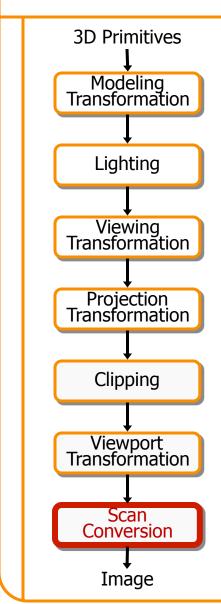


Rasterization

COS 426, Spring 2016
Princeton University

3D Rendering Pipeline (for direct illumination)





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

Rasterization

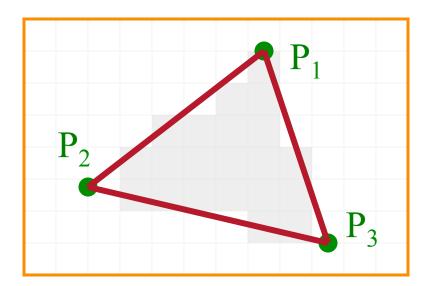


- Scan conversion (last time)
 - Determine which pixels to fill
- Shading
 - Determine a color for each filled pixel
- Texture mapping
 - Describe shading variation within polygon interiors
- Visible surface determination
 - Figure out which surface is front-most at every pixel

Shading



How do we choose a color for each filled pixel?

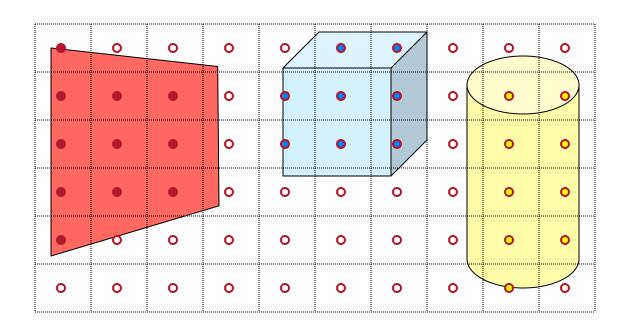


Emphasis on methods that can be implemented in hardware

Ray Casting



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

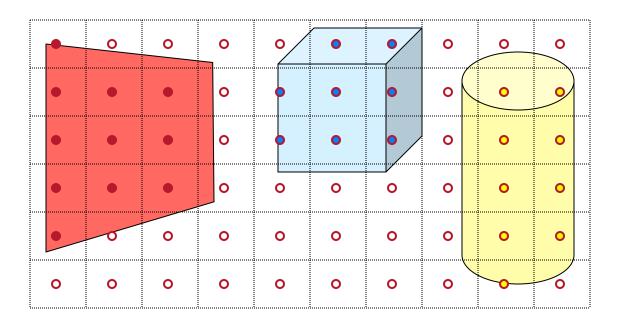


$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$

Polygon Shading



- Can take advantage of spatial coherence
 - Illumination calculations for pixels covered by same primitive are related to each other



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

Polygon Shading Algorithms

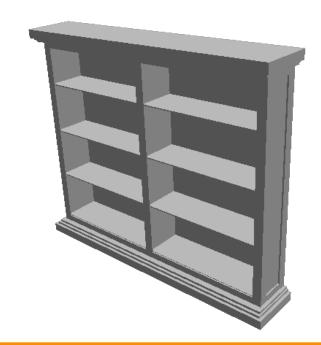


- Flat Shading
- Gouraud Shading
- Phong Shading

Flat Shading



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

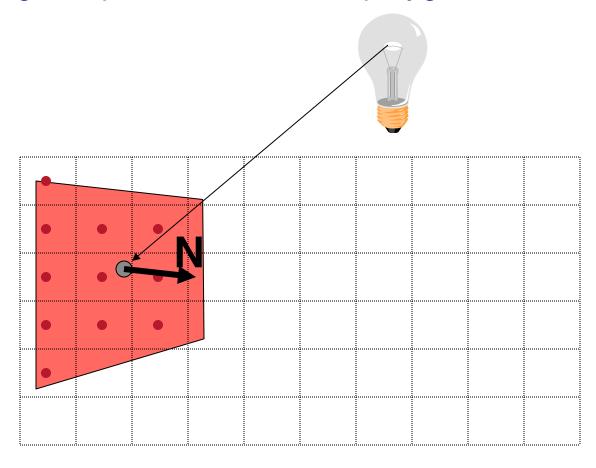


$$I = I_E + K_A I_{AL} + \sum_{i} \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$

Flat Shading



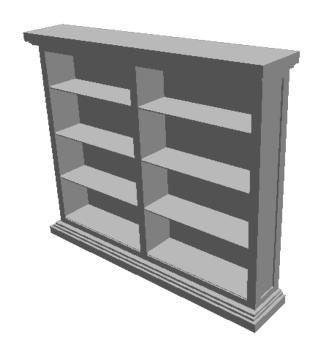
- One illumination calculation per polygon
 - Assign all pixels inside each polygon the same color

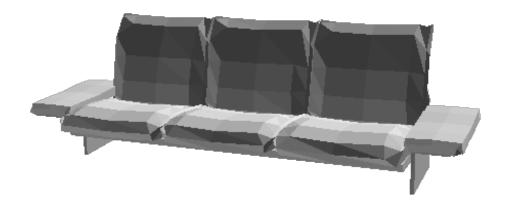


Flat Shading



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





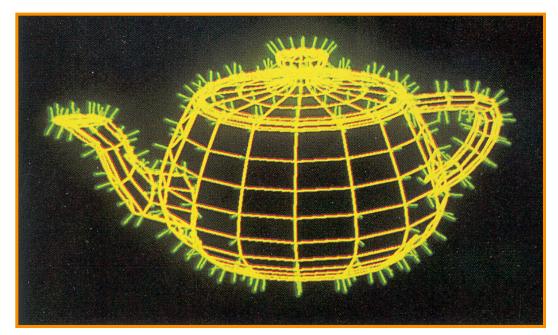
Polygon Shading Algorithms



- Flat Shading
- Gouraud Shading
- Phong Shading



 What if smooth surface is represented by polygonal mesh with a normal at each vertex?

