating front & back faces (2) Dot product with normals & topology (3) Cull (4) List, E_c (5) l, E_t	Contour Edge Cull (1) Edges separating front & back faces (2) Dot product with normals & topology (3) Cull (4) List, E_c (5) l, E_t	(Omitted)	(Omitted)	Inter-Cluster Priority (1) Clusters (2) Dot product with separating planes (3) Prefix scan (4) Quadtree (5) l, E_t	X Merge (1) Clusters (2) Dot product with separating planes (3) Prefix scan (4) Quadtree (5) l, E_t	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) E_r, S_k	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) n, S_k	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) $E_r, 2S_k$ (edges)	X Merge (1) Edges, X value (2) Comparison (3) Bucket (4) 2-way linked list (5) $E_r, 2S_k$ (edges)																																																																																																														
Type	(3)	(3)	(3)	(3)																																																																																																																				
Result structure	(4)	(4)	(4)	(4)																																																																																																																				
Number per frame, number of objects	(5)	(5)	(5)	(5)																																																																																																																				
(merge)	Initial Visibility (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Initial Visibility (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Visibility (1) Ray to vertex against all faces (2) Betweenness (3) Edges (4) Exhaustive search (5) Quantitative visibility of vertex (6) Objects, F_r	Edge/Volume Test (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Cull (1) Ray to vertex against all faces (2) Depth (3) Current (4) Quantitative (5) Objects, F_r	Y Sort (1) Face segment by Y range (2) Dot product with normal (3) Cull (4) Smaller ordered table (5) l, F_r	Depth Search (1) Segments, x left (2) 4-corner compare (3) Bucket (4) None (5) l, F_r	Depth Search (1) Segments, x left (2) 4-corner compare (3) Bucket (4) Answer/failure (5) l, F_r	Priority Search (1) Edges, value (2) Comparison (3) Bubble (4) Active segment list (5) n, m	Y Sort (1) Edges, X value (2) Comparison (3) Bucket (4) 1-way linked list (5) $N, 2S_k$ (edges)																																																																																																														
User interface	Intersection search with mask	Intersection search with mask	Intersection search with mask	Intersection search with mask	Y Cull (1) Faces by Y extent (2) Mini-max on Y axis (3) Cull (unordered) (4) X intercepts of relevant segments (5) E_s, E_c	X Merge (1) Segments, Y intercept (2) Comparison (3) Cull ordered list (4) X intercepts of relevant segments (5) S_r, S_v	IV Sort (Opt.) (1) Segments, overlap with sample span (2) Double comparison and comparison (3) Cull ordered list (4) Ordered merge (5) S_r, S_v	IV Sort (Opt.) (1) Segments, overlap with sample span (2) Double comparison and comparison (3) Cull ordered list (4) Ordered merge (5) S_r, S_v	Priority Search (1) Segments, priority (2) Logic network (3) Logic network (4) Visible segment (5) nm, S_k	Y Sort (1) Edges, X value (2) Comparison (3) Bucket (4) 1-way linked list (5) $N, 2S_k$ (edges)																																																																																																														
Search with mask	Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)	Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X_r/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)	Sort Along Edge (1) Intersections on edge, ordering (2) Comparison (3) Bubble (4) Answer (5) $E_s, X_r/E_s$ (6) $E_s, X_r/E_s$ (7) Omit if well hidden)	Y Sort (1) Segments, Z (2) Counters (3) Hardware (4) Segments at this X (5) nm, S_k	Z Search (1) Segments, Z (2) Depth by logarithmic search (3) Search (unordered) (4) Visible segment (5) n^2S_k , B_c	Span Cull (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search (unordered) (4) Visible segment (5) n^2S_k , B_c	Span Cull (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search (unordered) (4) Visible segment (5) n^2S_k , B_c	Z Search (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search of unsorted active list (4) Visible segment (5) n^2S_k , B_c	Z Search (1) Segments, depth with linear equations and comparison (2) Double comparison and comparison (3) Search of unsorted active list (4) Visible segment (5) n^2S_k , B_c																																																																																																															
Priority Search	(7)	(7)	(7)	(7)																																																																																																																				

Figure 29. Characterization of ten opaque-object algorithms & Comparison of the algorithms.

