3D Rasterization II

COS 426

3D Rendering Pipeline (for direct illumination)

3D Primitives

↓

Modeling Transformation

↓

Lighting

↓

Viewing Transformation

↓

Projection Transformation

↓

Clipping

↓

Viewport Transformation

↓

Scan Conversion

↓

Image

Rasterization

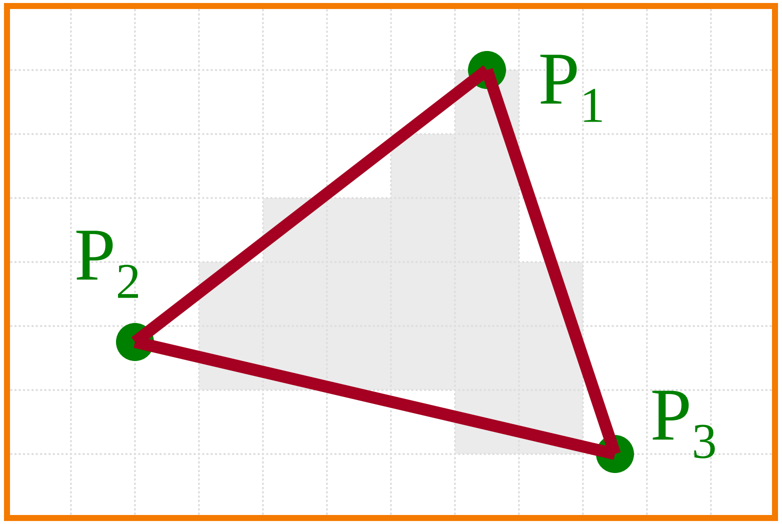
- Scan conversion
  - Determine which pixels to fill
- Shading
  - Determine a color for each filled pixel
- Texture mapping
  - Describe shading variation within polygon interiors
- Visible surface determination
  - Figure out which surface is front-most at every pixel

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

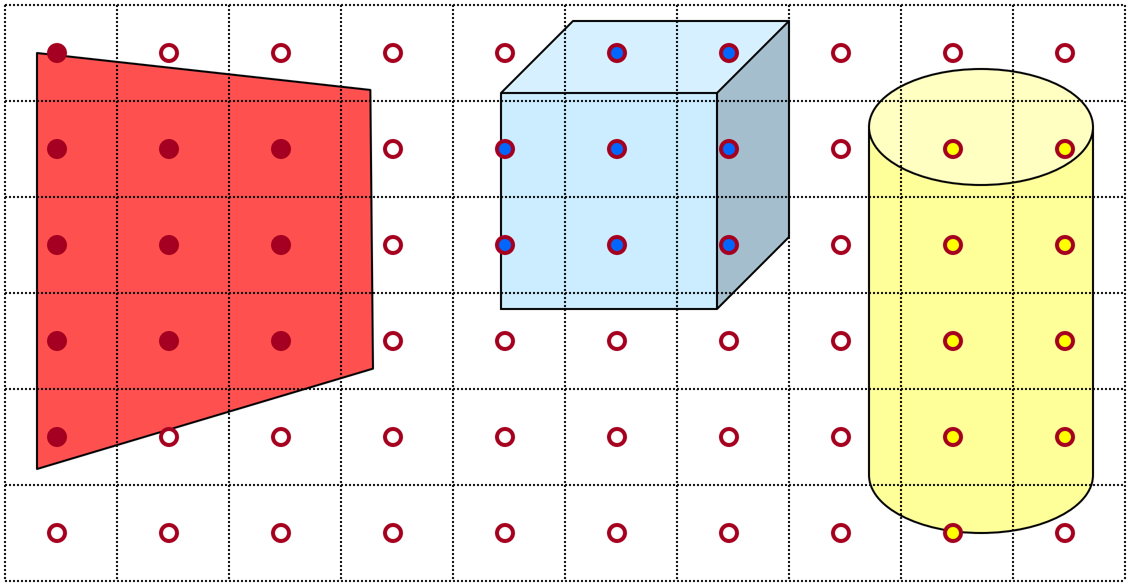
- How do we choose a color for each filled pixel?



Emphasis on methods that can be implemented in hardware

Ray Casting

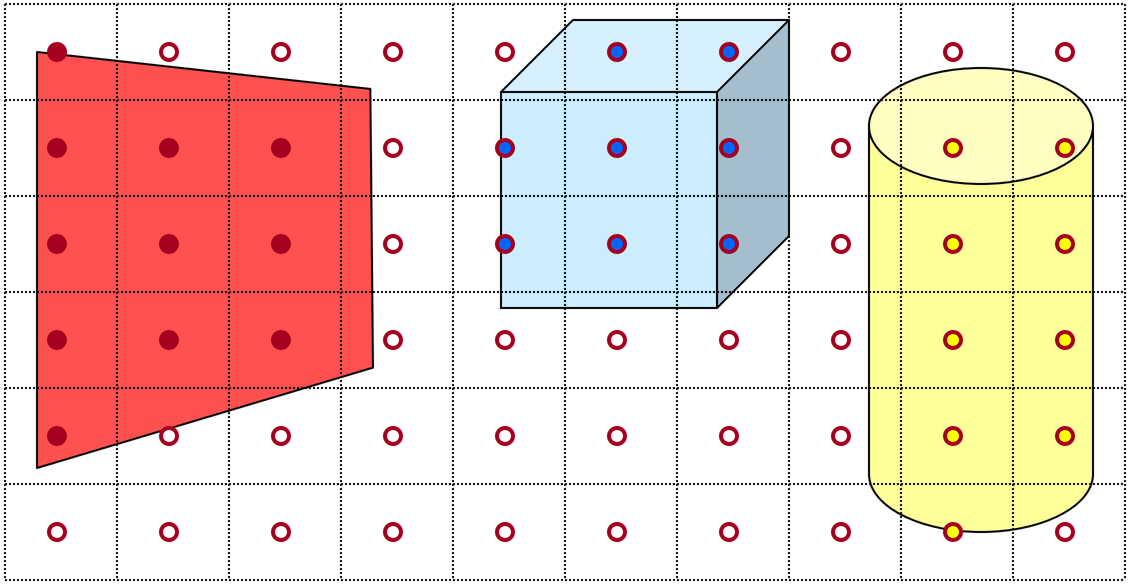
- Simplest shading approach is to perform independent lighting calculation for every pixel



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

Polygon Shading

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



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

Polygon Shading Algorithms

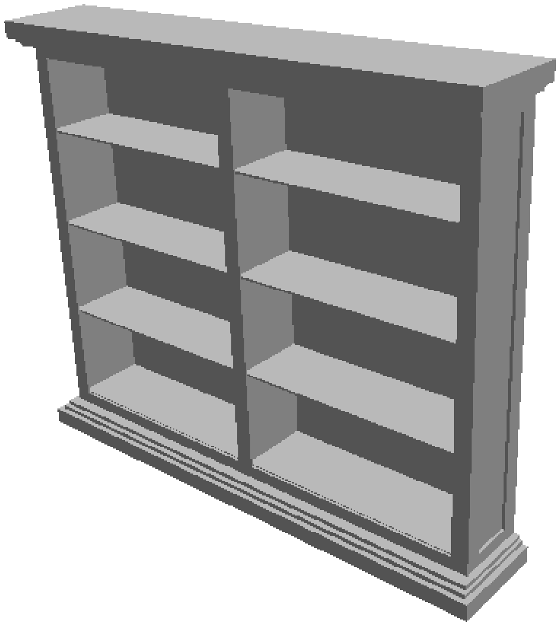
- Flat Shading
- Gouraud Shading
- Phong Shading

Polygon Shading Algorithms

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

Flat Shading

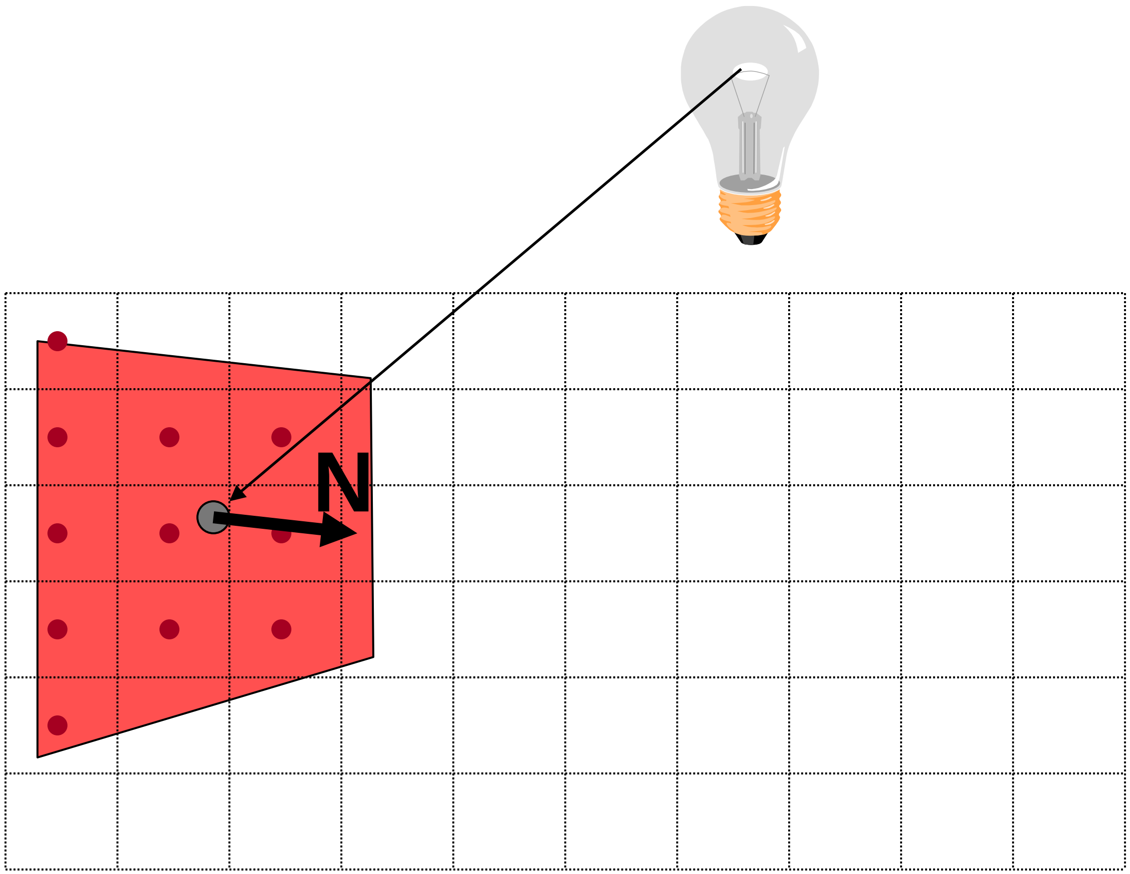
- What if a faceted object is illuminated only by directional light sources and is either diffuse or viewed from infinitely far away



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

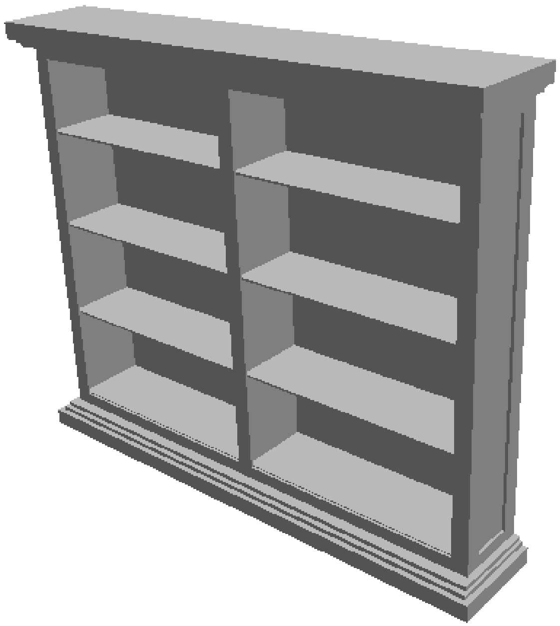
Flat Shading

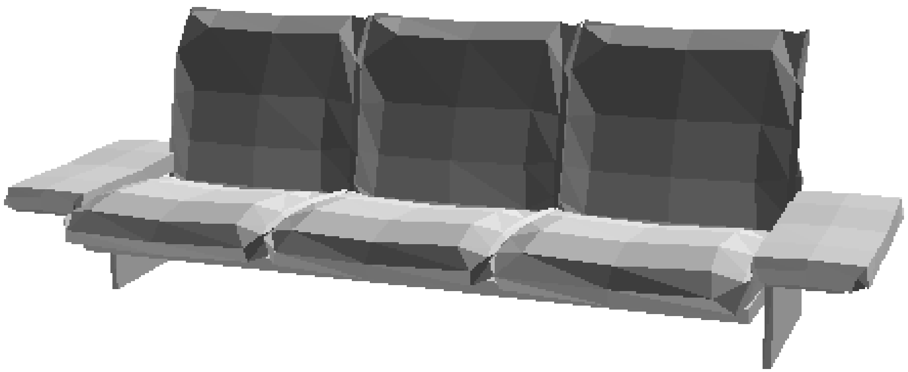
- One illumination calculation per polygon
  - Assign all pixels inside each polygon the same color