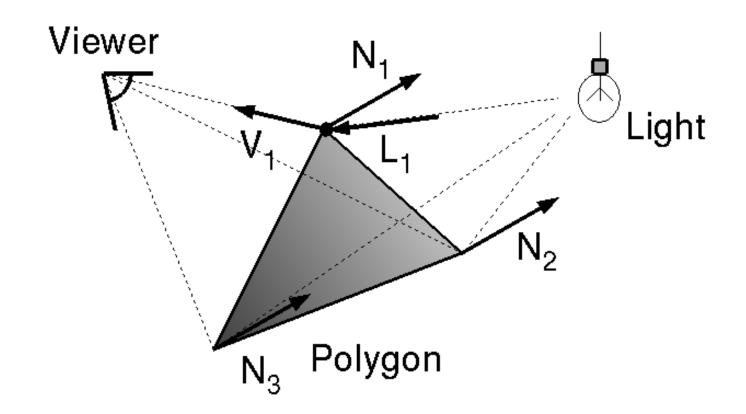


Watt Plate 7

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



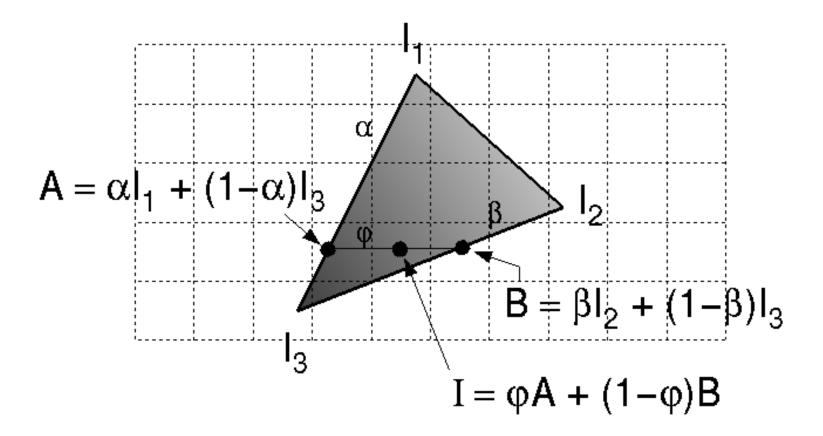
- Method 1: One lighting calculation per vertex
 - Assign pixels inside polygon by interpolating colors computed at vertices





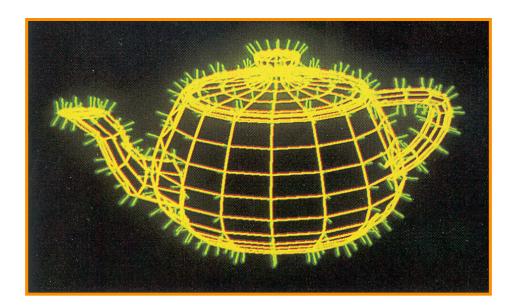
Bilinear interpolation of colors at vertices

down and across scan lines = barycentric coords





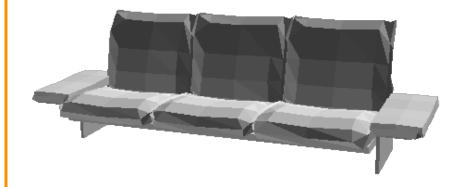
- Smooth shading over adjacent polygons
 - Curved surfaces
 - Illumination highlights
 - Soft shadows

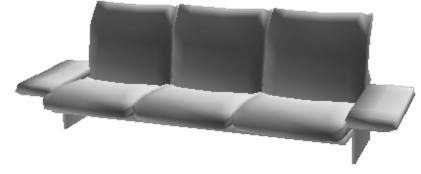


Mesh with shared normals at vertices



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





Flat Shading

Gouraud Shading

Polygon Shading Algorithms

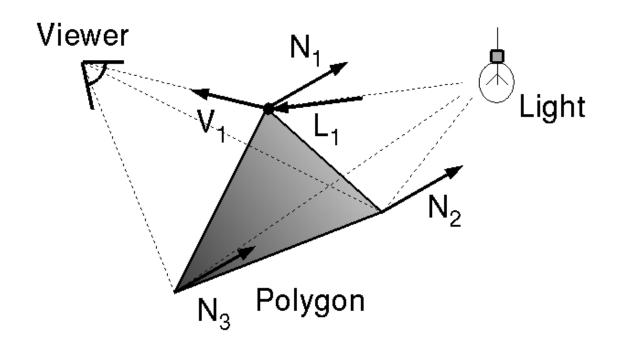


- Flat Shading
- Gouraud Shading
- Phong Shading (≠ Phong reflectance model)

Phong Shading



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

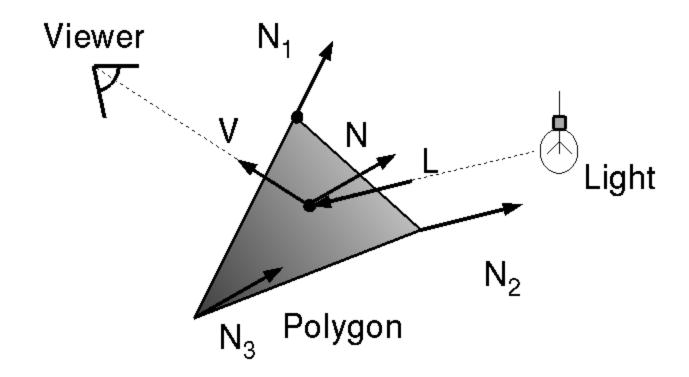


$$I = I_E + K_A I_{AL} + \sum_i \left(K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i \right)$$

Phong Shading



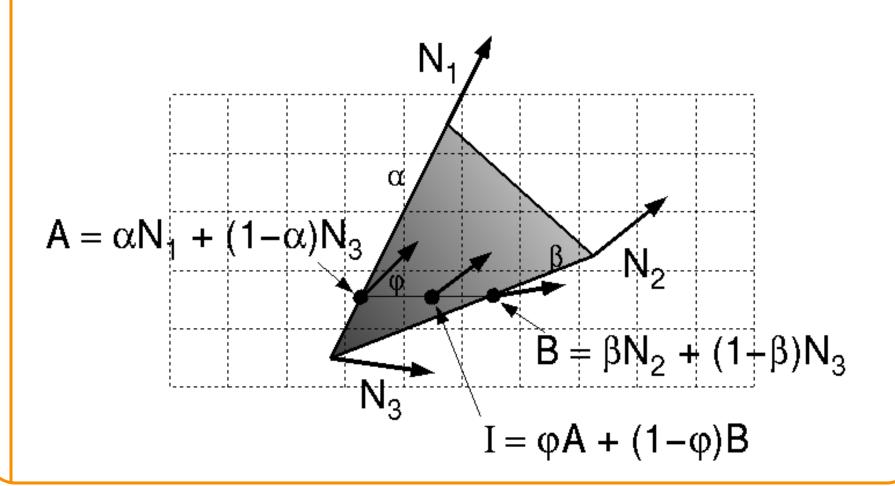
- One lighting calculation per pixel
 - Approximate surface normals for points inside polygons by bilinear interpolation of normals from vertices