[Sutherland '74]



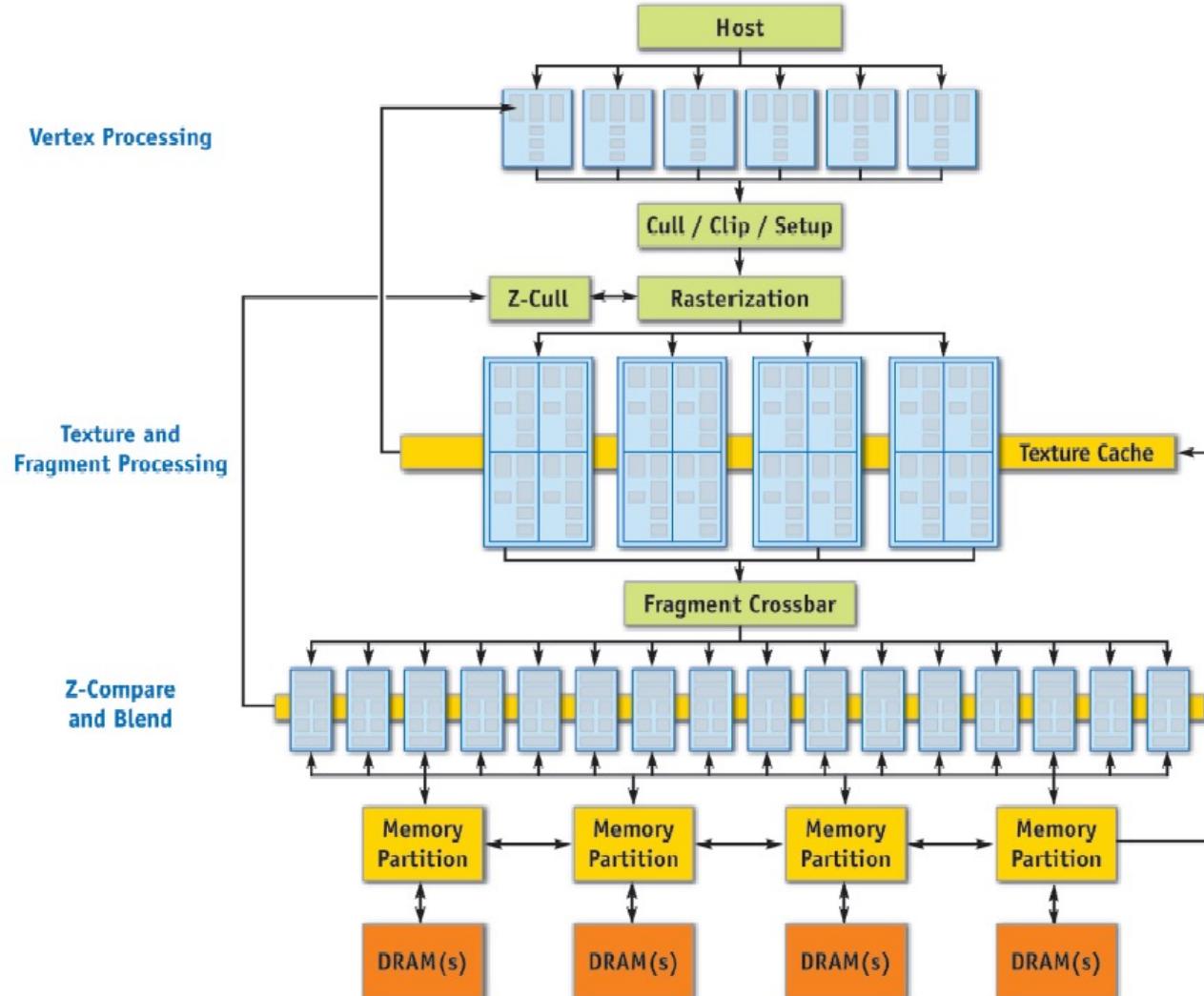
Rasterization Summary

- Scan conversion
 - Sweep-line algorithm
- Shading algorithms
 - Flat, Gouraud, Phong
- Texture mapping
 - Mipmaps
- Visibility determination
 - Z-buffer