Flat Shading

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



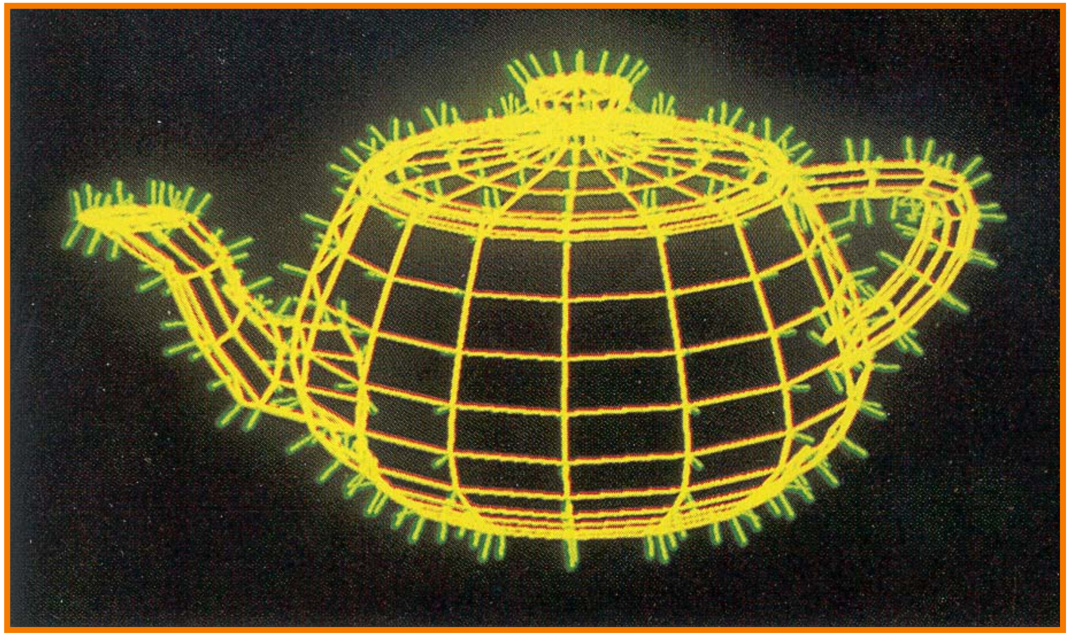


Polygon Shading Algorithms

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

Gouraud Shading

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

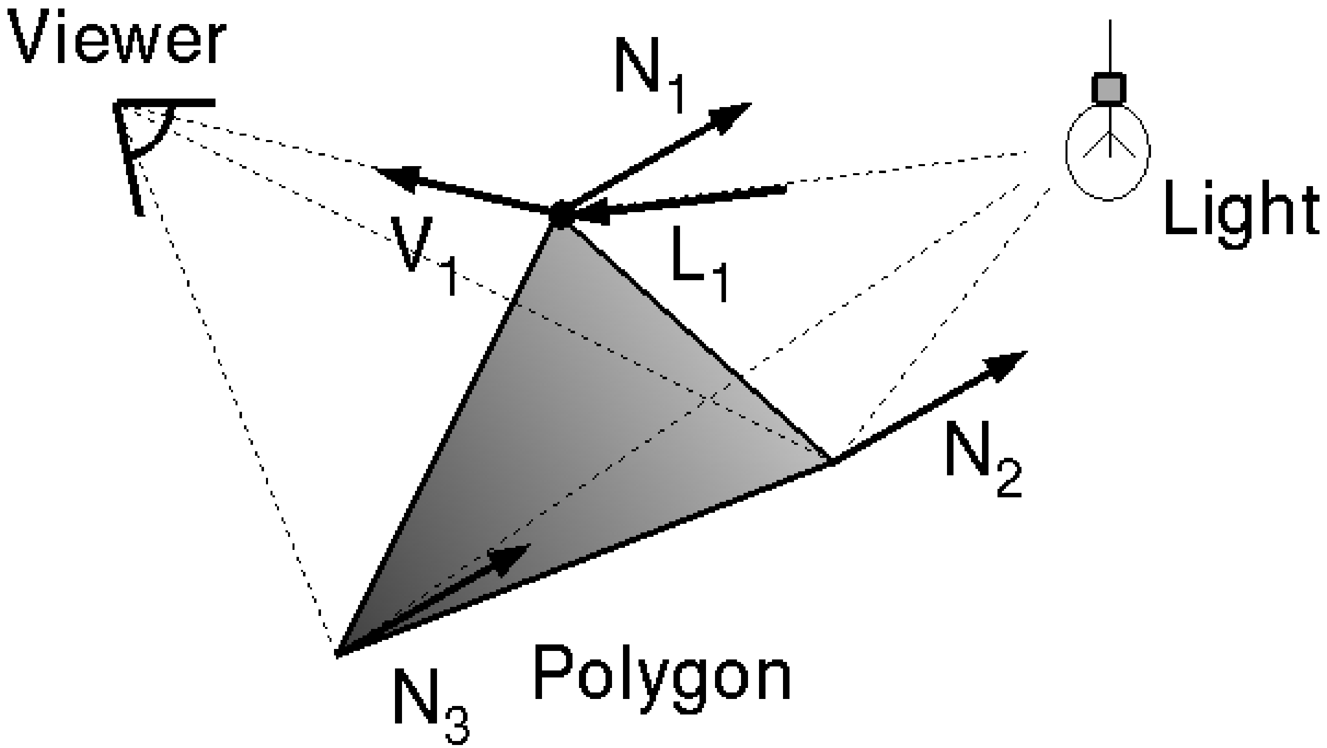


Watt Plate 7

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

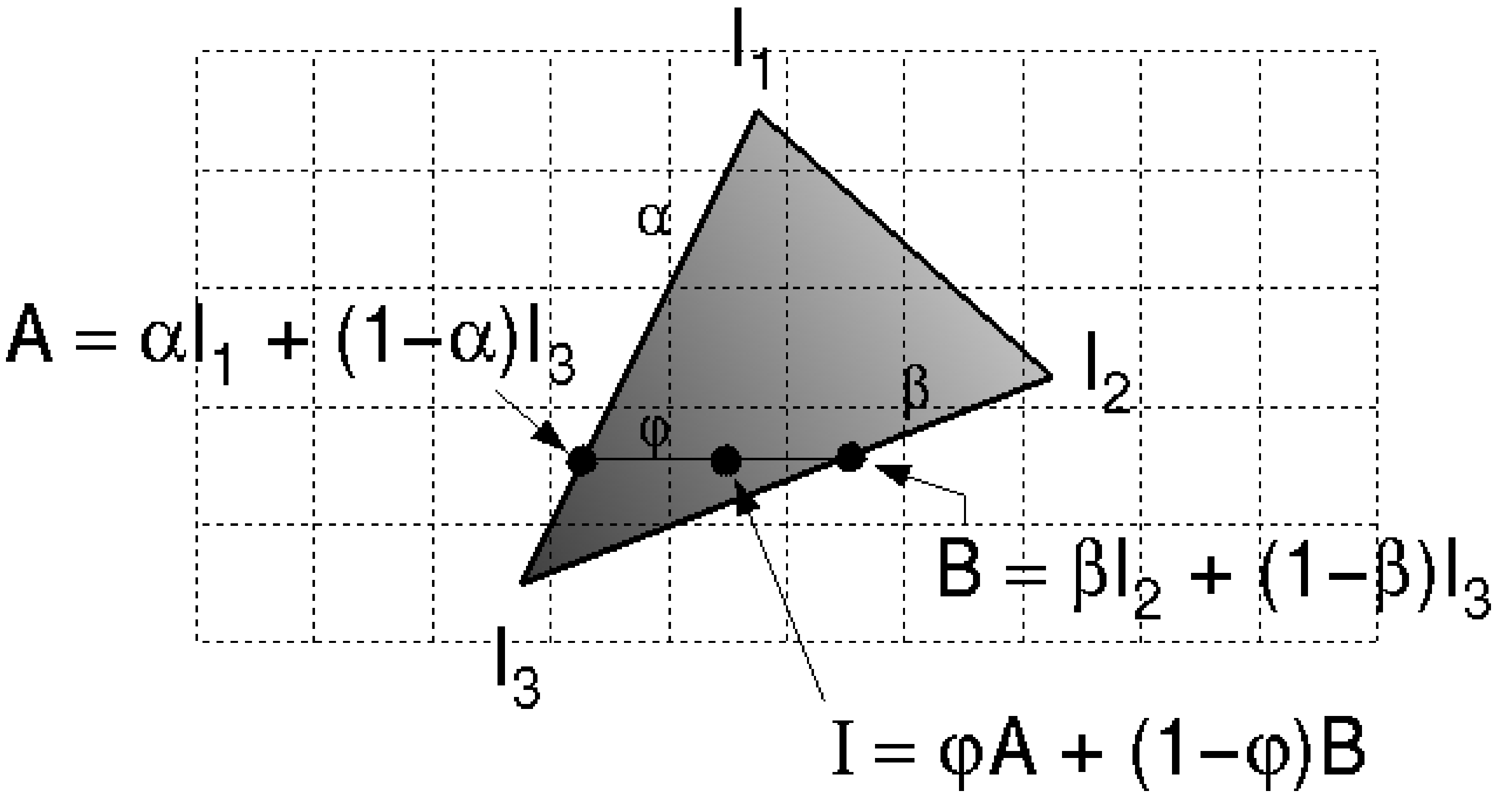
Gouraud Shading

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



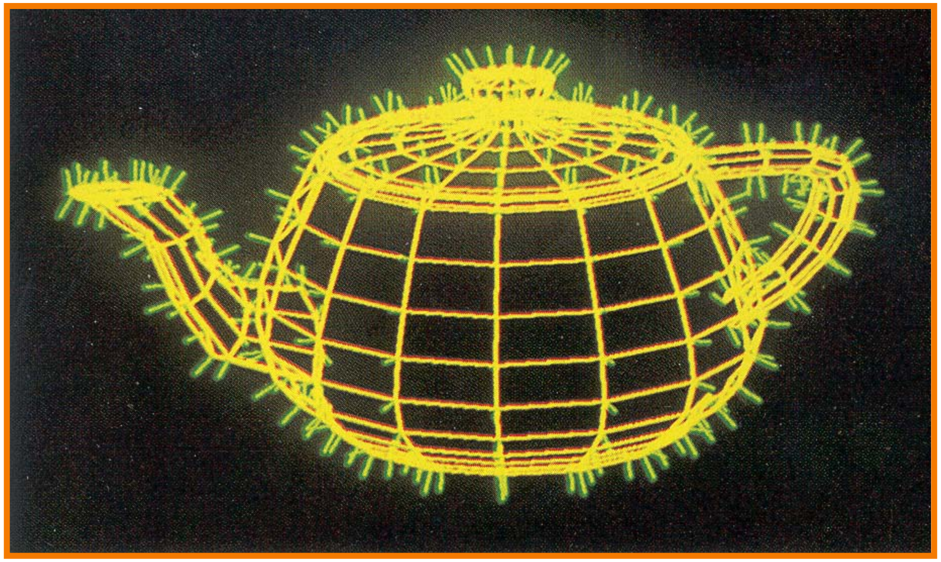
Gouraud Shading

- Bilinearly interpolate colors at vertices down and across scan lines



Gouraud Shading

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

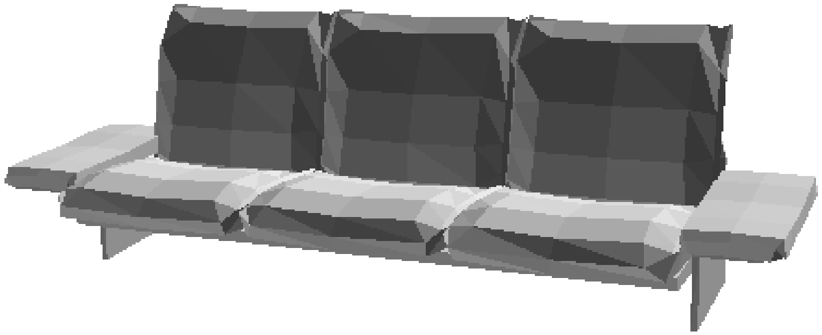


Mesh with shared normals at vertices

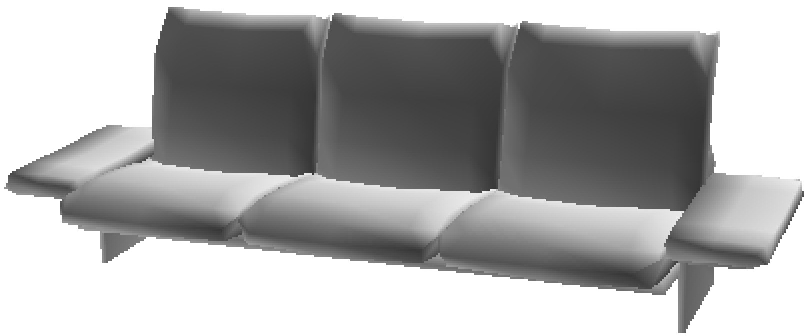
Watt Plate 7

Gouraud Shading

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



Flat Shading



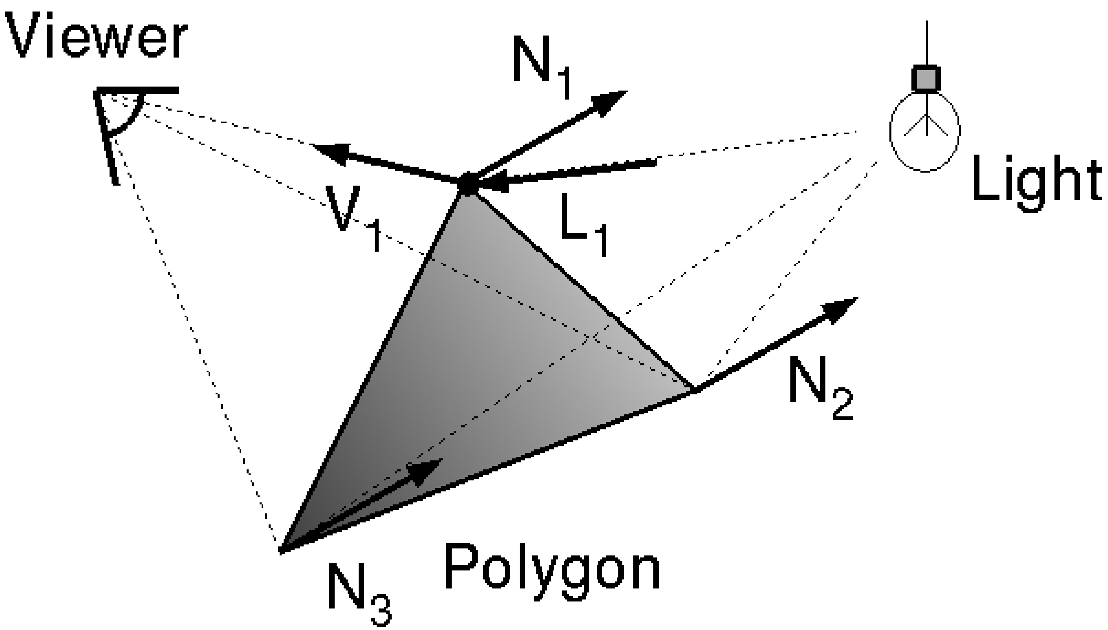
Gouraud Shading

Polygon Shading Algorithms

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

Phong Shading

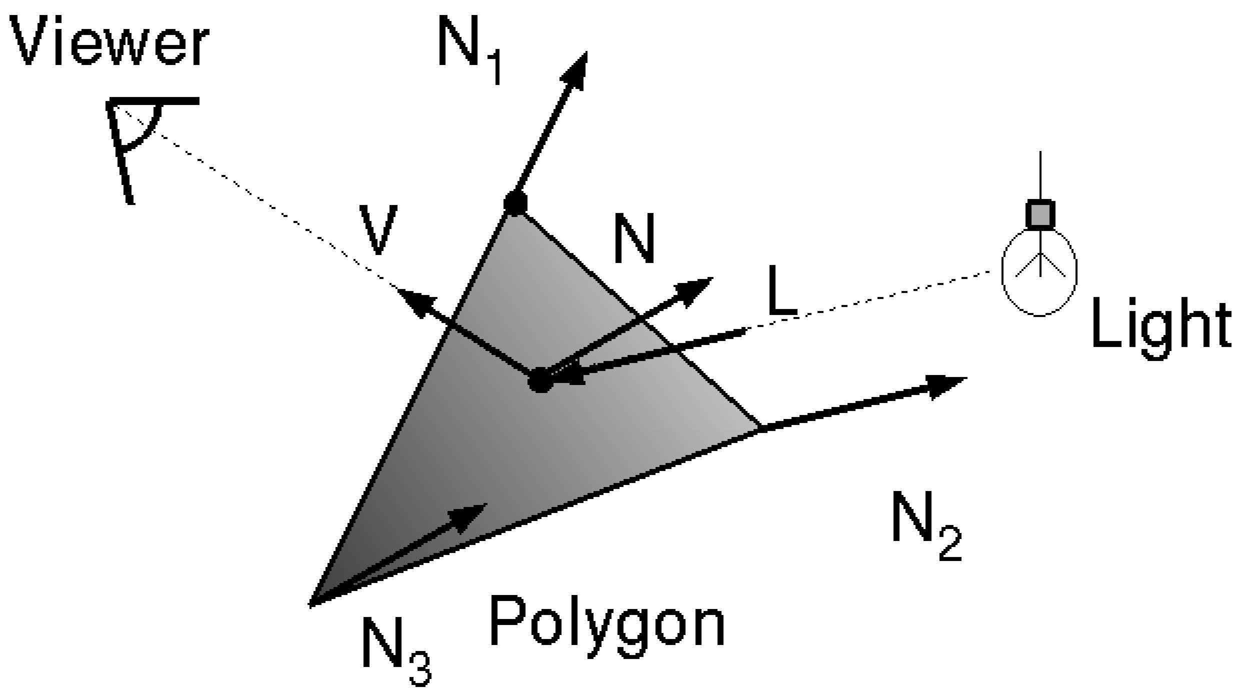
- What if polygonal mesh is too coarse to capture illumination effects in polygon interiors?