Phong Shading



Bilinear interpolation of surface normals at vertices

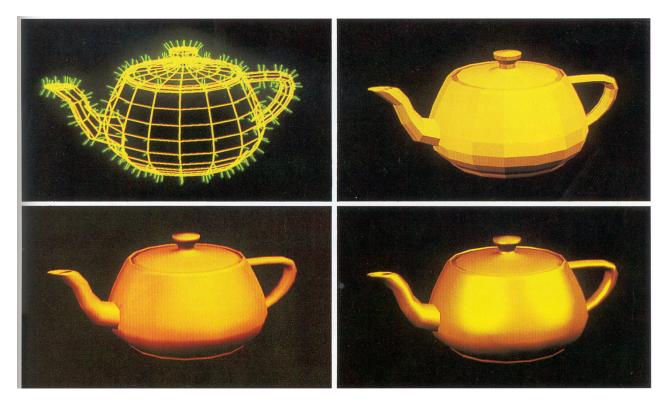


Polygon Shading Algorithms



Wireframe

Flat



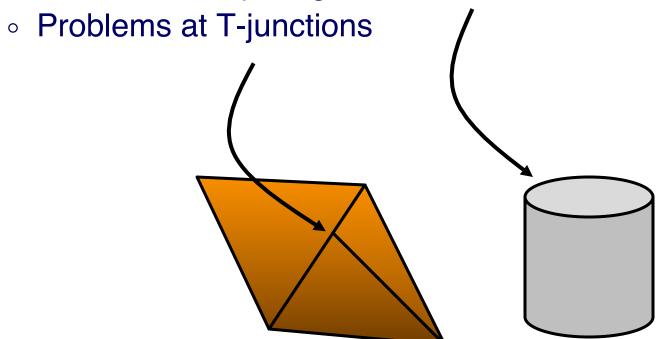
Gouraud

Phong

Shading Issues



- Problems with interpolated shading:
 - Polygonal silhouettes still obvious
 - Perspective distortion (due to screen-space interpolation)
 - Problems computing shared vertex normals



Rasterization

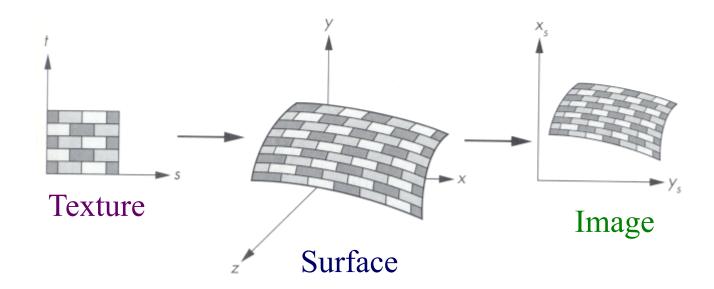


- Scan conversion
 - Determine which pixels to fill
- Shading
 - Determine a color for each filled pixel
- Texture mapping
 - Describe shading variation within polygon interiors
- Visible surface determination
 - Figure out which surface is front-most at every pixel

Textures



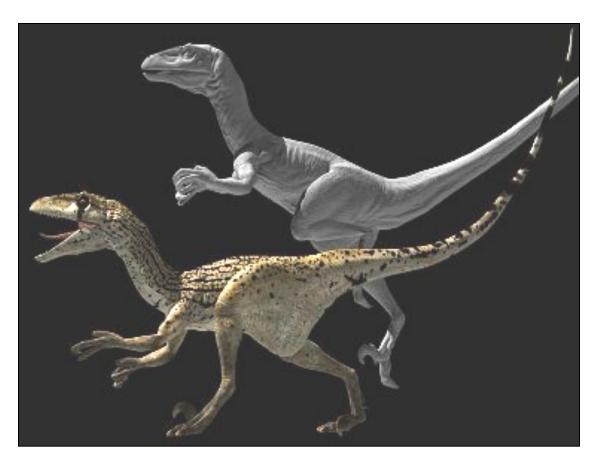
- Describe color variation in interior of 3D polygon
 - When scan converting a polygon, vary pixel colors according to values fetched from a texture image



Textures



Add visual detail to surfaces of 3D objects





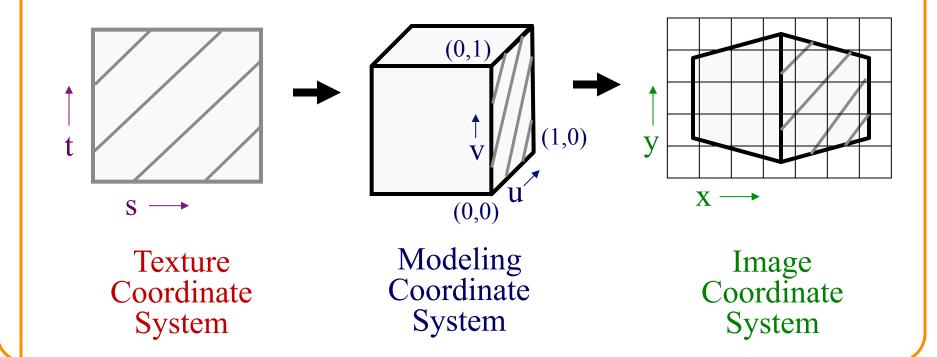
[Daren Horley]

Texture Mapping



Steps:

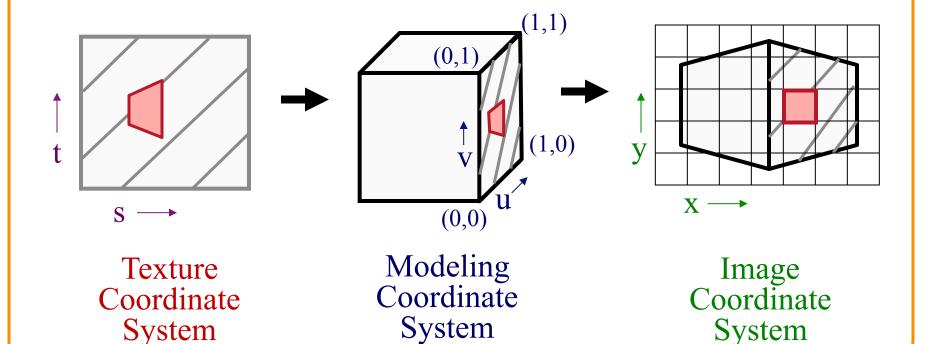
- Define texture
- Specify mapping from texture to surface
- Look up texture values during scan conversion