This is all in hardware



GPU Architecture



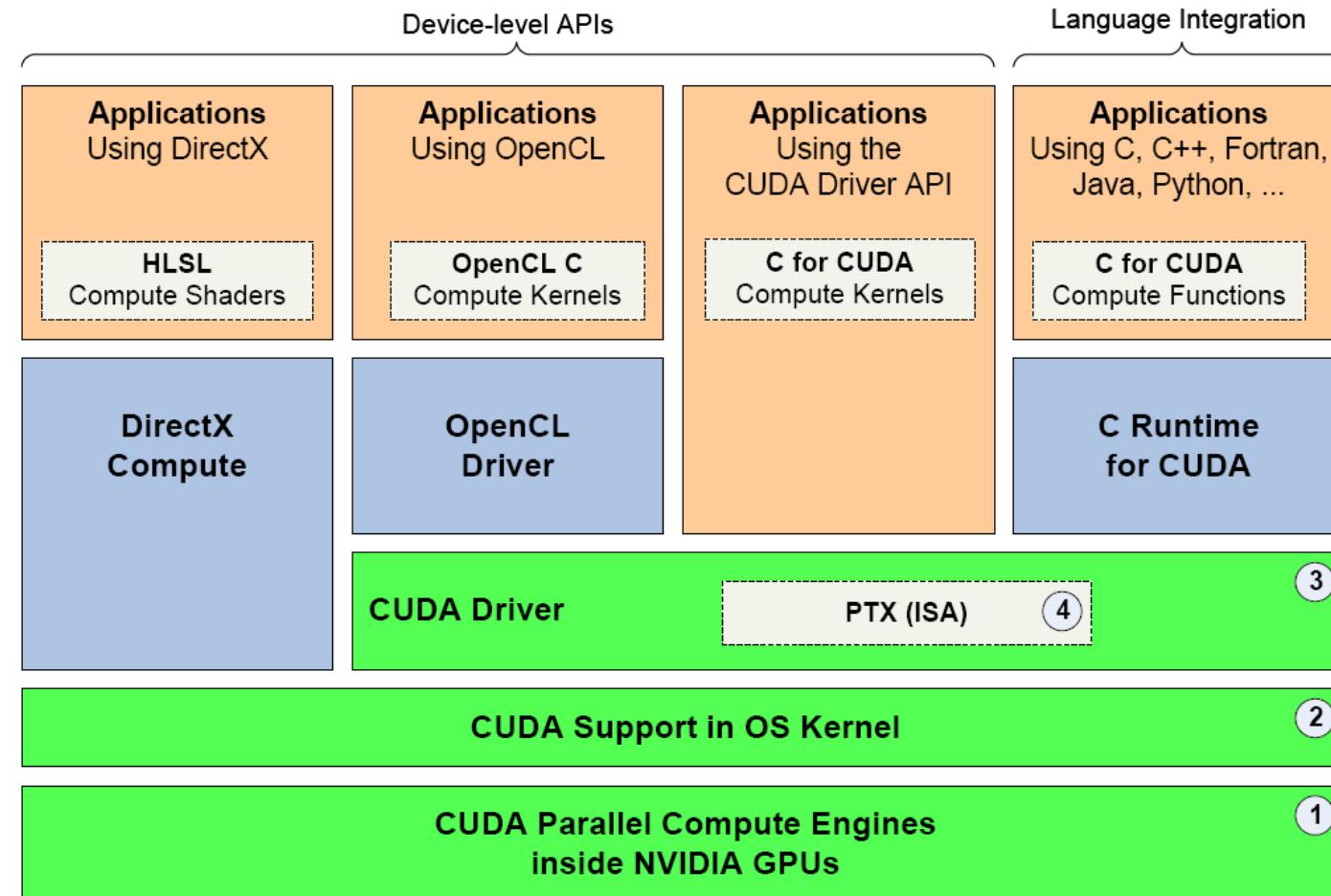
GeForce 6 Series Architecture

GPU Gems 2, NVIDIA



Actually ...