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

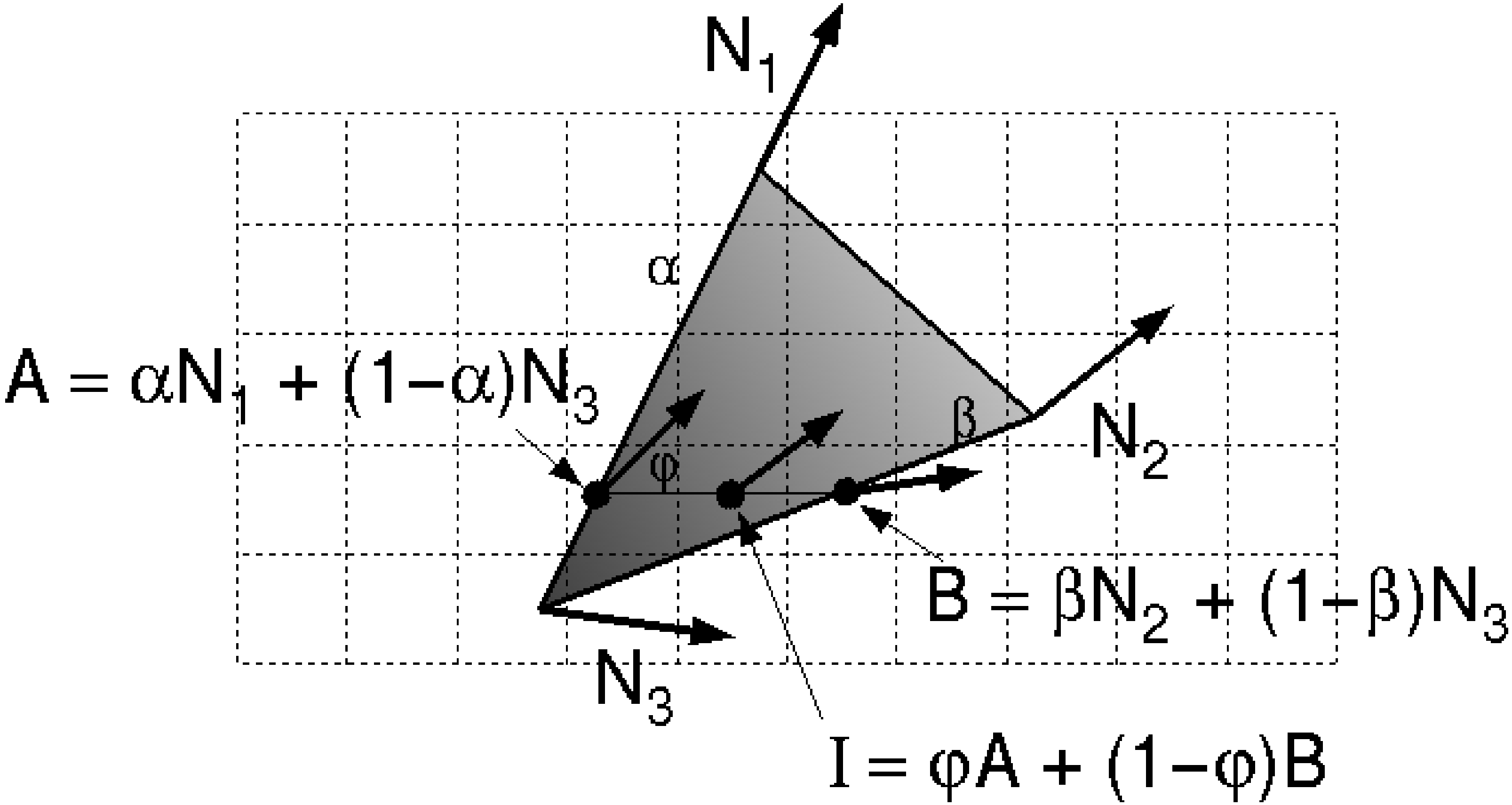
Phong Shading

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



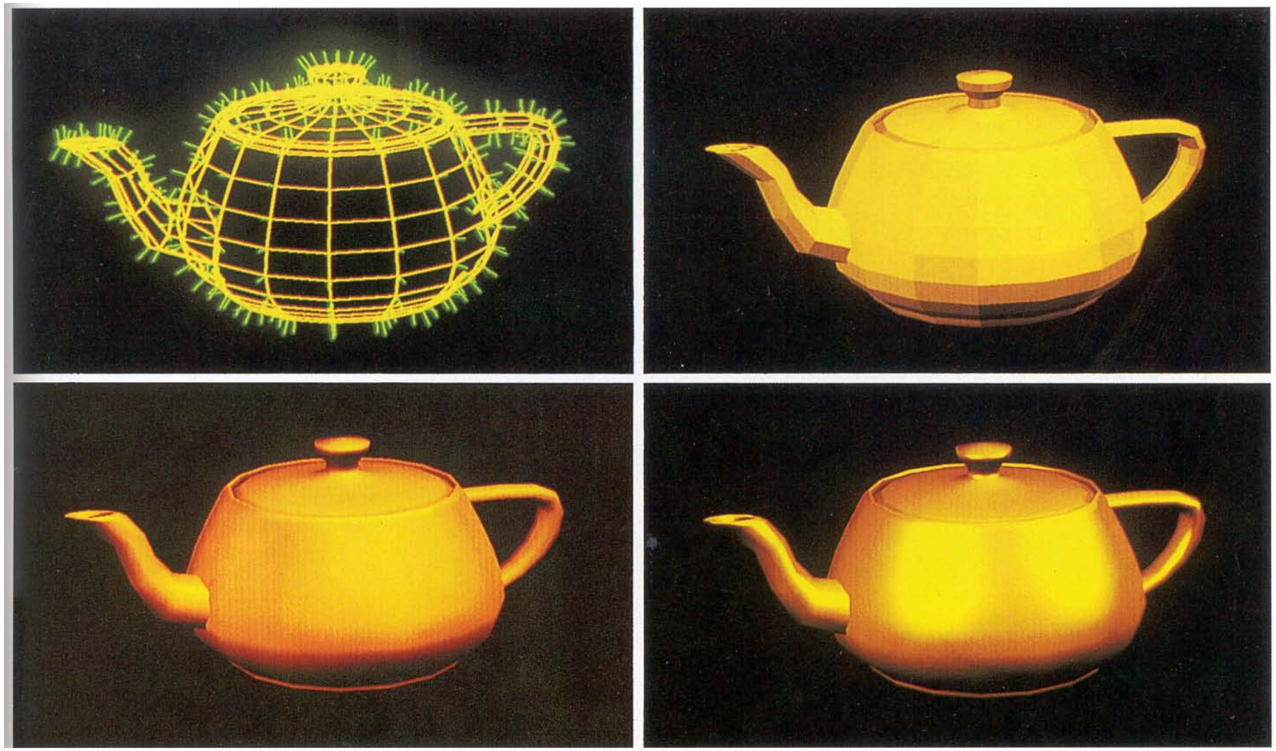
Phong Shading

- Bilinearly interpolate surface normals at vertices down and across scan lines



Polygon Shading Algorithms

Wireframe Flat

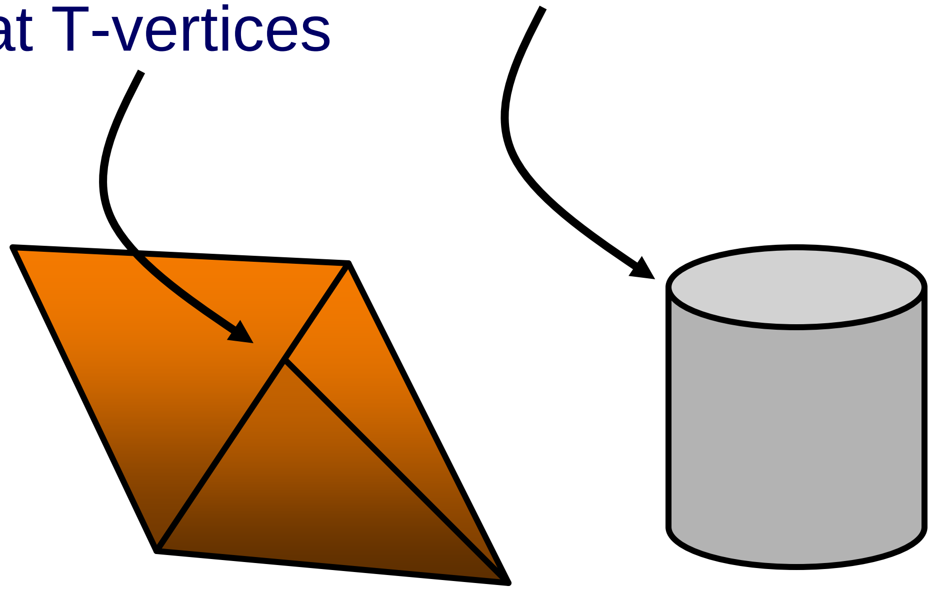


Gouraud Phong

Watt Plate 7

Shading Issues

- Problems with interpolated shading:
  - Polygonal silhouettes
  - Perspective distortion
  - Orientation dependence (due to bilinear interpolation)
  - Problems computing shared vertex normals
  - Problems at T-vertices

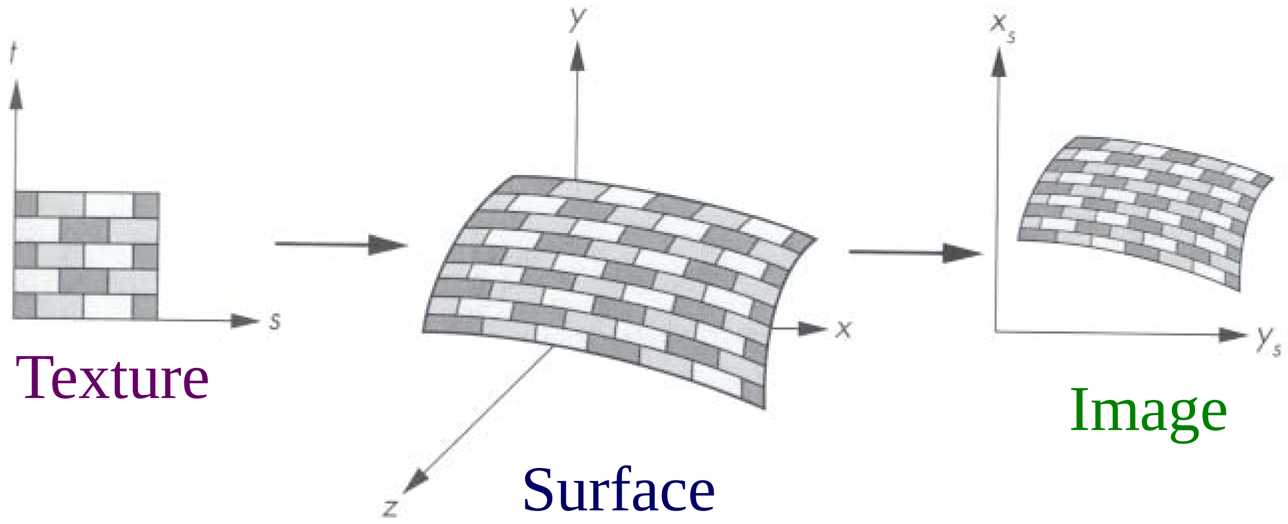


Rasterization

- 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

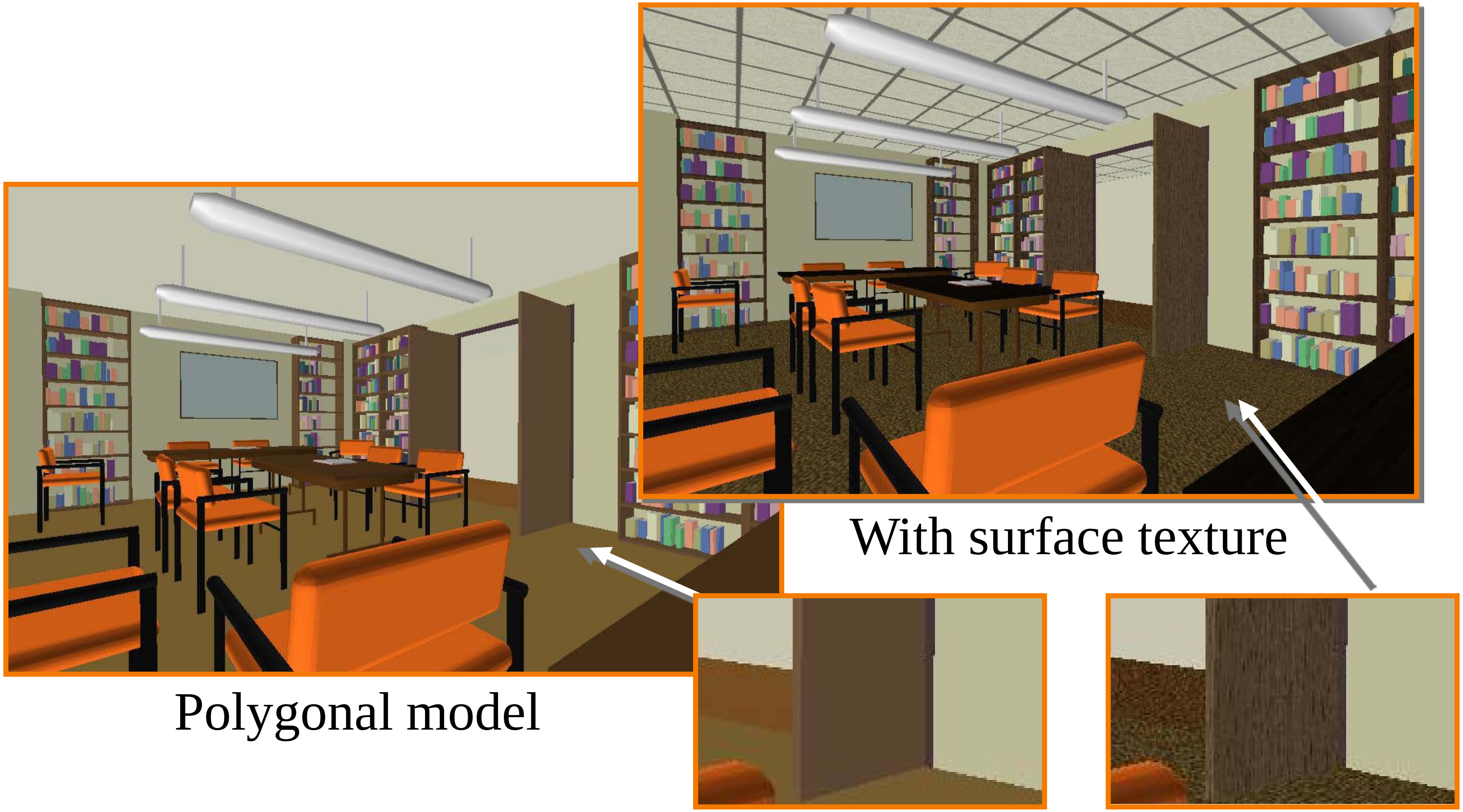
- Describe color variation in interior of 3D polygon
  - When scan converting a polygon, vary pixel colors according to values fetched from a texture image



Angel Figure 9.3

Surface Textures

- Add visual detail to surfaces of 3D objects

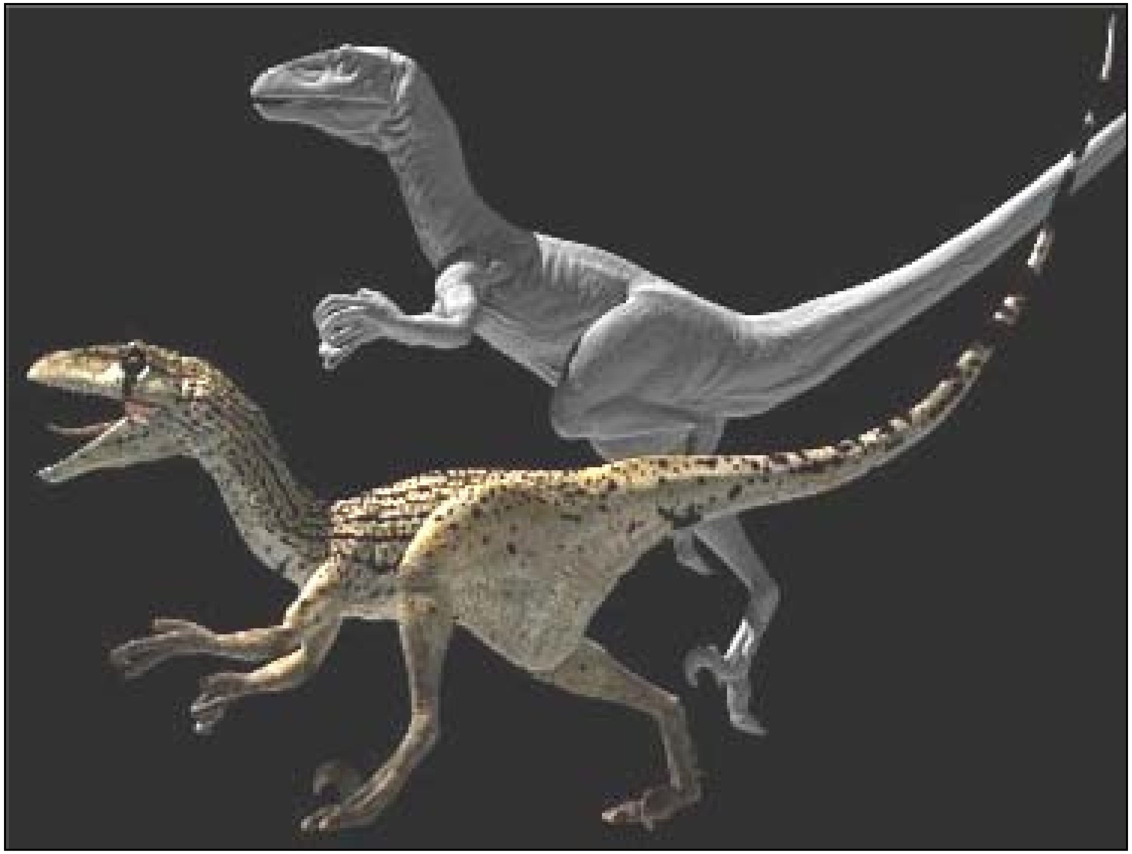


With surface texture

Polygonal model

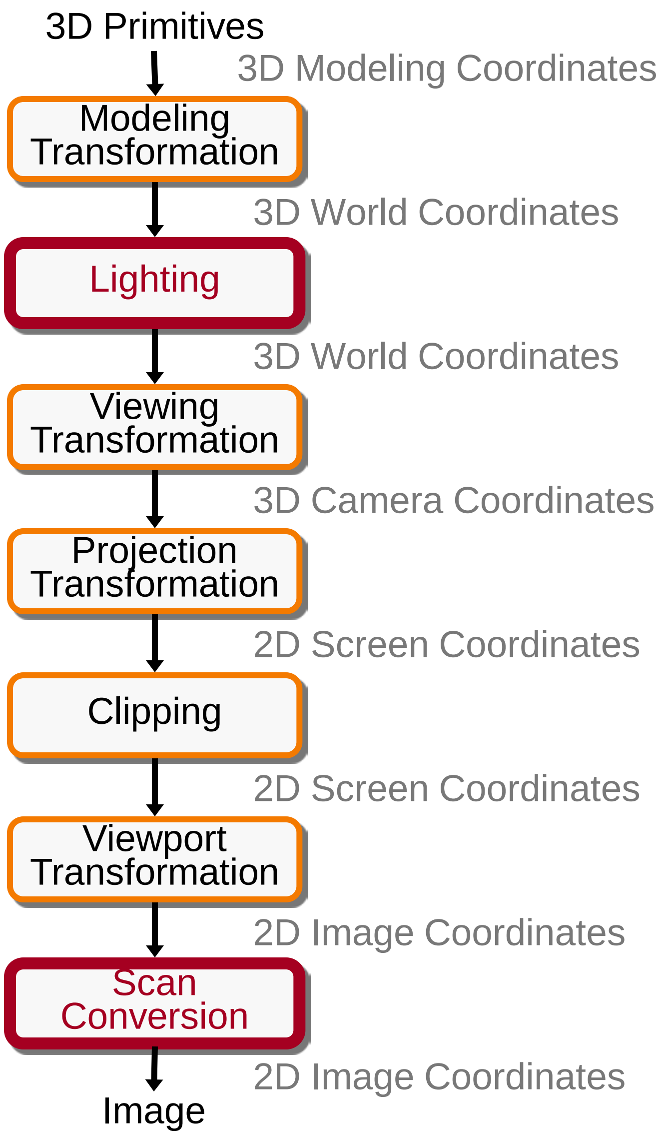
Surface Textures

- Add visual detail to surfaces of 3D objects



[Daren Horley]