Texture Mapping



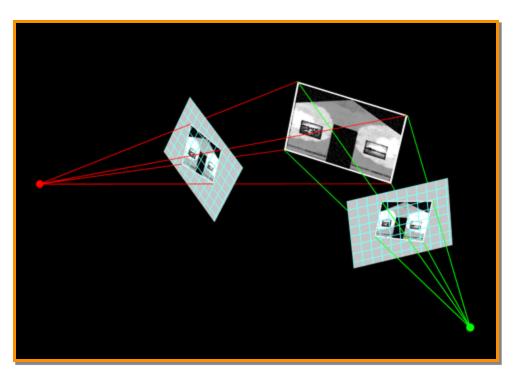
- When scan converting, map from ...
 - image coordinate system (x,y) to
 - modeling coordinate system (u,v) to
 - texture image (s,t)



Texture Mapping



- Texture mapping is a 2D projective transformation
 - texture coordinate system: (s,t) to
 - image coordinate system (x,y)



[Allison Klein]

Texture Overview



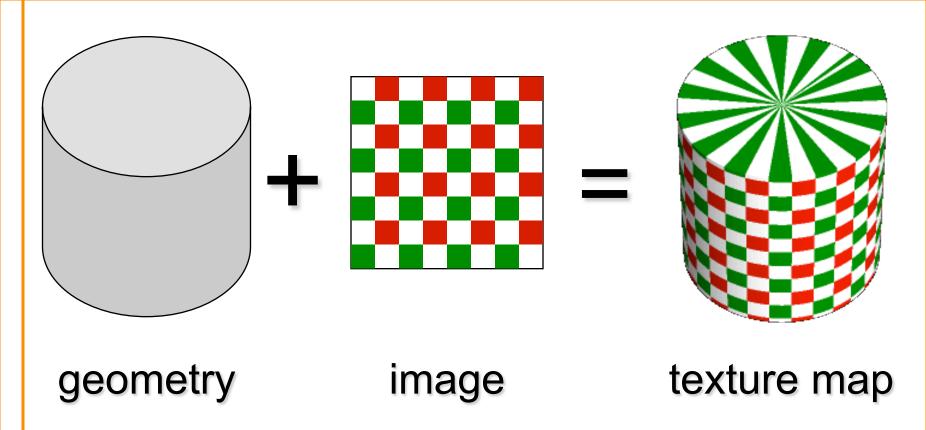
- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering

Texture Overview



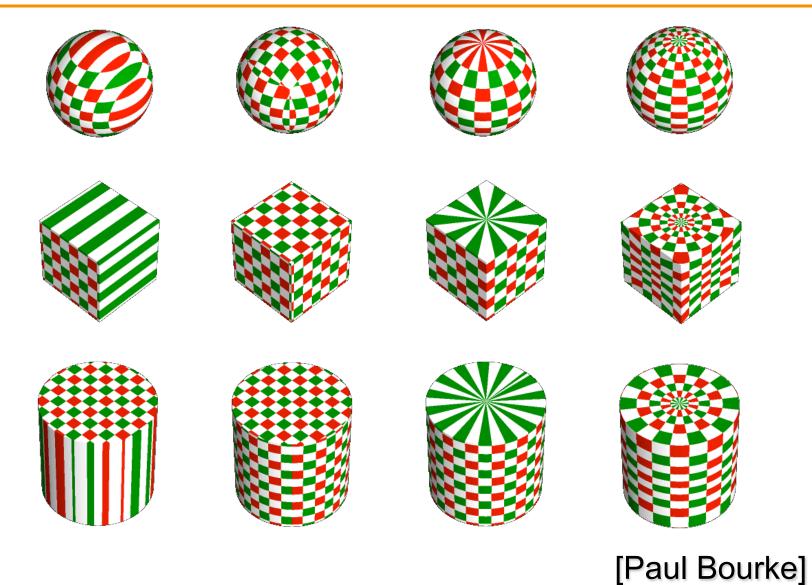
- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering





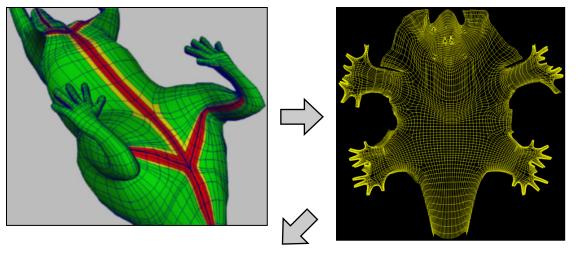
 Q: How do we decide where on the geometry each color from the image should go?



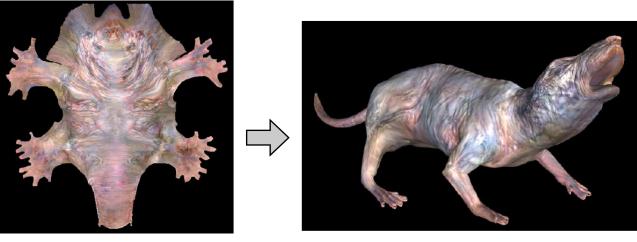




Option1: unfold the surface

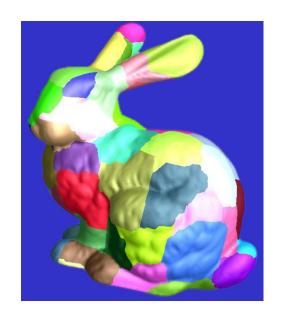


[Piponi2000]

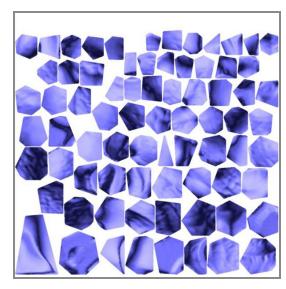




Option2: make an atlas



charts



atlas



surface

[Sander2001]

Texture Overview

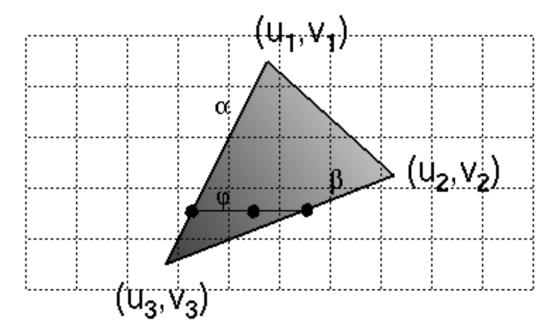


- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering
 - Non-photorealistic rendering