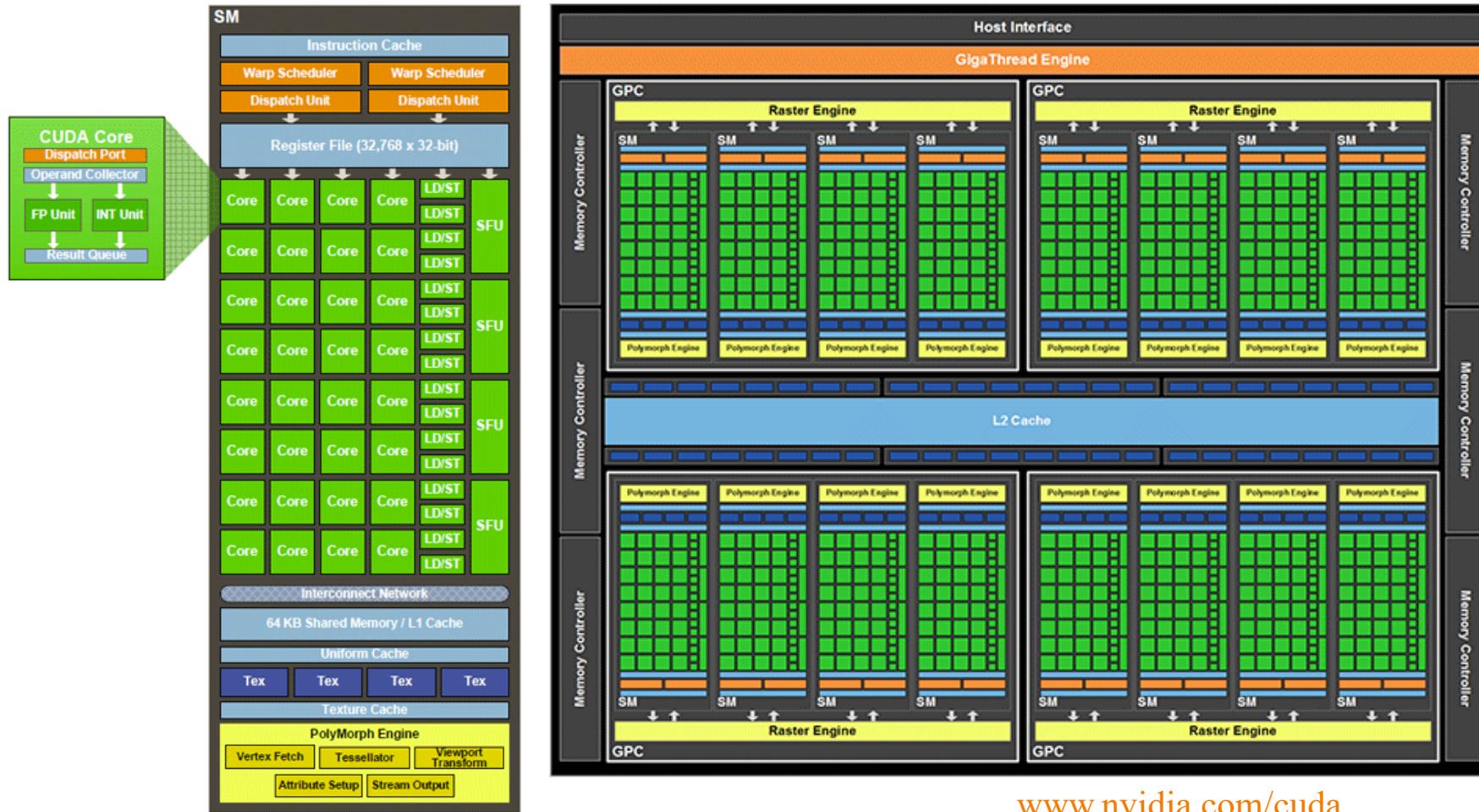
- Modern graphics hardware is programmable





Trend ...

- GPU is general-purpose parallel computer



www.nvidia.com/cuda