3D Rendering Pipeline (for direct illumination)



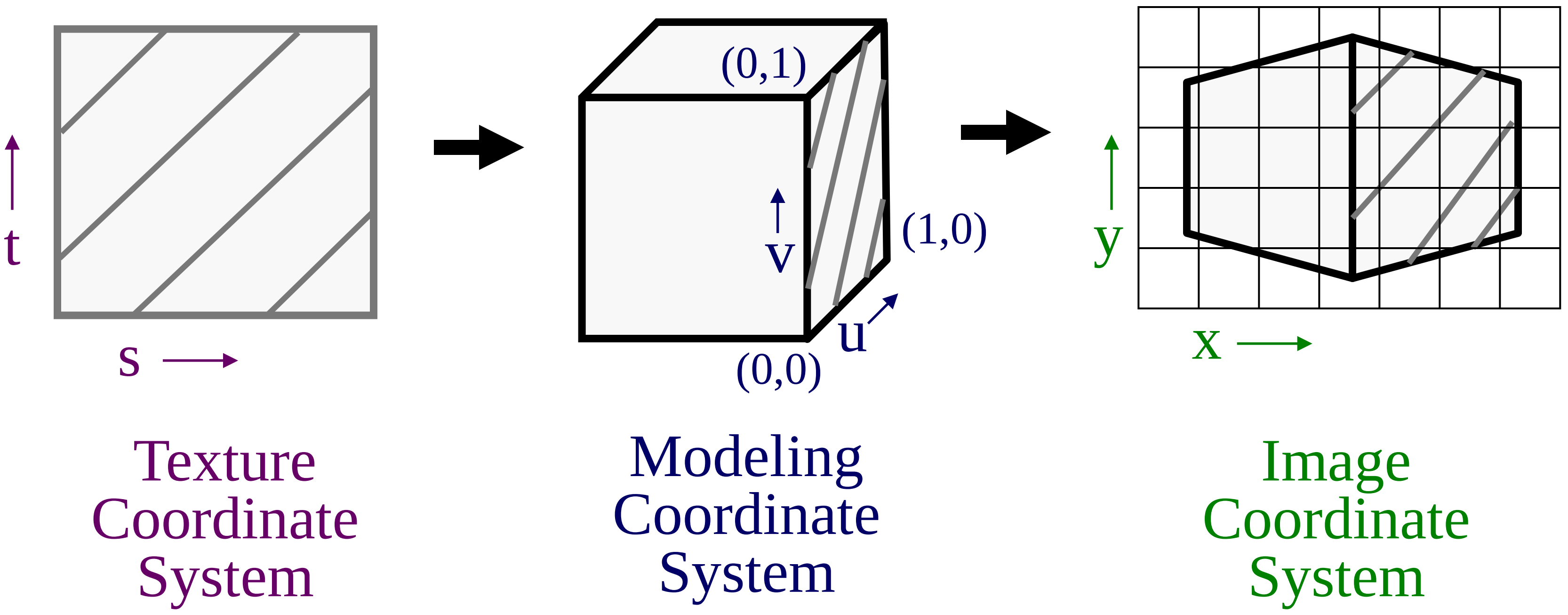
Texture mapping

Texture Mapping Overview

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

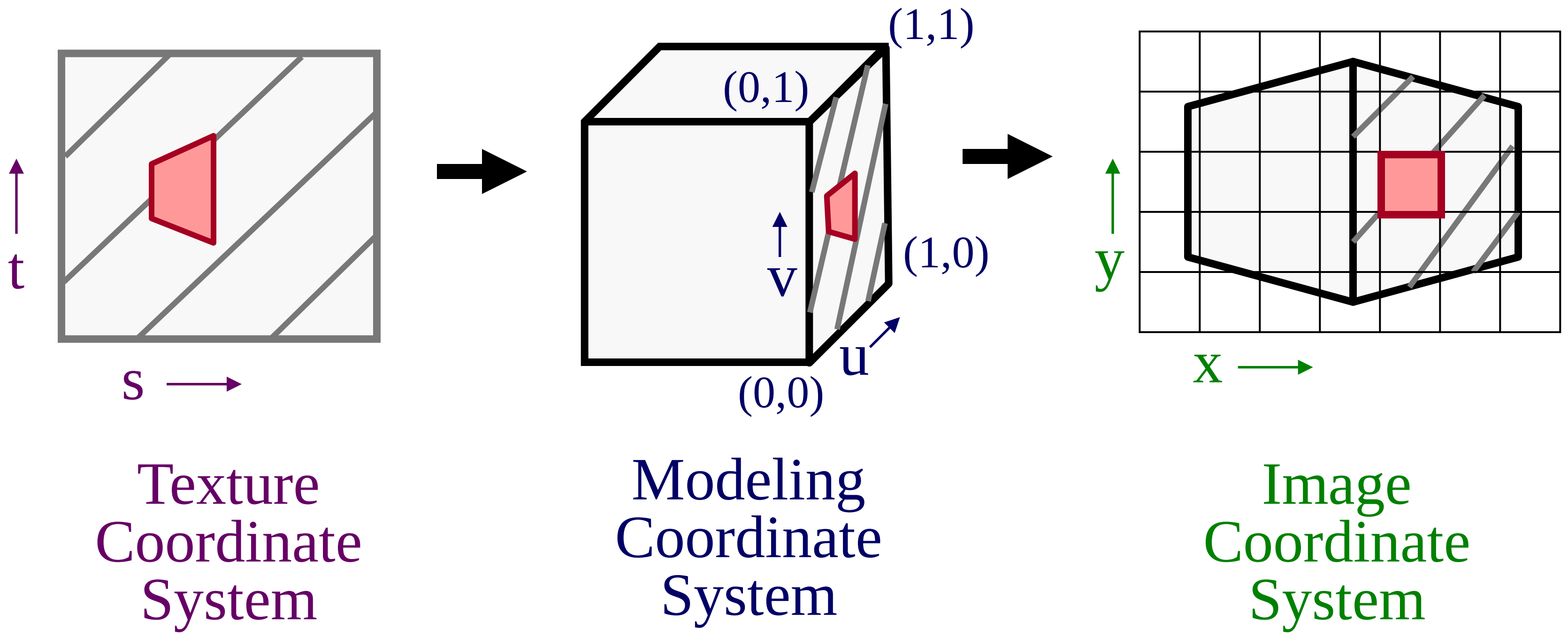
Texture Mapping

- Steps:
  - Define texture
  - Specify mapping from texture to surface
  - Lookup texture values during scan conversion



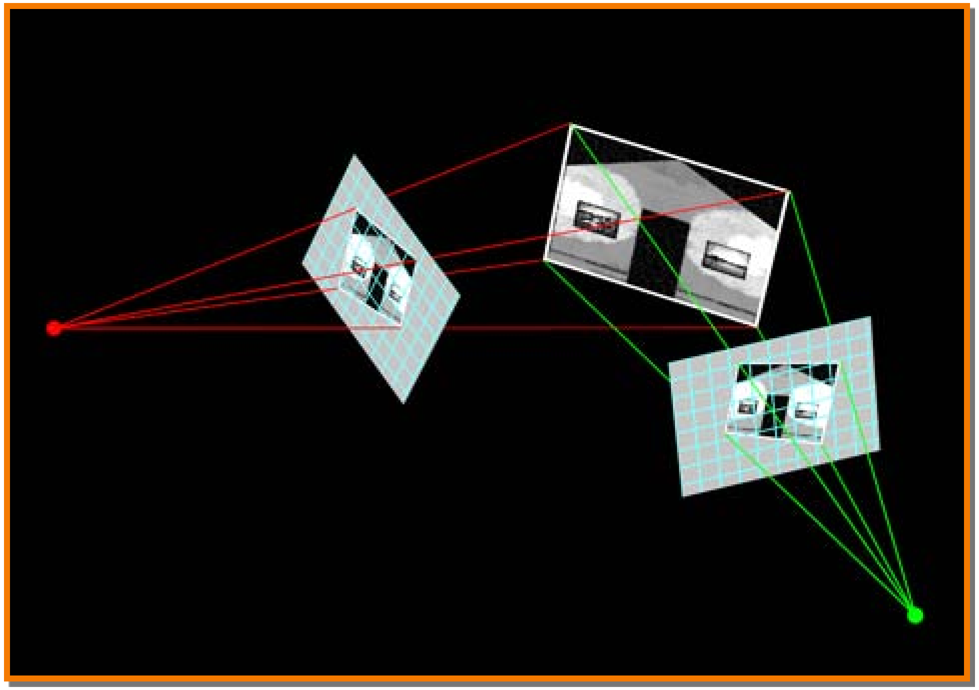
Texture Mapping

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



Texture Mapping

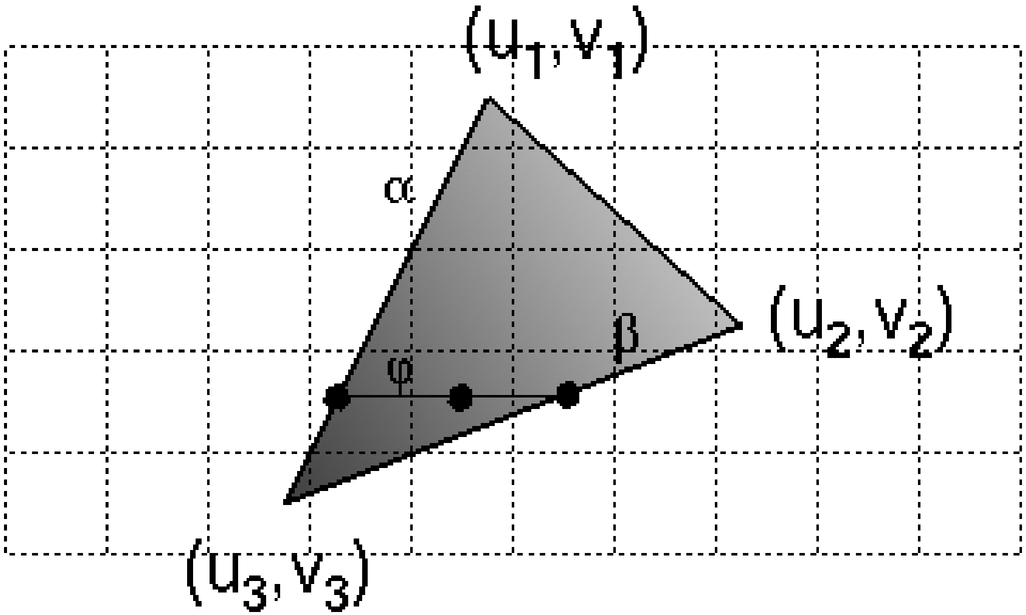
- Texture mapping is a 2D projective transformation
  - texture coordinate system: (t,s) to
  - image coordinate system (x,y)



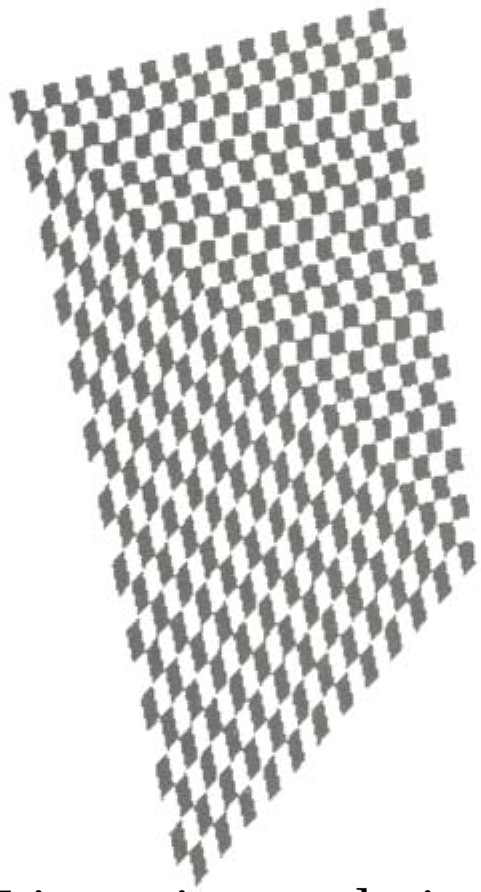
[Allison Klein]

Texture Mapping

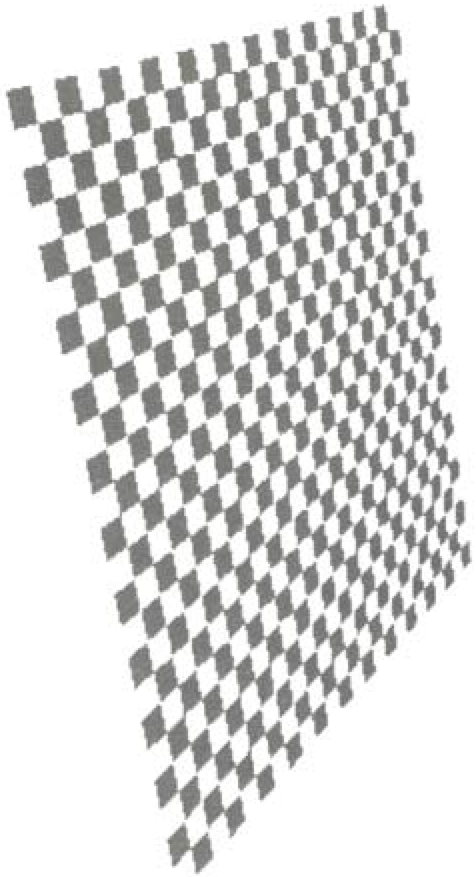
- Scan conversion
  - Interpolate texture coordinates down/across scan lines
  - Distortion due to bilinear interpolation approximation
    - » Cut polygons into smaller ones, or
    - » Perspective divide at each pixel