Texture Mapping

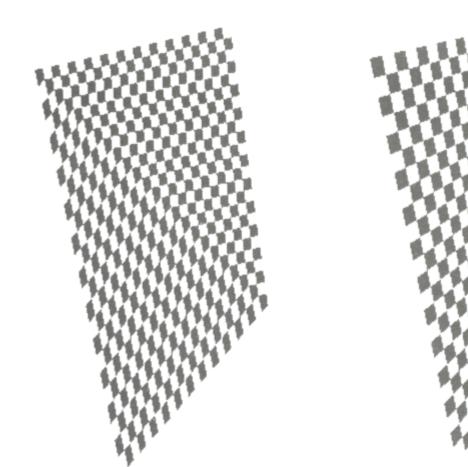


- Scan conversion
 - Interpolate texture coordinates down/across scan lines
 - Distortion due to bilinear interpolation approximation
 - » Cut polygons into smaller ones, or
 - » Perspective divide at each pixel



Texture Mapping





Linear interpolation of texture coordinates

Correct interpolation with perspective divide

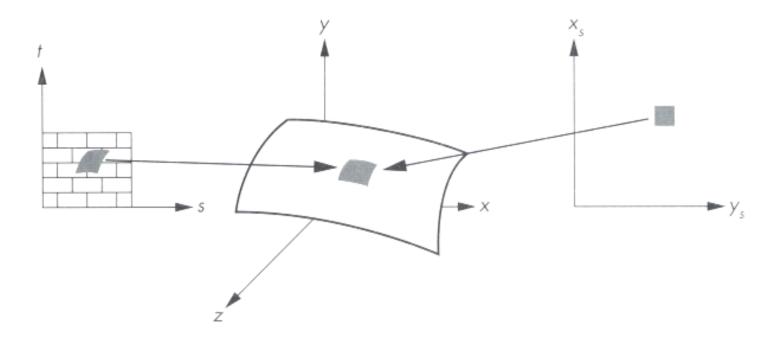
Texture Overview



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

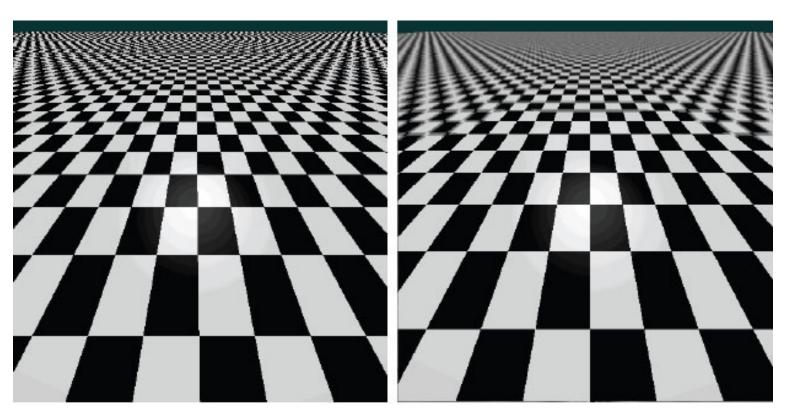


 Must sample texture to determine color at each pixel in image





Aliasing is a problem

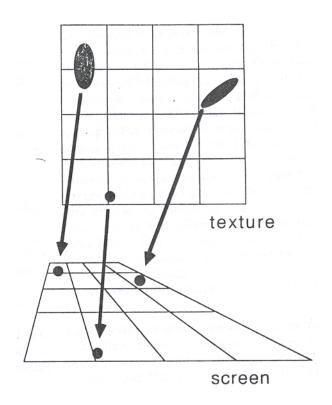


Point sampling

Area filtering



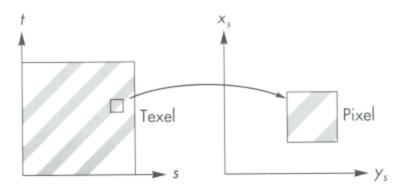
Ideally, use elliptically shaped convolution filters



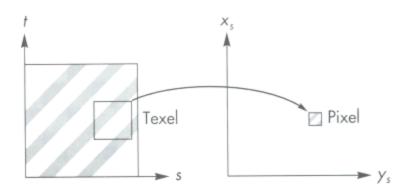
In practice, use rectangles or squares



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



Magnification



Minification

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













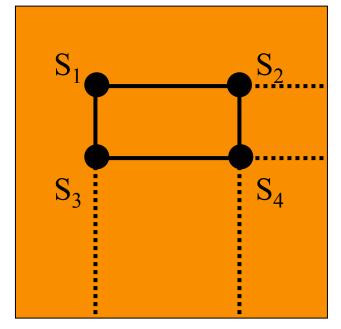


Summed-area tables



- At each texel keep sum of all values down & right
 - \circ 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

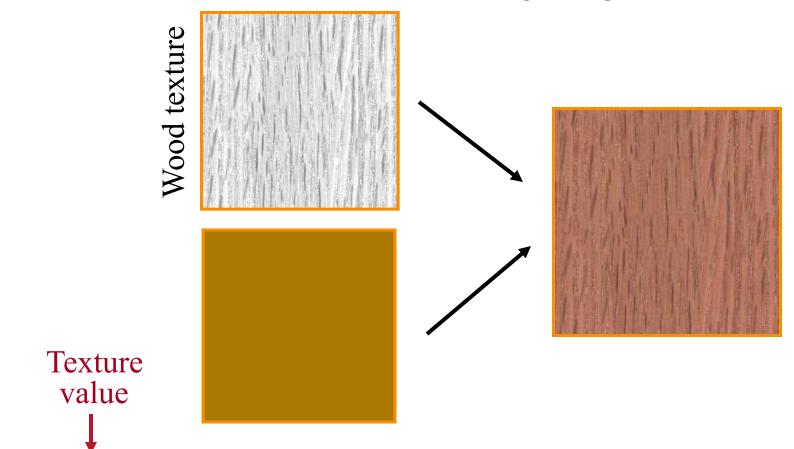


- Texture mapping stages
 - Parameterization
 - Mapping
 - Filtering
- Texture mapping applications
 - Modulation textures
 - Illumination mapping
 - Bump mapping
 - Environment mapping
 - Image-based rendering

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
- \circ K_T
- o n



Texture value

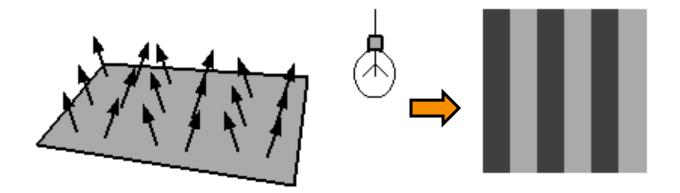
$$I = I_E + K_A I_A + \sum_{L} (K_D(s, t)(N \cdot L) + K_S(V \cdot R)^n) S_L I_L + K_T I_T + K_S I_S$$

Bump/Normal Mapping



Texture values perturb surface normals:

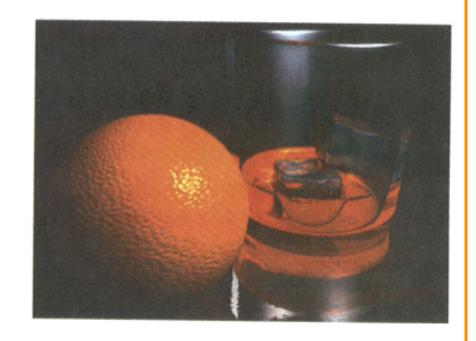
- Use gradient of grayscale image ("bump")
- Encode normals (or offsets) in RGB
- Encode normal offsets in tangent space



Bump Mapping

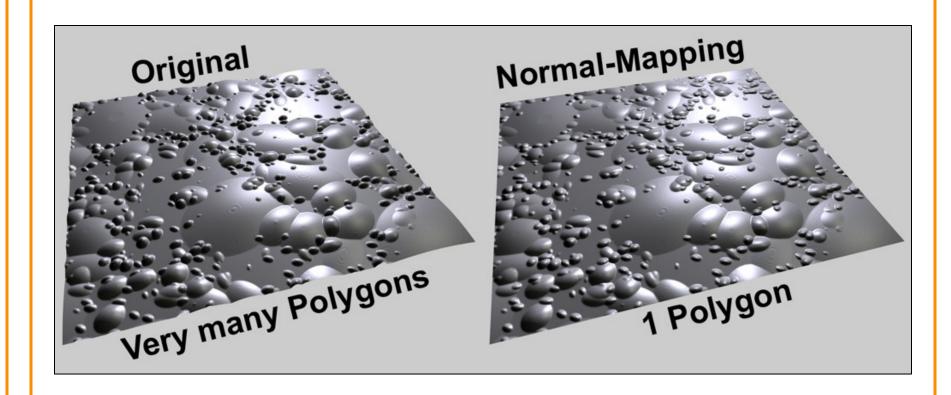






Normal Mapping



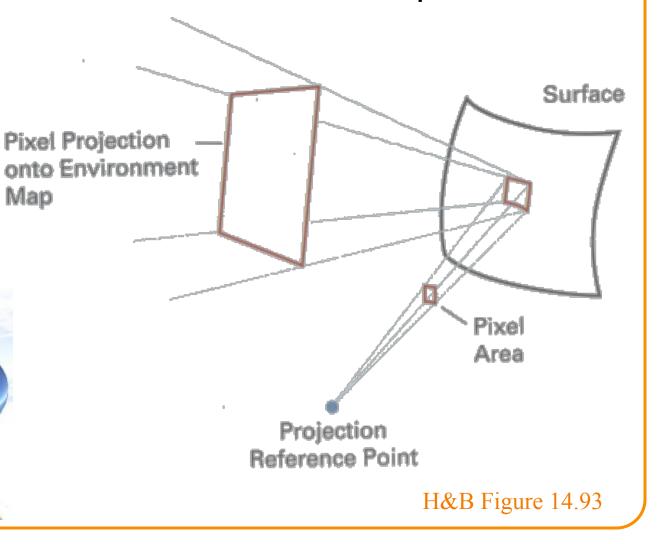


Environment Mapping

Map



Texture values are reflected off surface patch



Gamer3D/Wikipedia

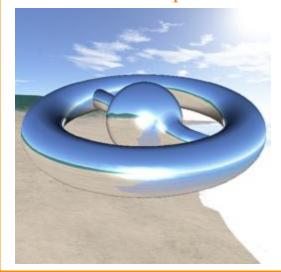


Image-Based Rendering



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
- Compute on the fly,
 e.g. Perlin noise →



Texture Summary



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

Rasterization

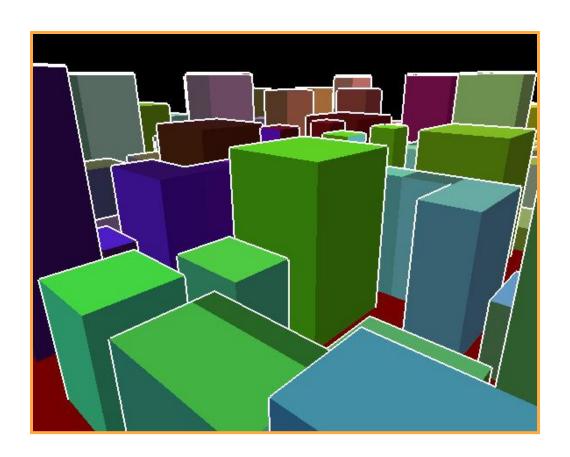


- Scan conversion
 - Determine which pixels to fill
- Shading
 - Determine a color for each filled pixel
- Texture mapping
 - Describe shading variation within polygon interiors
- Visible surface determination
 - Figure out which surface is front-most at every pixel

Visible Surface Determination



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



Depth sort

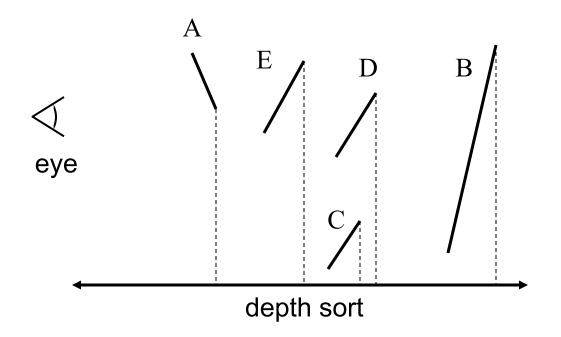


"Painter's algorithm"