Texture Mapping



Linear interpolation
of texture coordinates



Correct interpolation
with perspective divide

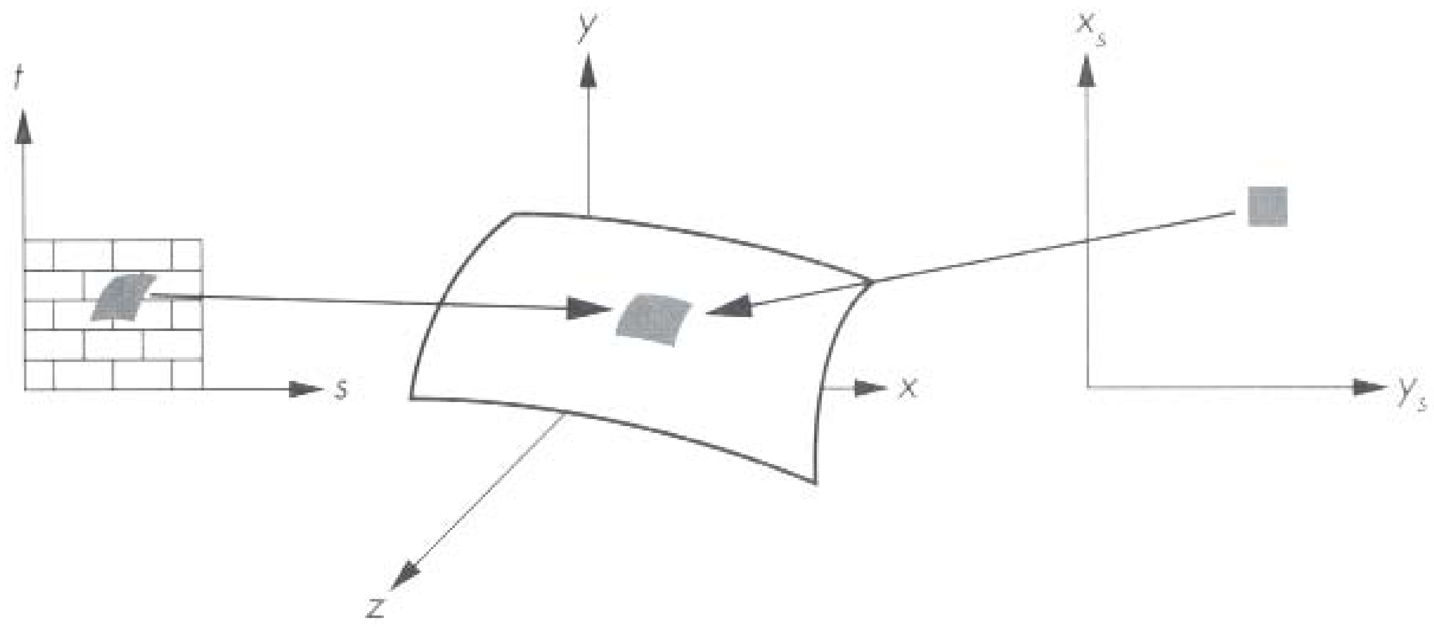
Hill Figure 8.42

Texture Mapping Overview

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

Texture Filtering

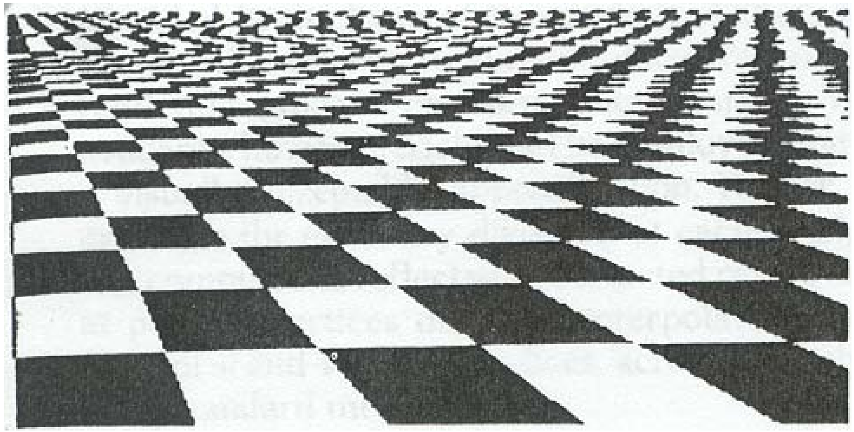
- Must sample texture to determine color at each pixel in image



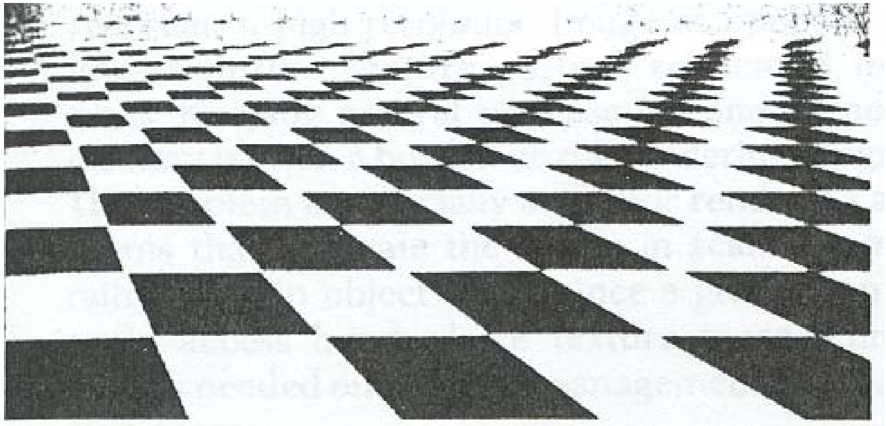
Angel Figure 9.4

Texture Filtering

- Aliasing is a problem



Point sampling

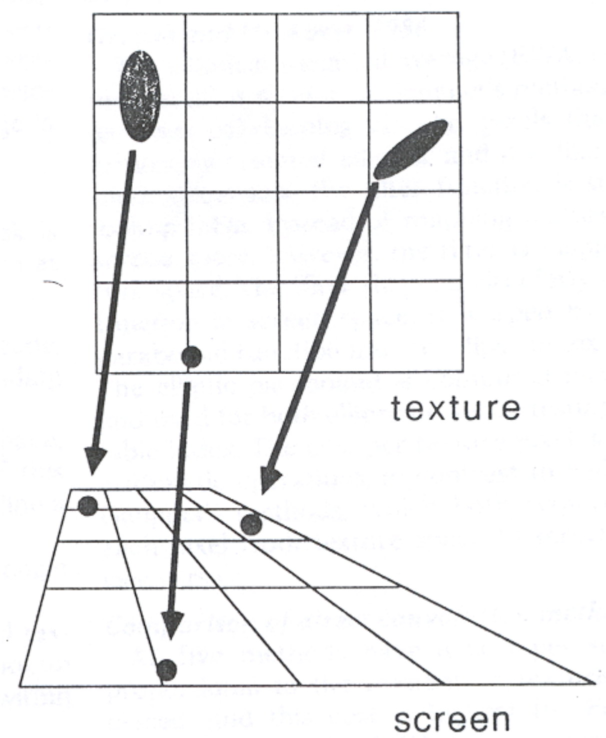


Area filtering

Angel Figure 9.5

Texture Filtering

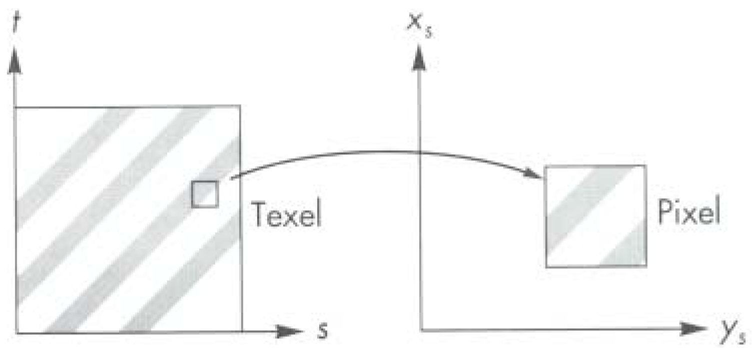
- Ideally, use elliptically shaped convolution filters



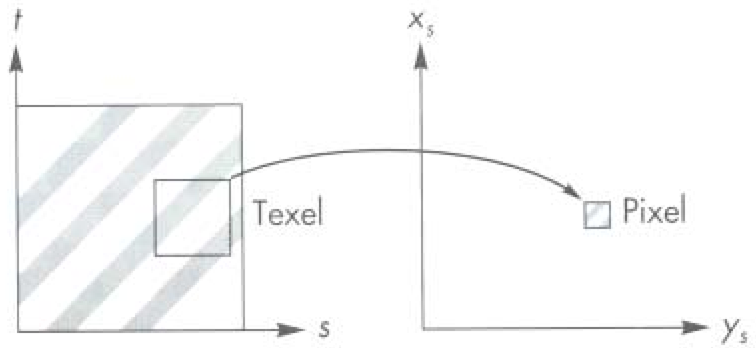
In practice, use rectangles

Texture Filtering

- Size of filter depends on projective warp
  - Can prefiltering images
    - » Mip maps
    - » Summed area tables



Magnification

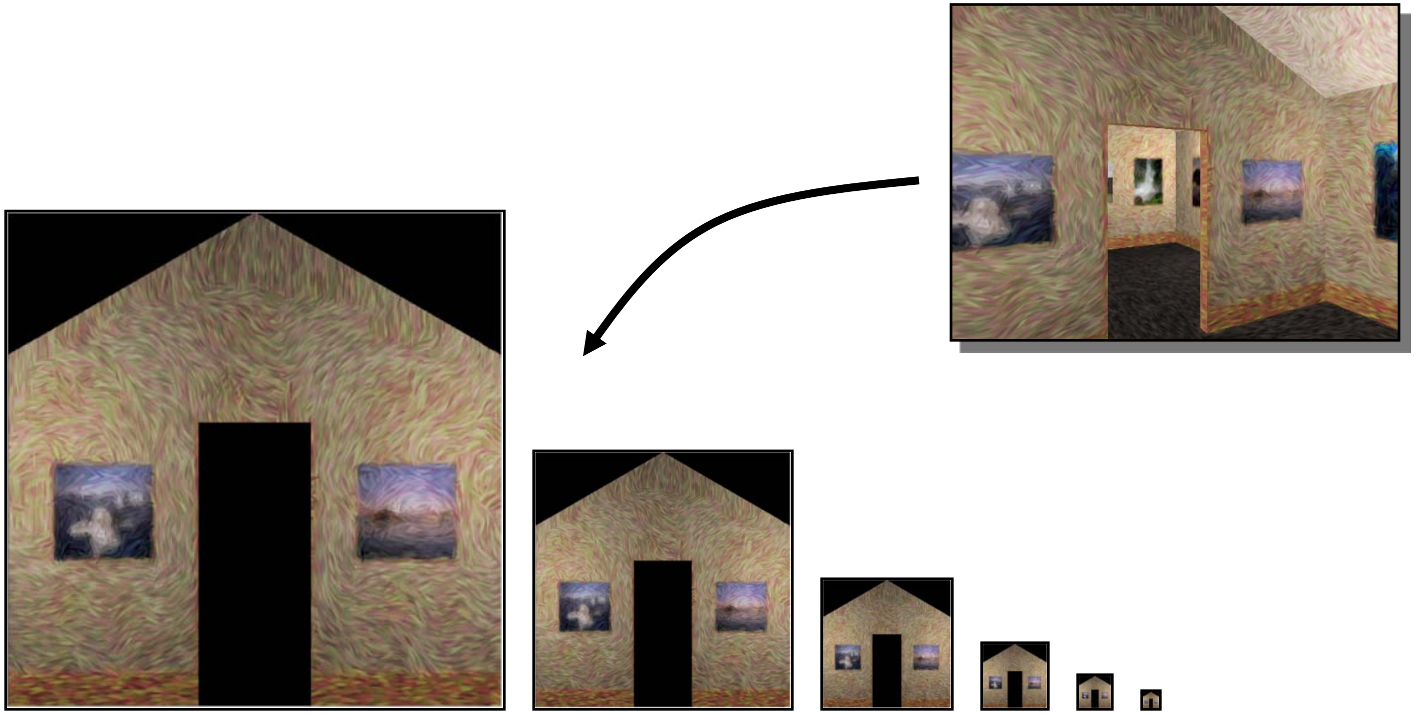


Minification

Angel Figure 9.14

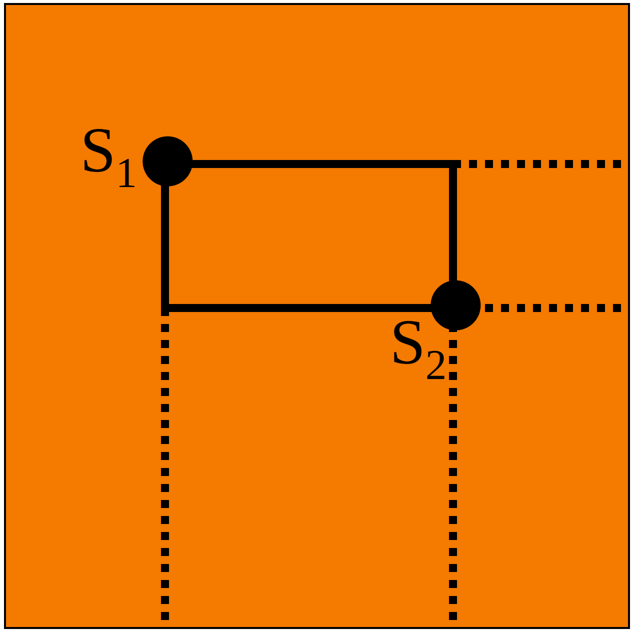
Mip Maps

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



Summed-area tables

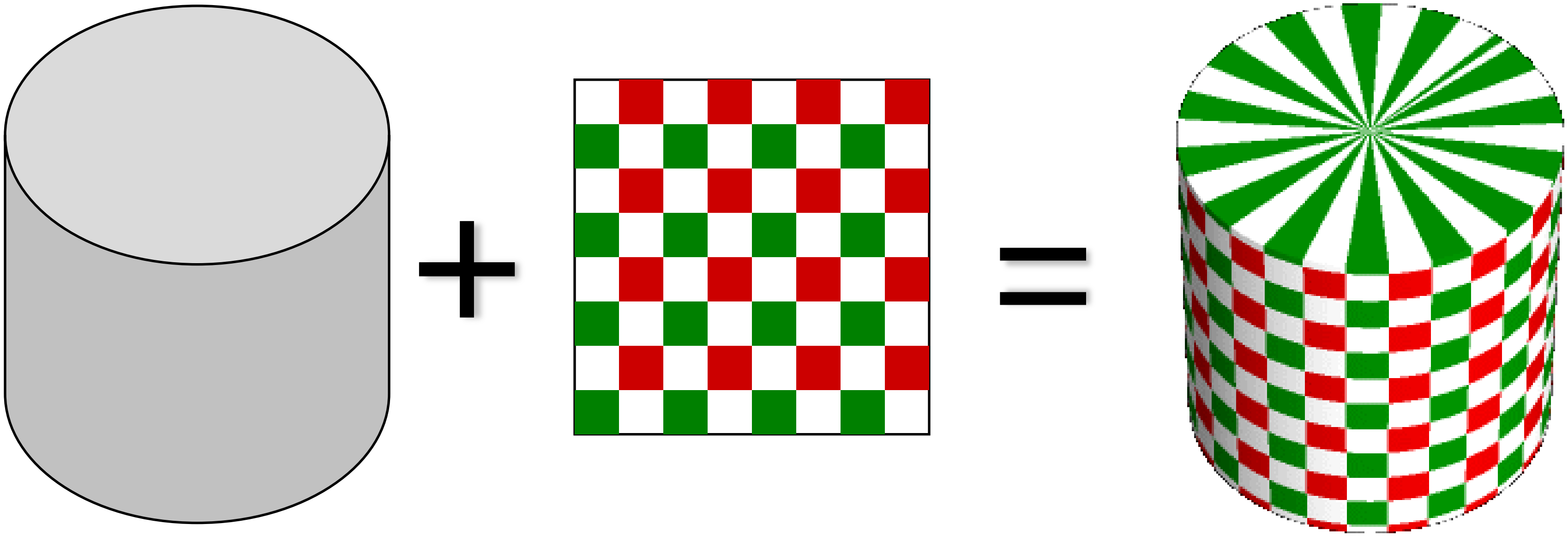
- At each texel keep sum of all values down & right
  - To compute sum of all values within a rectangle, simply subtract two entries
  - Better ability to capture very oblique projections
  - But, cannot store values in a single byte



Texture Mapping Overview

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

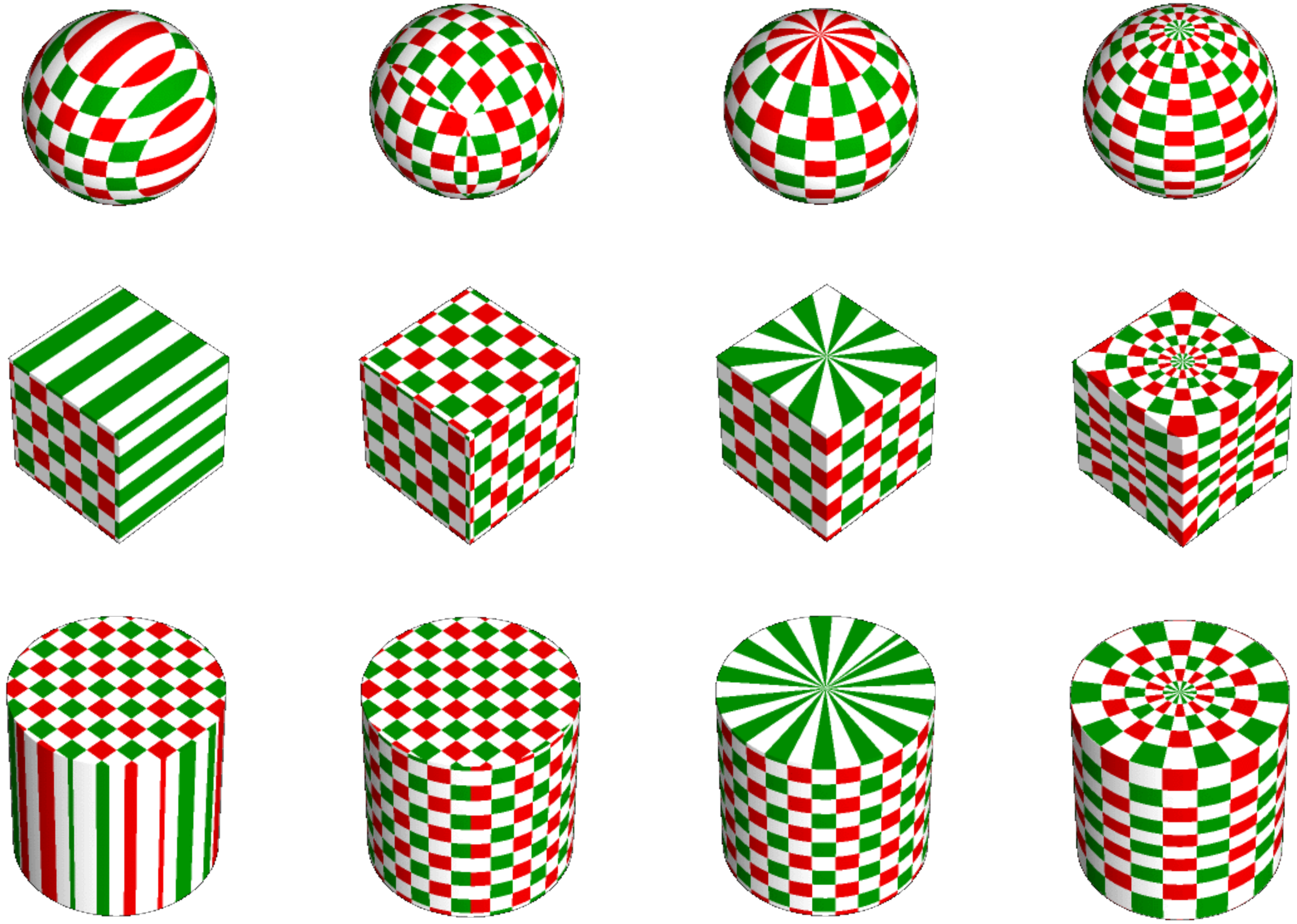
Parameterization



geometry image texture map

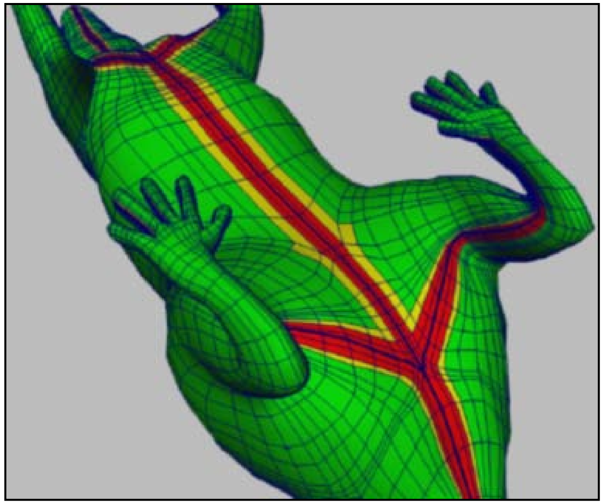
- Q: How do we decide *where* on the geometry each color from the image should go?

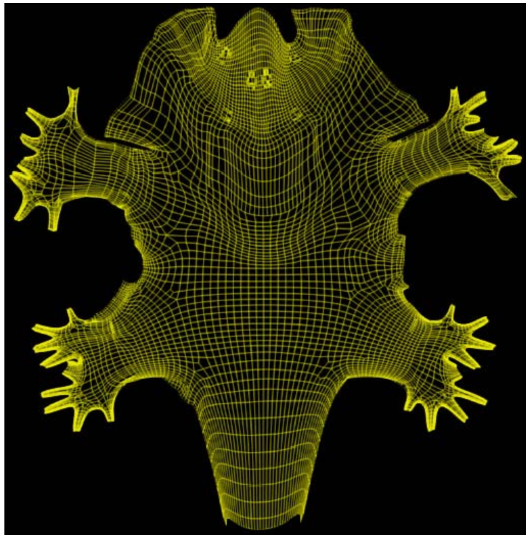
Option: Varieties of projections

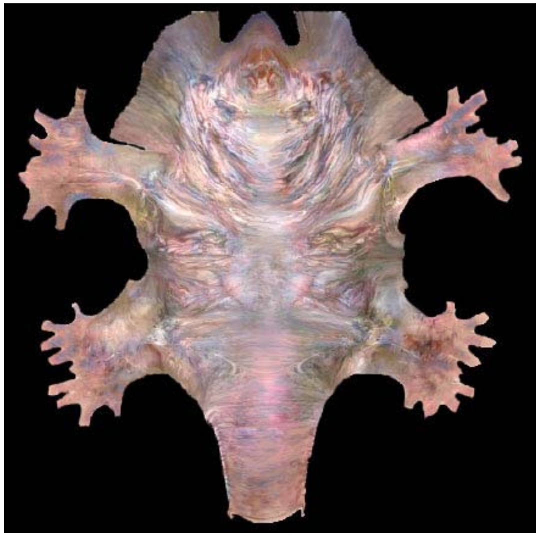


[Paul Bourke]

Option: unfold the surface



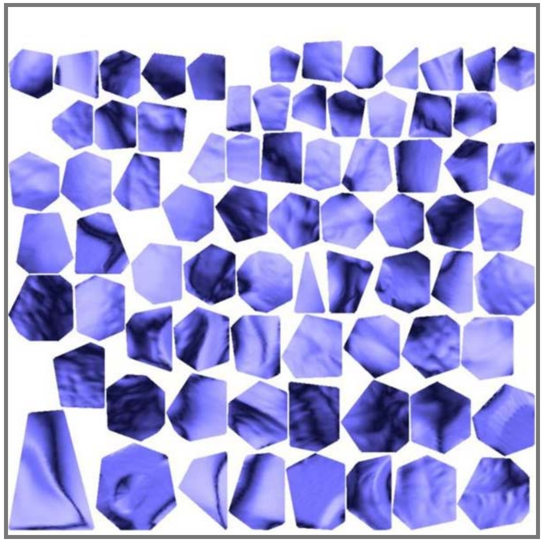




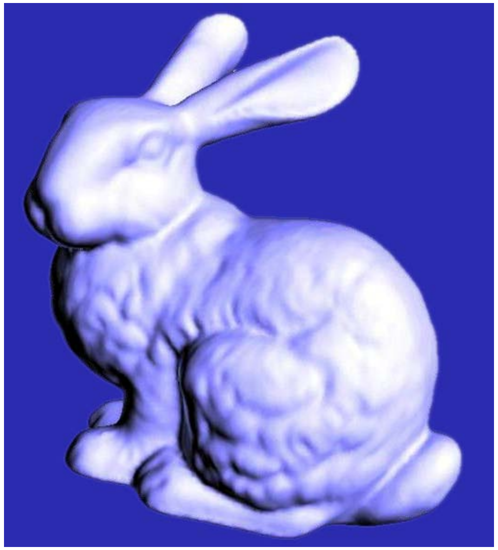
[Piponi2000]

Option: make an atlas

charts



atlas



surface

[Sander2001]

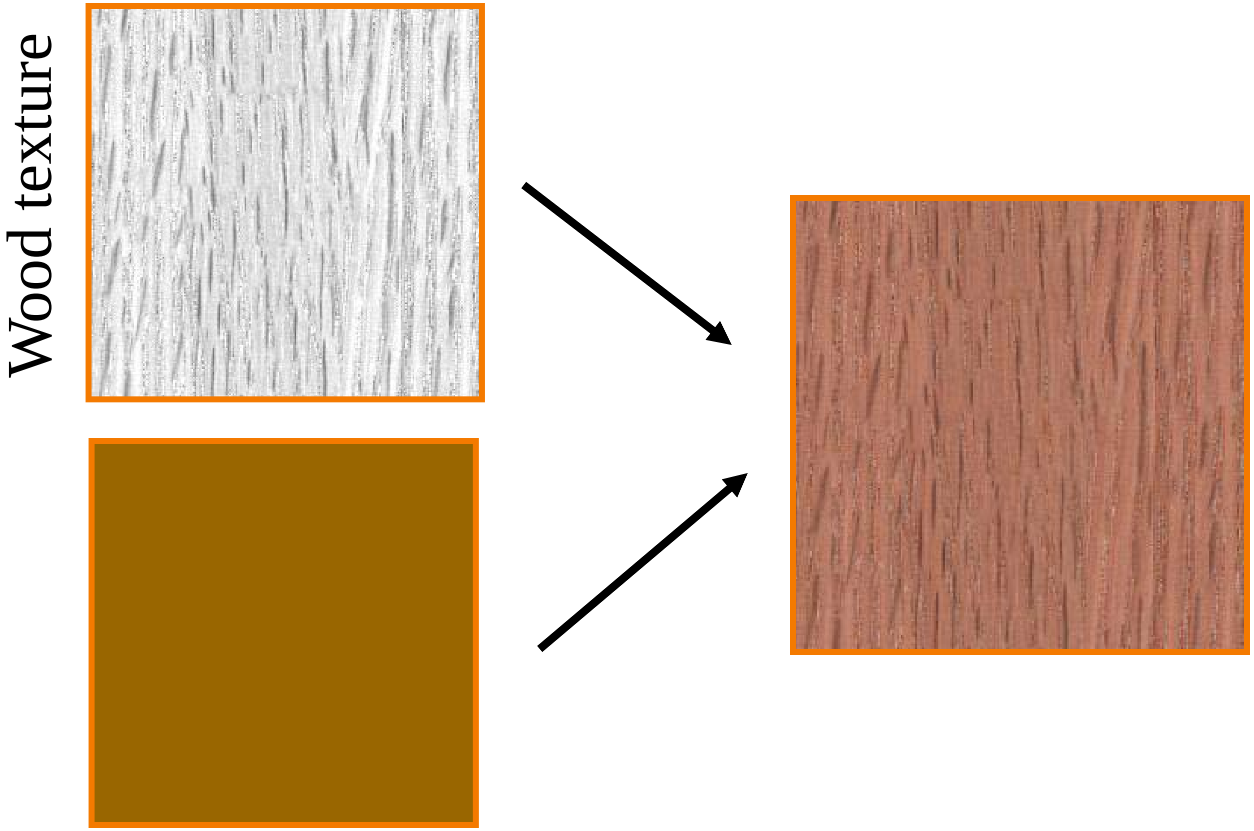
Texture Mapping Overview

- Texture mapping methods
  - Mapping
  - Filtering
  - Parameterization

- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering

Modulation textures

Map texture values to scale factor



Texture value

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

Illumination Mapping

Map texture values to surface material parameter

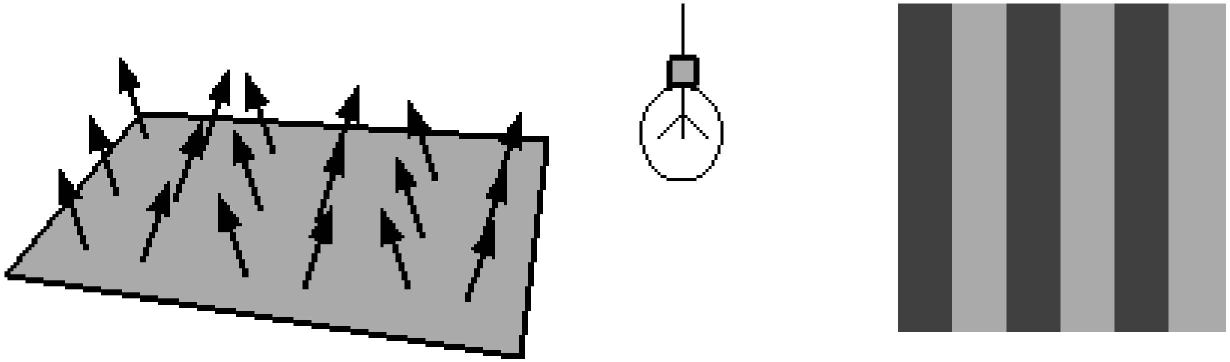
- K_A
- K_D
- K_S
- K_T
- n

$K_T = T(s,t)$

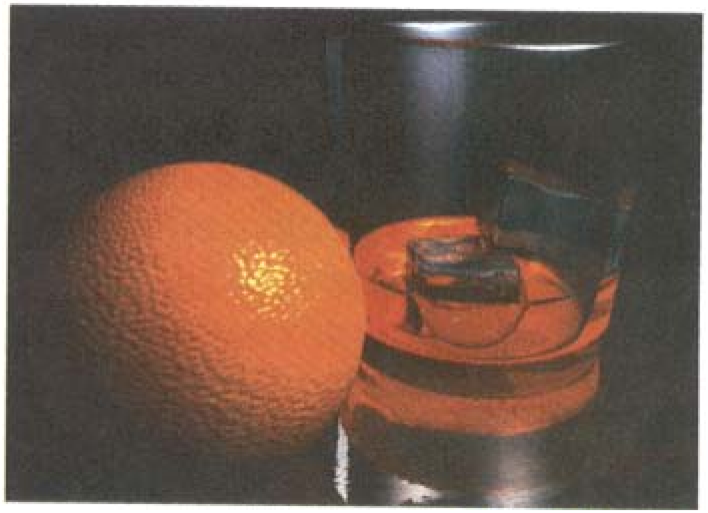
$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S$$