Sort surfaces in order of decreasing maximum depth

Scan convert surfaces in back-to-front order,

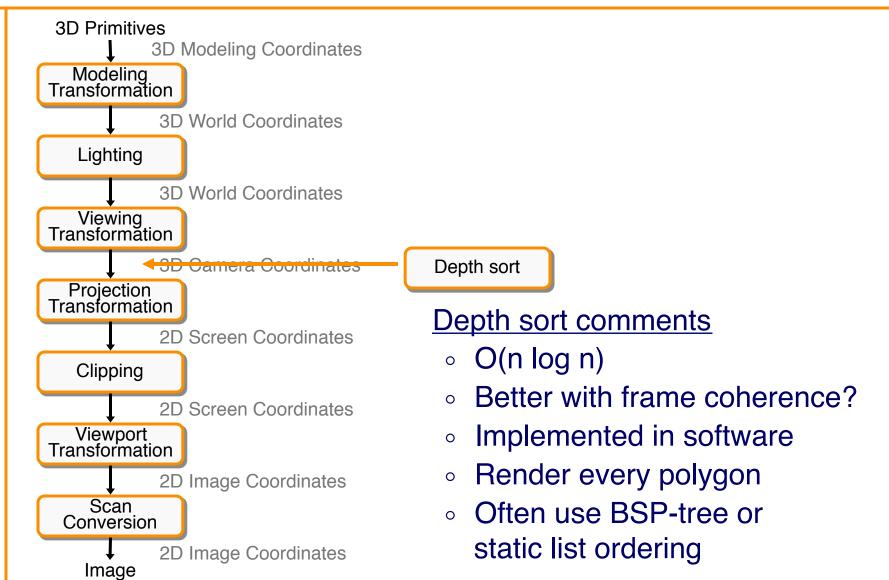
overwriting pixels





3D Rendering Pipeline



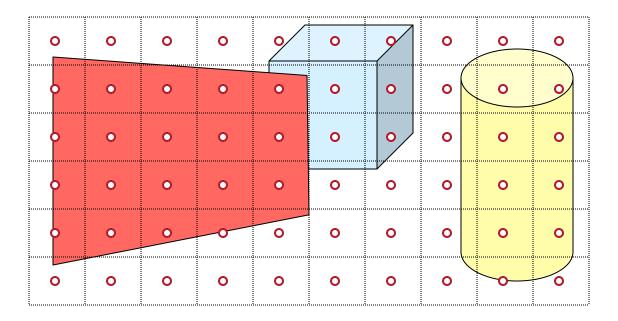


Z-Buffer



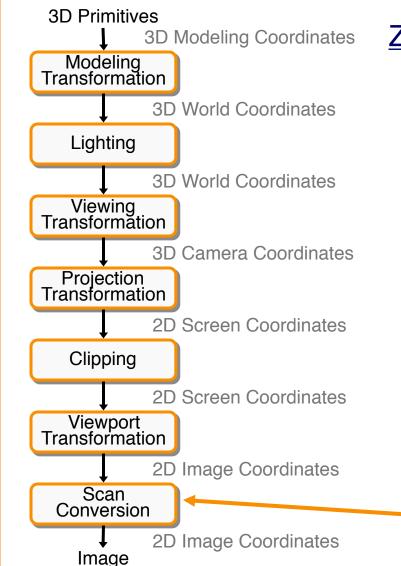
Maintain color & depth of closest object per pixel

- Framebuffer now RGBAz initialize z to far plane
- Update only pixels with depth closer than in z-buffer
- Depths are interpolated from vertices, just like colors



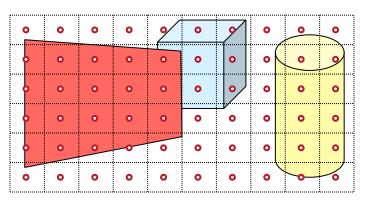
Z-Buffer





Z-buffer comments

- + Polygons rasterized in any order
- + Process one polygon at a time
- + Suitable for hardware pipeline
- Requires extra memory for z-buffer
- Subject to aliasing (A-buffer)
- Commonly in hardware



Z-Buffer

Hidden Surface Removal Algorithms

OPAQUE-OBJECT



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

A Characterization of Ten Hidden-Surface Algorithms

ALGORITIONS										
		COMPARIS	SON ALGORITHMS	OBJECT SPACE	(partly each)	IMAGE SPACE	DEPTH PRIORIT	TY ALGORITHMS		_
						PAGE SPACE	1			
edges edges volumes							\			
			Care instance	LIST PRIORITY		area sampli	ng	point sampling		
					ALGORITHMS /	dynamicall,	1			
					priority	computed	\			
		•	•	•	•	,	7			
	APPEL	GALIMBERTI, et al	LOUTREL		L	1				
	1967	1969 et al	1967	ROBERTS 1963	SCHUMACKER, et al 1969	NEWELL, et al	WARNOCK 1968	WATKINS 1970	ROMNEY, et al 1967	BOUKNIGHT 1969
RESTRICTIONS	TP,NP	TP,NP	TP,NP	TP, CC, CF, NP	CF, NP, LS (TP)	None	(TR) None	Kone	TR,CF,NP	1909
	Promote visibility	Promote visibility	Promote visibility		Frame coherence	None used				2000000
COHERENCE	of a vertex to all	of a vertex to all edges at vertex	of a vertex to all edges at vertex		in depth No X coherence used	None used		nline X erence	De C	nline
SORTING	Back Edge Cull 1) Edges separating	Back Edge Cull Description Back Edge Cull Back Edge Cull	Back Edge Cull 1) Edges separating	Back Edge Cull Dedges separating	Intra-Cluster Priority	7 507	2 Sort (Opt) 1) Faces, max 2	Y Sort	Y 9	Y Sort 1) Edges, Min Y
What,	back-facing planes	back-facing planes	back-facing planes 2) Dot product with	back-facing planes 2) Dot product with	1) Faces - visibility	2) 100	2) Comparison of	2) par on	gons, Y empoints 2) Comparison	2) Comparison
what prop-	normals & topology 3) Cull	normals & topology 3) Cull		normals & topology	normals	3) : 2gm	max points 3) n log m	4) Ti e o	(3) 2 bucket	(4) Table of lists
(2)	normals & topology 3) Cull 4) List of edges, E _s 5) 1, E _t	4) List of edges, E _s 5) 1, E _e	3) Cull 4) List of edges,E _s 5) 1, E _t	4) List of edges,E	Ordered (ab)	Ordered table	4) Ordered 5) 1. F	3, 1,	4) Table of lists 5) 1, Fr	5) 1, E _r
Method	a reason and a second) 0, (oi:					
Type (3)	Contour Edge Cull 1) Edges separating front 4 back faces			0 4 60	Priority Clusters	Newell Special		x Merge w1) Edges, X value	X Sort 1) Edges, X value	1) Merge 1) Edges, X value
(4)	2) Dot product with	(Omatted)	(Omitted)	W 21 400	2) Dot product with separating planes	2) Dept box and	n X and Y, sum of	3) Merge (ordred)	2) Comparison	2) Comparison 3) Merge (ordered) 4) Linked list
Result	3) Cull 4) List, E _c	2000-000		E	3) Prefix scan binary tree	3) bbli vp	angles g3) Radix 4 subdivi-	4) 2-way linked	4) Table of lists 5) n, S,	4) Linked list 5) E _r , 2S _L (edges)
(5) Number per	5) 1, E, C	To the second se		, Es			sion with overlap 4) Stacks of unordered tables	5) E _r , S _k		1
frame, num- ber of ob-							5) L _v , F _r /factor 1			
jects	Initial Visibility 1) Ray to vertex	Initial bil	1 to vertex	Edge/Volume Test 1) Edges, visit	A Cul	Y Sort 1) Face segment	Depth Search 1) Surrounder faces	X Sort	X Priority Search I) Edges, X value	1) Edges, X value
(merge) Number of	2) Benth	against a faces 2) Depth,	2) Betweenness,	relative old s 2) Lin	2) t j duct th	by Y range 2) Y intercept	2) 4-corner compare 3) Exhaustive	λ left 2) Comparison	(2) Comparison	2) Comparison
Number of new entries per frame,	Surry dedress 3) Exhau vi arv 4) Quanti tiv	Surround Exhau search Quantitative	surroundedness 3) Exhaustive sear 4) Quantitativ	Pro amni.	3) Cull 4) Smaller ordered table	3) Bucket 4) None	4) Answer/failure 5) L, F,/factor 2	5) Bubble 4) 2-way linked	4) Active segment	4) 1-way linked list 5) N, 2S _g (edges)
length of	vis it (va		visibility 5) tobaccts	5) dges,	5) 1, F _t	5) F * split faces	1 '	11st 5) n, Sa	5) n, m	, ., .,
ards.				roojects				1 (50)	İ	
earches	1 otersect one F	Edge Intersection	In set on one E		Y Cull I) Faces by Y extent	X Merge	TV Sort (Opt)	Span Cull	2 Search 1) Segments, depth	Z Search 1) Segments, depth
ength o	2) Penetration	2) Inters	Ith I her sect in		2) Mini-max on X intercepts	X intercept 2) Comparison		with sample span	[2] Linear equations	1) Segments, depth 2) Linear equations
	with sweep triangle	vrdey)	cull (unordered)		 Cull (unordered) X intercepts of 	3) Ordered merge 4) Ordered list	needed	2) Double comparison 3) Cull ordered list	(3) Search (unordered	and comparison 3) Search of un-
	4) Intersection list 5) Es, Ec	Interse ist	4) Intersection list 5) E, E, - 1		relevant segments 5) n, E,	5) S _r , S _v /2		5) n*S _v * f (>1), S	4) Visible segment 5) n*2S ₁ ,D _C	ordered active list 4) Visible segment
	Sort Alone Edge	1) Intersections on	Sort Along Edge 1) Intersections on		X Sort			Z Search	(Omitted if X	5) n*2S _f , D _c
	1) Intersections on edge, ordering	edge, ordering	edge, ordering		1) Segments 2) Counters			1) Segments, Z 2) Depth by	last time)	1
	2) Comparison 3) Bubble	3) 4) Answer	3) 4) Answer		3) Hardware 4) Segments at		1	logarithmic search 3) Search (unordered	l n	
	4) Answer 5) E _s , X _y /E _s	5) E, X,/E,	5) E, X,/E,		this X 5) nm, S ₂			4) Visible segment 5) n*Sy*f(>1), Dc	ĺ	
	(Omit if well hidden)	(must be done)	(Omit if well hidden					,		
					Priority Search 1) Segments, priorit 2) Logic network	,	I	1	ŀ	l
					3) Logic network					
					4) Visible segment 5) nm, Sg					
	ı		I	I		1			FC	. 1