Bump Mapping

Texture values perturb surface normals



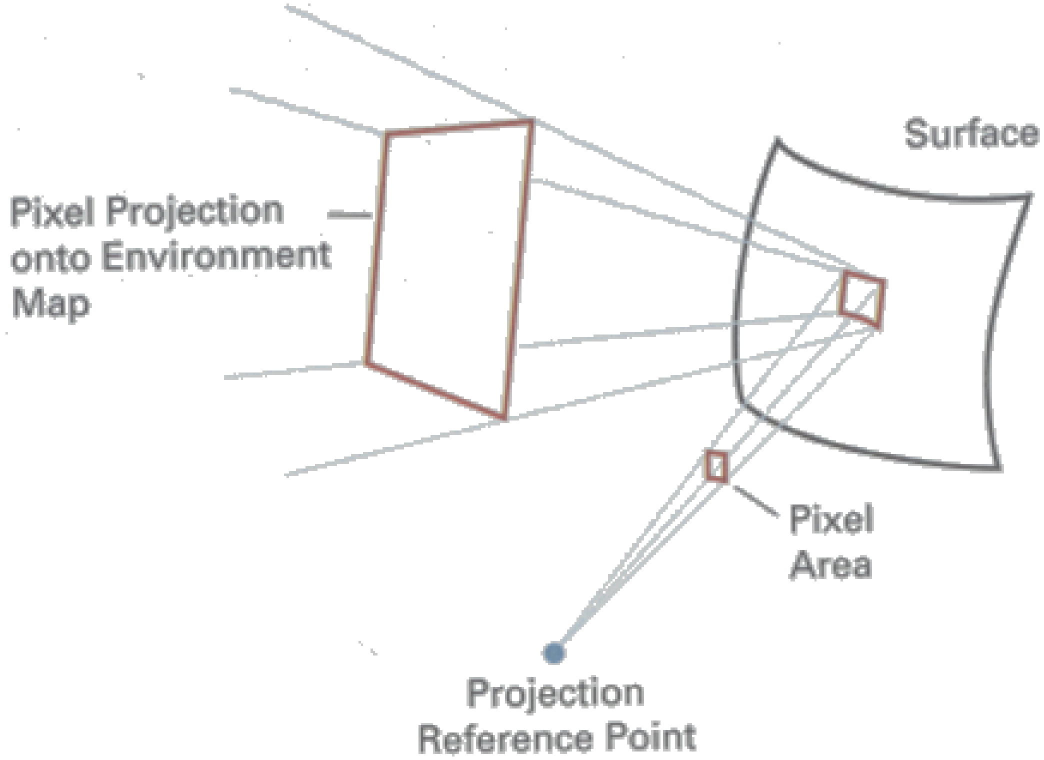
Bump Mapping



H&B Figure 14.100

Environment Mapping

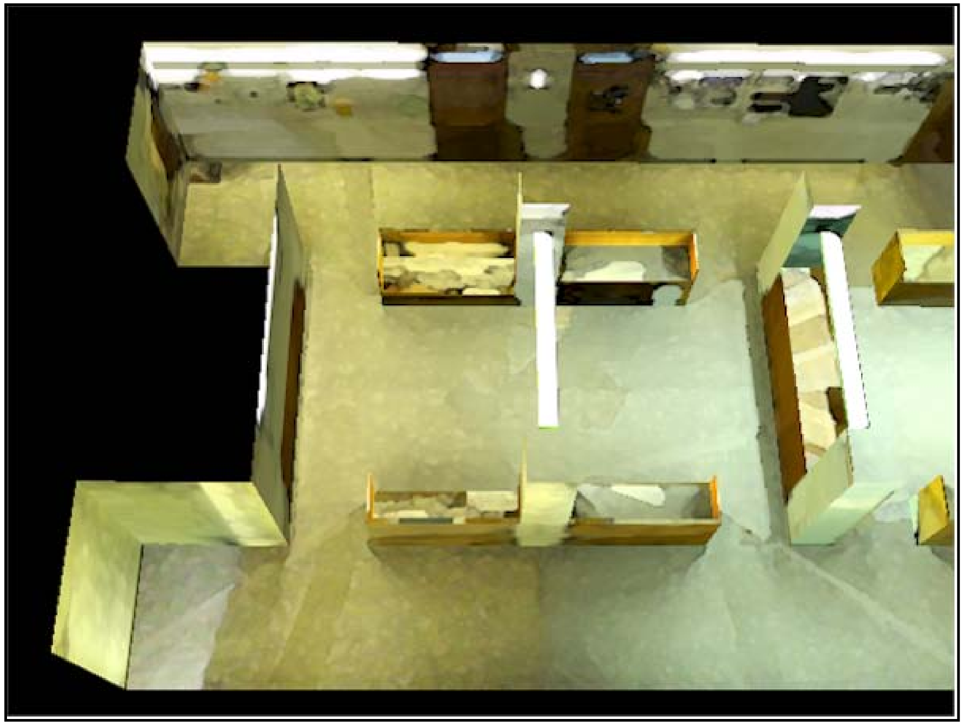
Texture values are reflected off surface patch



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

Texture Mapping Summary

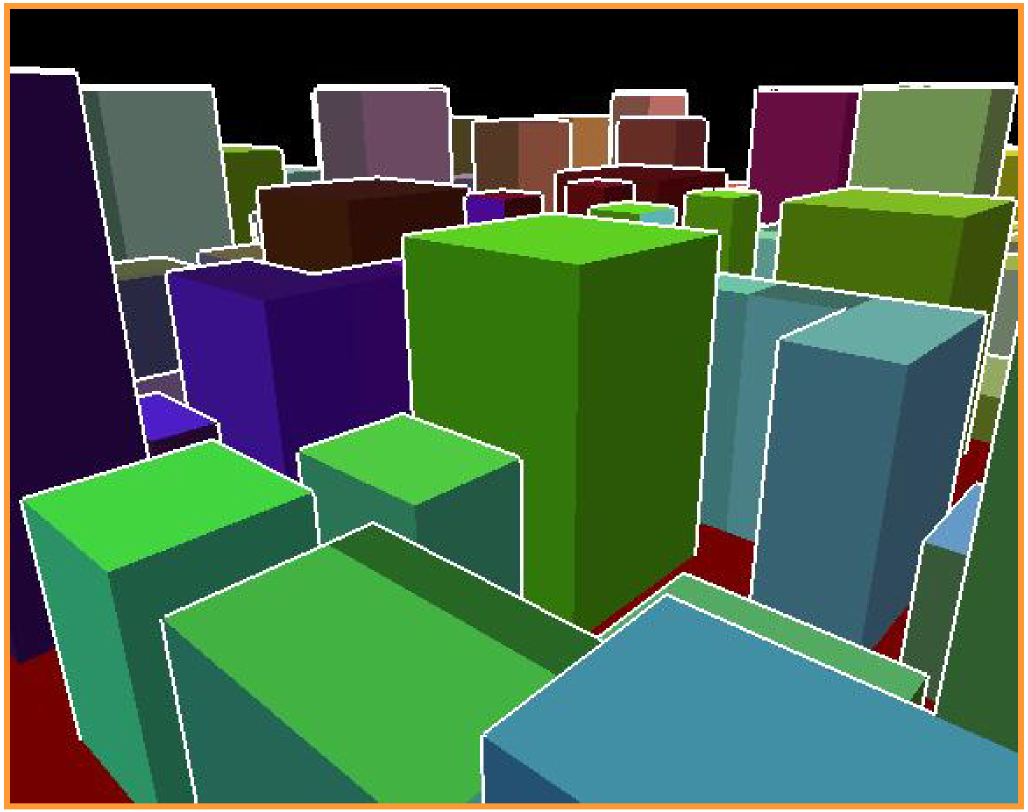
- Texture mapping methods
  - Parameterization
  - Mapping
  - Filtering
- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume textures

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

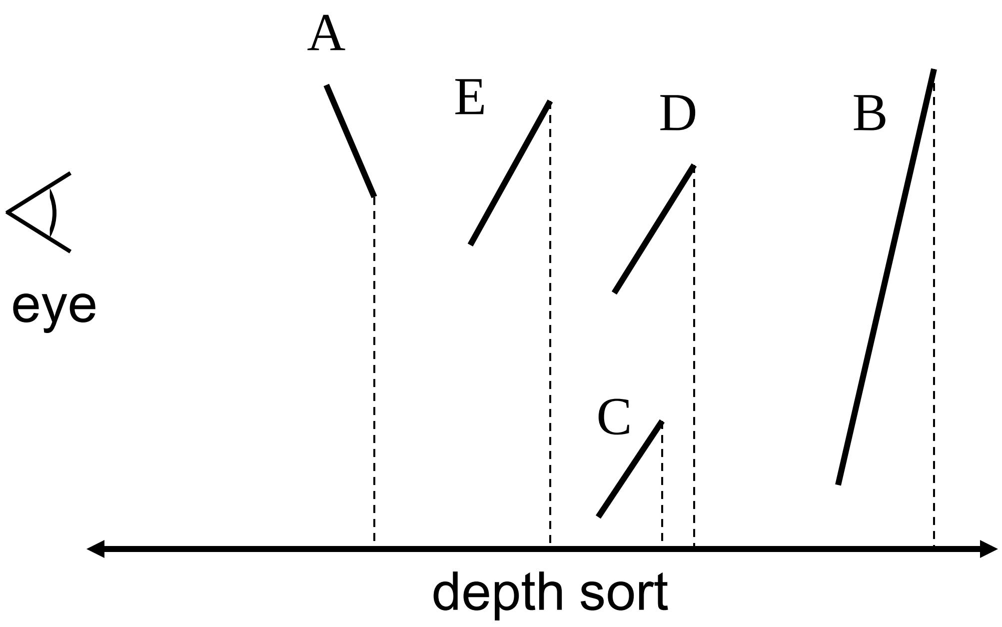
Visible Surface Determination

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

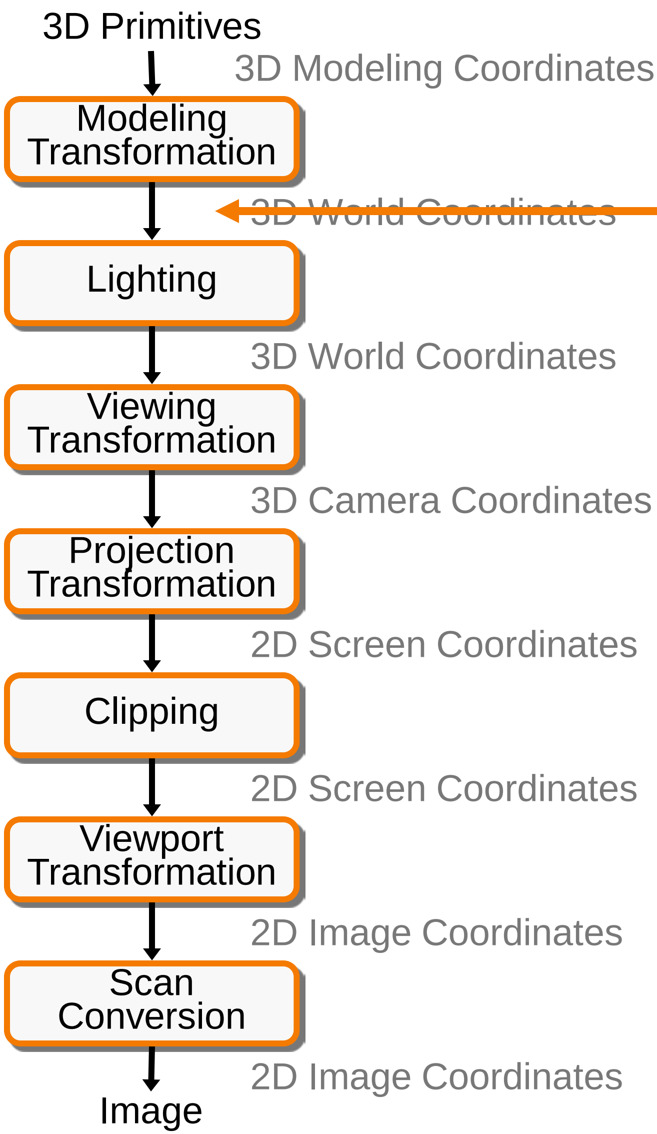


Depth sort

- "Painter's algorithm"
  - Sort surfaces in order of decreasing maximum depth
  - Scan convert surfaces in back-to-front order, overwriting pixels



3D Rendering Pipeline



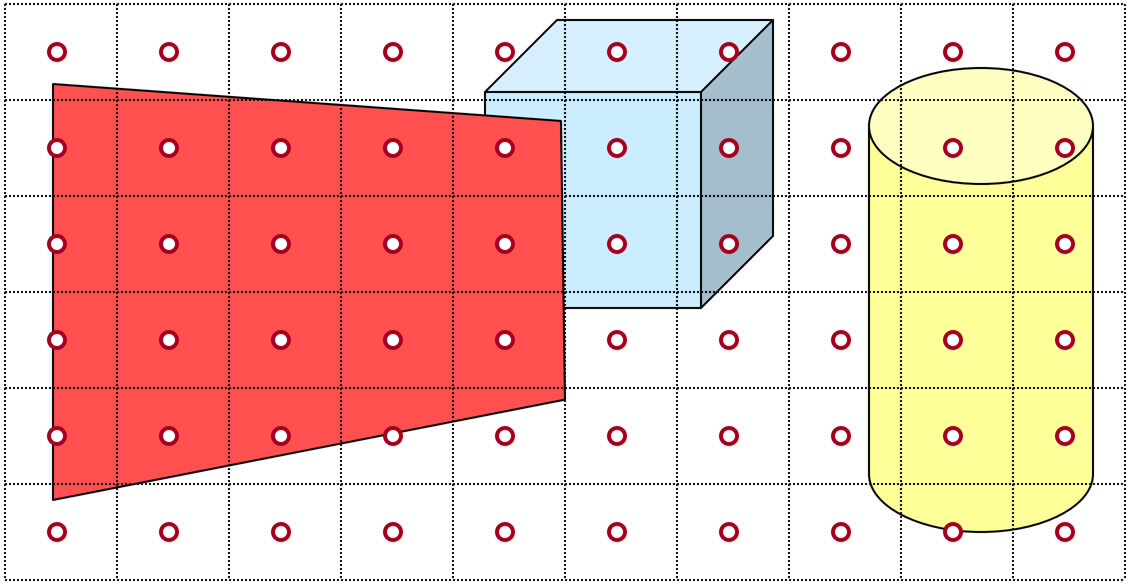
Depth sort

Depth sort comments

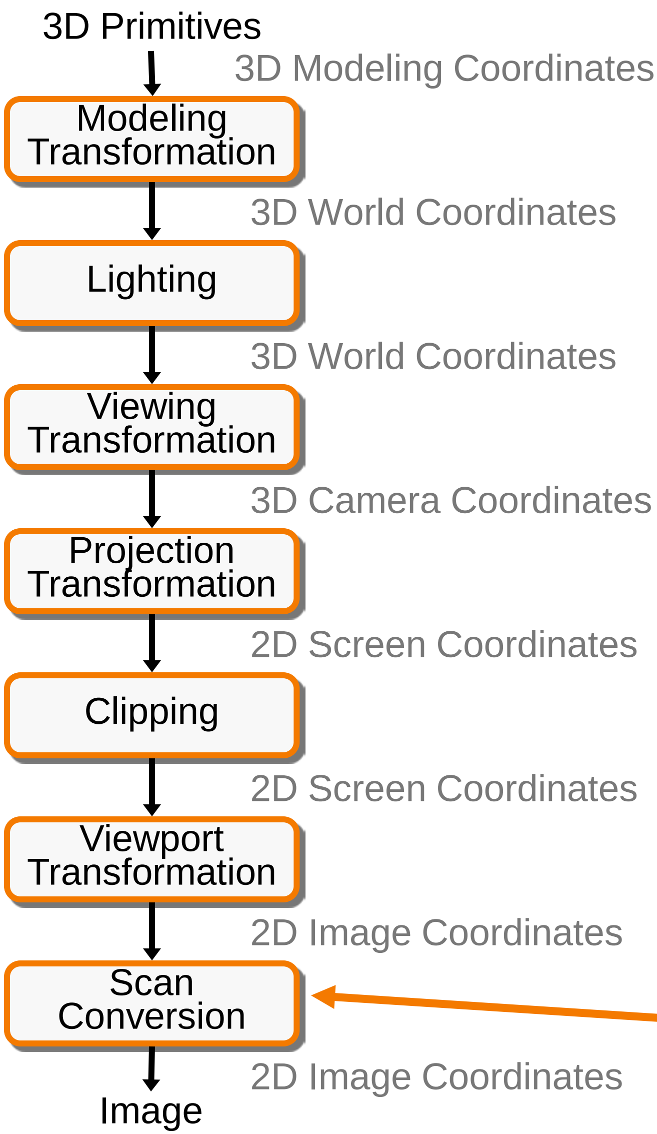
- O(n log n)
- Better with frame coherence?
- Implemented in software
- Render every polygon
- Often use BSP-tree or static list ordering

Z-Buffer

- Color & depth of closest object for every pixel
  - Update only pixels whose depth is closer than in buffer
  - Depths are interpolated from vertices, just like colors

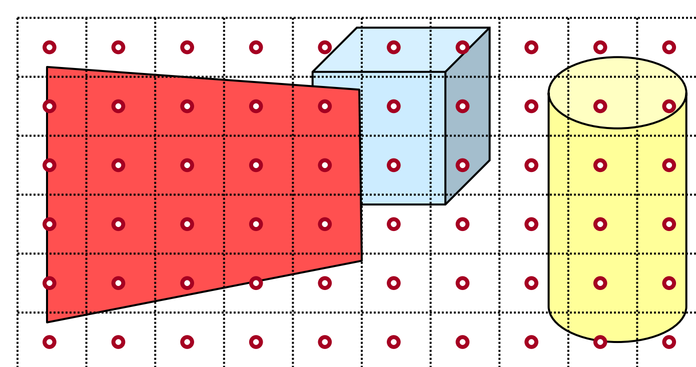


Z-Buffer

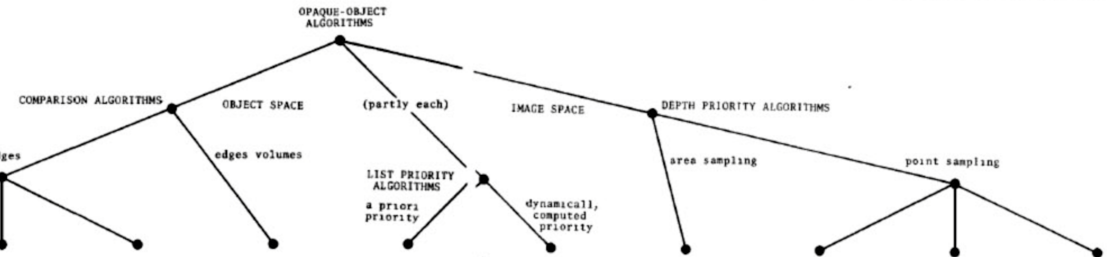


Z-buffer comments

- \+ 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



Hidden Surface Removal Algorithms



	APPEL 1967	GALIMBERTI, et al 1969	LOUTREL 1967	ROBERTS 1963	SCHUMACKER, et al 1969	NEWELL, et al 1972	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	None	TR,CF,NP	
COHERENCE	Promote visibility of a vertex to all edges at vertex	Promote visibility of a vertex to all edges at vertex	Promote visibility of a vertex to all edges at vertex		Frame coherence in depth No X coherence used	None used	Area coherence	Scanline X coherence	Scanline Depth Coherence	Scanline X Coherence
SORTING (1) What, what property (2) Method (3) Type (4) Result structure (5) Number per frame, number of objects (merge) Number of new entries per frame, length of list (search) Number of searches, length of list	Back Edge Cull 1) Edges separating back-facing planes 2) Dot product with normals & topology 3) Cull 4) List of edges, E_s 5) 1, 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) 1, 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) 1, 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) 1, E_t	Intra-Cluster Priority 1) Faces - visibility 2) Dot product with normals 3) Exhaustive search 4) Ordered table 5) 0, (off-line)	Z Sort 1) Faces, max 2 2) Comparison of max points 3) n logm 4) Ordered table 5) 1, F_r	Z Sort (Opt.) 1) Faces, max Z 2) Comparison of max points 3) n log m 4) Ordered table 5) 1, F_r	Y Sort 1) Edges, min Y 2) Comparison 3) Bucket 4) Table of Lists 5) 1, E_r	Y Sort 1) Polygons, Y endpoints 2) Comparison 3) 2 bucket 4) Table of lists 5) 1, F_r	Y Sort 1) Edges, Min Y 2) Comparison 3) Bucket 4) Table of lists 5) 1, E_r
	Contour Edge Cull 1) Edges separating front & back faces 2) Dot product with normals & topology 3) Cull 4) List, E_c 5) 1, E_t	(Omitted)	(Omitted)	Clipping Cull 1) Intersect edge with visible volume 2) 3) Cull 4) E_s 5) $1_?^s$ E_s	Inter-Cluster Priority 1) Clusters 2) Dot product with separating planes 3) Prefix scan binary tree 4) ordered table 5) 1, C_t	Newell Special 1) Faces, pairwise visibility 2) Depth, bounding boxes, separation 3) Bubble, splitting 4) Ordered table 5) 1,F_r+split faces	Warnock Special 1) Faces with window 2) Depth, mini-max in X and Y, sum of angles 3) Radix 4 subdivision with overlap 4) Stacks of unordered tables 5) L_v, F_r/factor 1	X Merge 1) Edges, X value 2) Comparison 3) Merge (ordred) 4) 2-way linked list 5) E_r, S_ℓ	X Sort 1) Edges, X value 2) Comparison 3) 2 bucket 4) Table of lists 5) n, S_ℓ	X Merge 1) Edges, X value 2) Comparison 3) Merge (ordered) 4) Linked list 5) E_r, $2S_\ell$ (edges)
	Initial Visibility 1) Ray to vertex against all faces 2) Depth, Surroundedness 3) Exhaustive search 4) Quantitative visibility of vertex 5) # Objects, F_r	Initial Visibility 1) Ray to vertex against all faces 2) Depth, surroundedness 3) Exhaustive search 4) Quantitative visibility of vertex 5) #objects, F_r	Initial Visibility 1) Ray to vertex against all faces 2) Betweenness, surroundedness 3) Exhaustive search 4) Quantitative visibility of vertex 5) #objects, F_r	Edge/Volume Test 1) Edges, visibility relative to volumes 2) Linear Programming 3) Mini-max sort 4) Answer 5) E_s+ split edges, #objects	Back-Face Cull 1) Faces 2) Dot product with face normal 3) Cull 4) Smaller ordered table 5) 1, F_t	Y Sort 1) Face segment by Y range 2) Y intercept 3) Bucket 4) None 5) F_r + split faces H_f^r	Depth Search 1) Surrounder faces 2) 4-corner compare 3) Exhaustive 4) Answer/failure 5) L_v, F_r/factor 2	X Sort 1) Segments, X left 2) Comparison 3) Bubble 4) 2-way linked list 5) n, S_ℓ	X Priority Search 1) Edges, X value 2) Comparison 3) Priority search 4) Active segment list 5) n, m	X Sort 1) Edges, X value 2) Comparison 3) Bubble 4) 1-way linked list 5) N, $2S_\ell$ (edges)
	Edge Intersection 1) Intersect one E_s with all E_c 2) Penetration with sweep triangle 3) Cull (unordered) 4) Intersection list 5) E_s, E_c	Edge Intersection 1) Intersect one E_s with all E_c 2) Intersect in picture plane, depth 3) Cull (unordered) 4) Intersection list 5) E_s, E_s - 1	Edge Intersection 1) Intersect one E_s with all E_c 2) Intersect in picture plane, depth 3) Cull (unordered) 4) Intersection list 5) E_s, E_s - 1		Y Cull 1) Faces by Y extent 2) Mini-max on X intercepts 3) Cull (unordered) 4) X intercepts of relevant segments 5) n, E_s	X Merge 1) Segments, X intercept 2) Comparison 3) Ordered merge 4) Ordered list 5) S_r, $S_v/2$	TV Sort (Opt.) Sort windows into scan-line order if needed	Span Cull 1) Segments, overlap with sample span 2) Double comparison 3) Cull ordered list 4) Active list 5) n^*S_v * f (>1), S_ℓ	Z Search 1) Segments, depth 2) Linear equations and comparison 3) Search (unordered) 4) Visible segment 5) n^*2S_ℓ, D_c (Omitted if X priorities same as last time)	Z Search 1) Segments, depth 2) Linear equations and comparison 3) Search of unordered active list 4) Visible segment 5) n^*2S_ℓ, D_c
	Sort Along Edge 1) Intersections on edge, ordering 2) Comparison 3) Bubble 4) Answer 5) E_s, X_v/E_s (Omit if well hidden)	Sort Along Edge 1) Intersections on edge, ordering 2) 3) 4) Answer 5) E_s, X_v/E_s (must be done)	Sort Along Edge 1) Intersections on edge, ordering 2) 3) 4) Answer 5) E_s, X_v/E_s (Omit if well hidden)		X Sort 1) Segments 2) Counters 3) Hardware 4) Segments at this X 5) nm, S_ℓ			Z Search 1) Segments, Z 2) Depth by logarithmic search 3) Search (unordered) 4) Visible segment 5) n^*S_v*f(>1), D_c		
					Priority Search 1) Segments, priority 2) Logic network 3) Logic network 4) visible segment 5) nm, S_ℓ					

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

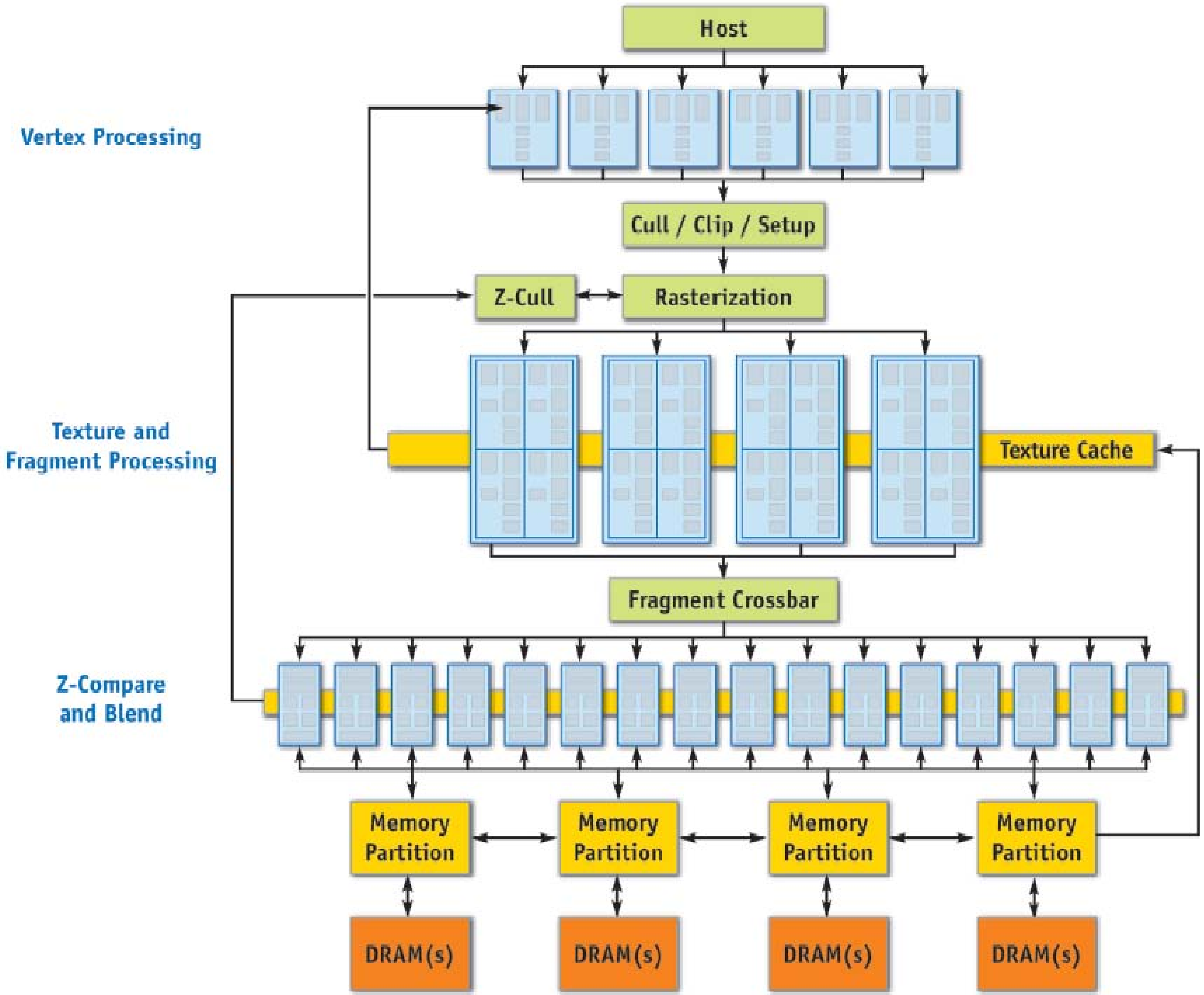
[Sutherland '74]

Rasterization Summary

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

This is all in hardware

GPU Architecture

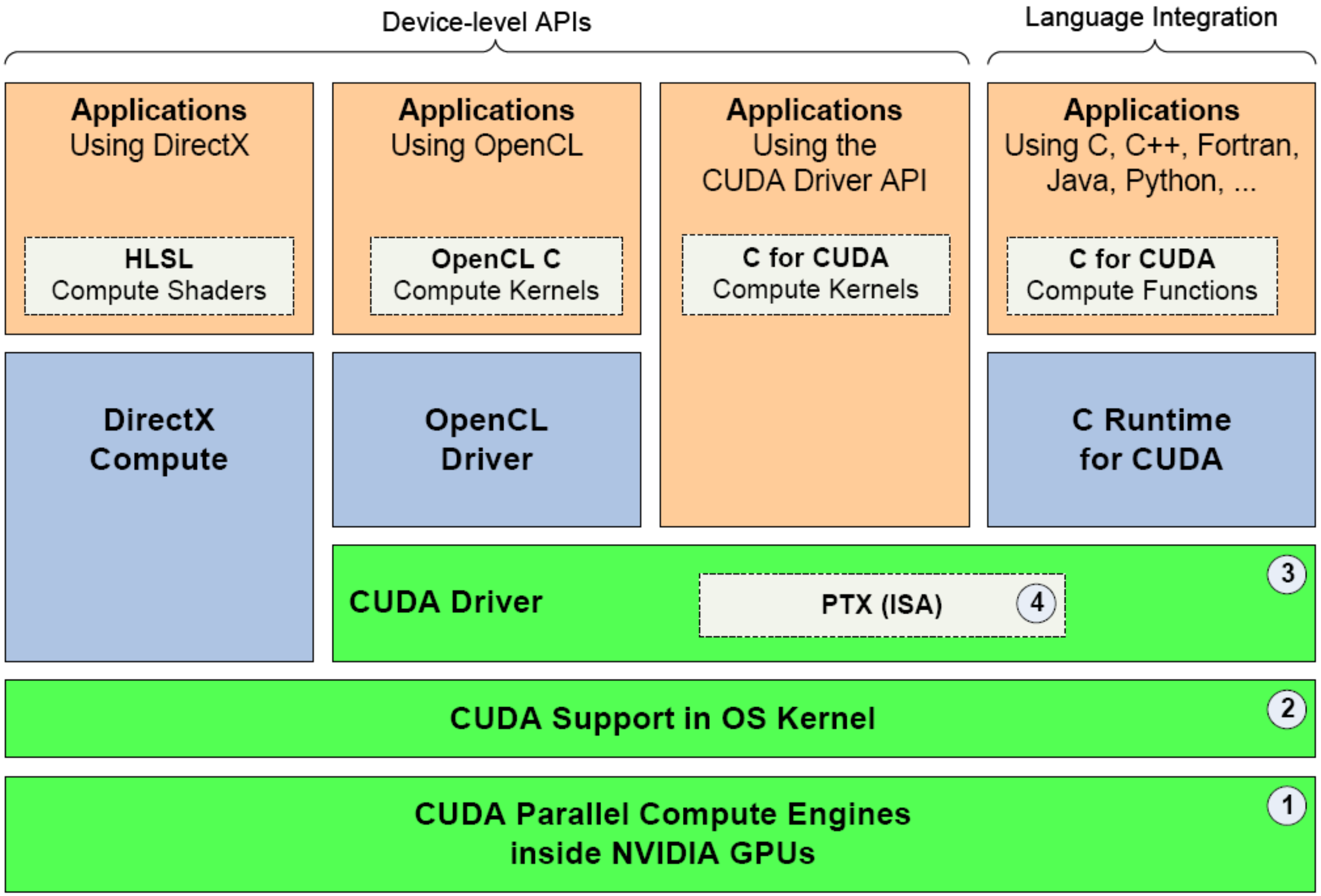


GeForce 6 Series Architecture

GPU Gems 2, NVIDIA

Actually …

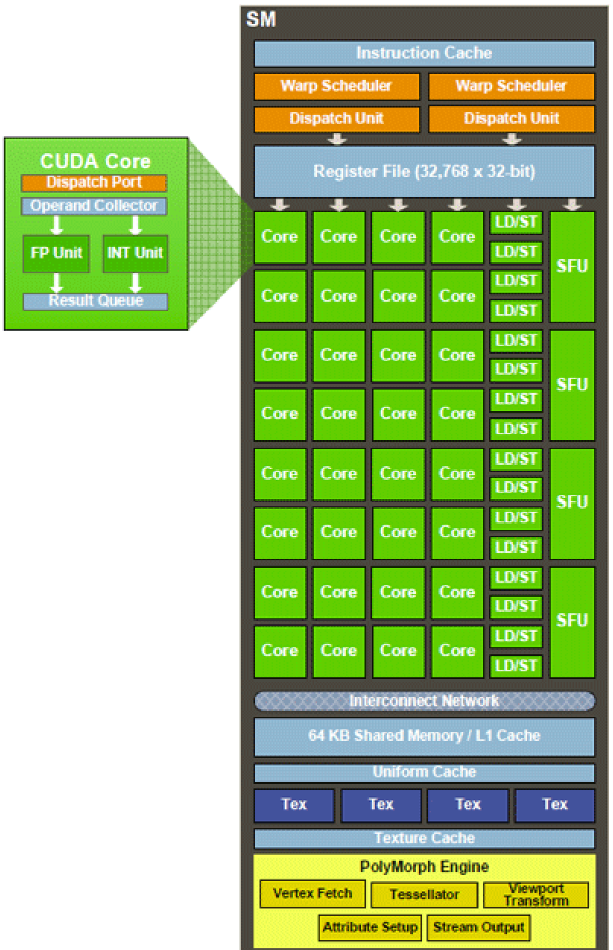
- Graphics hardware is programmable



www.nvidia.com/cuda

Trend …

- GPU is general-purpose parallel computer



www.nvidia.com/cuda