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

[Sutherland '74]

Rasterization Summary

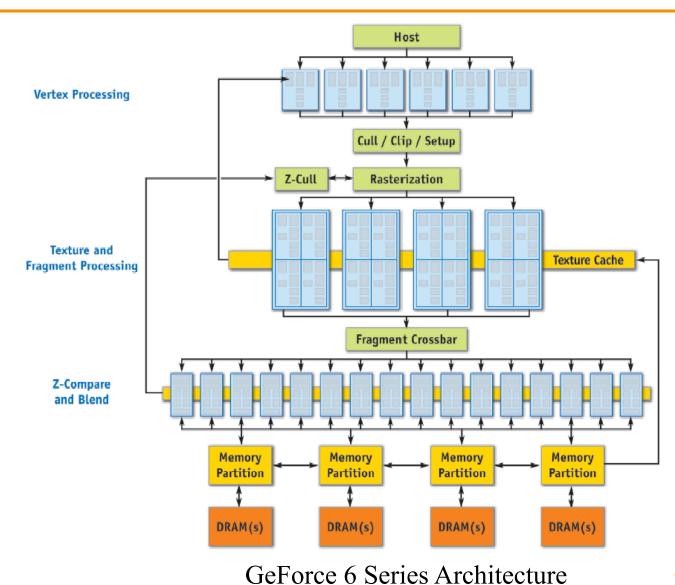


- Scan conversion
 - Sweep-line algorithm
- Shading algorithms
 - Flat, Gouraud
- Texture mapping
 - Mipmaps
- Visibility determination
 - Z-buffer

This is all in hardware

GPU Architecture



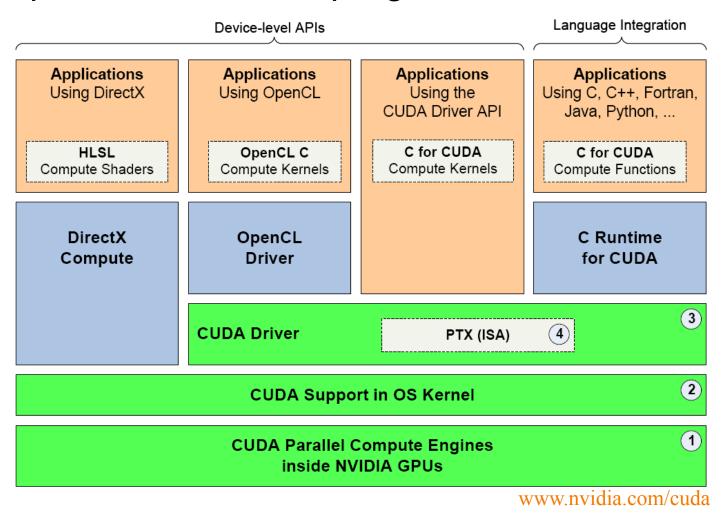


GPU Gems 2, NVIDIA

Actually ...



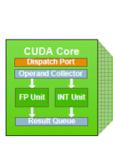
Graphics hardware is programmable

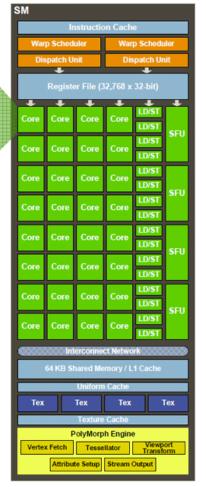


Trend ...



GPU is general-purpose parallel computer







www.nvidia.com/cuda