Texture Mapping & Hidden Surface Removal

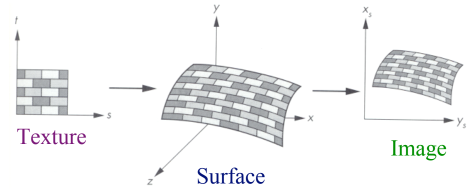
Adam Finkelstein

Princeton University

COS 426, Spring 2005

Textures

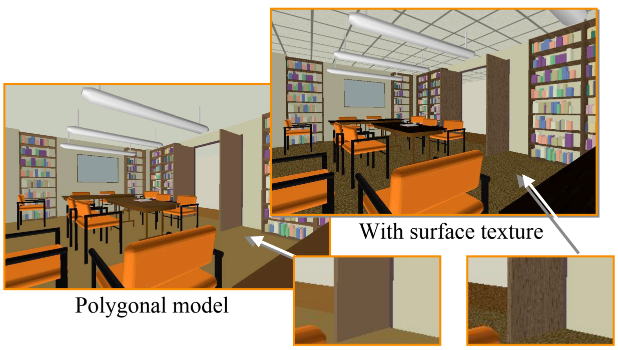
- Describe color variation in interior of 3D polygon
  - When scan converting a polygon, vary pixel colors according to values fetched from a texture



Angel Figure 9.3

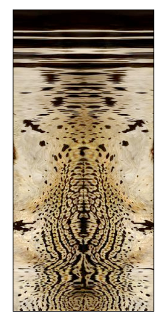
Surface Textures

- Add visual detail to surfaces of 3D objects



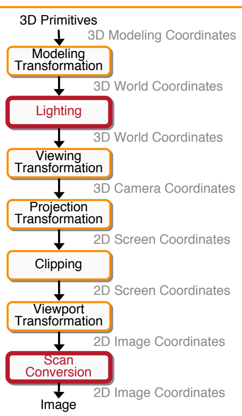
Surface Textures

- Add visual detail to surfaces of 3D objects



[Daren Horley]

3D Rendering Pipeline (for direct illumination)

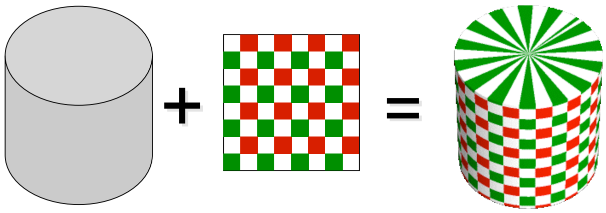


Texture mapping

Texture Mapping Overview

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

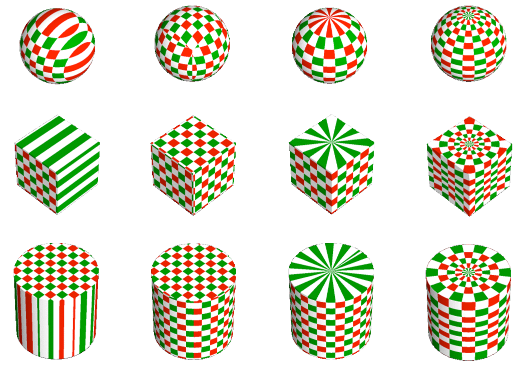
Parameterization



geometry image texture map

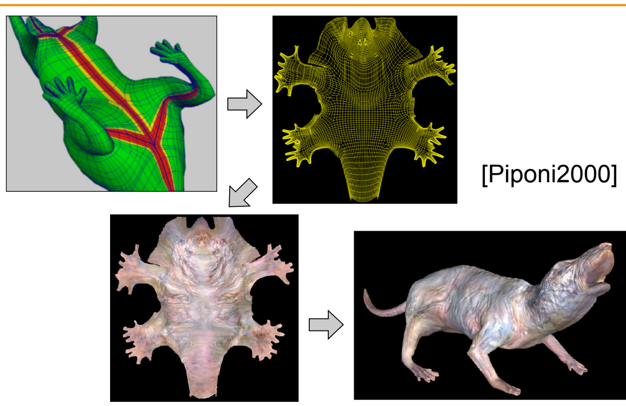
- Q: How do we decide *where* on the geometry each color from the image should go?

Option: Varieties of projections



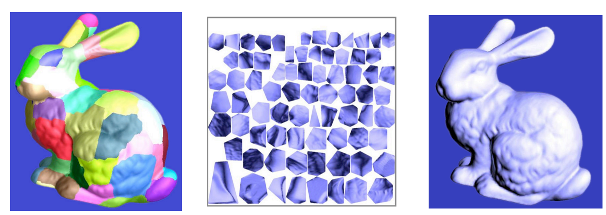
[Paul Bourke]

Option: unfold the surface



[Piponi2000]

Option: make an atlas



charts atlas surface

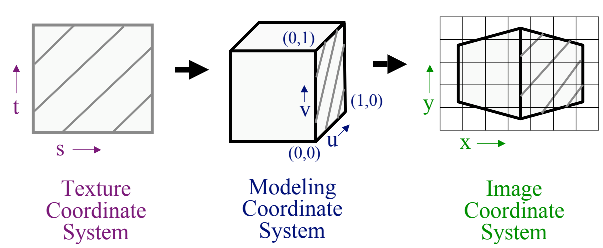
[Sander2001]

Texture Mapping Overview

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

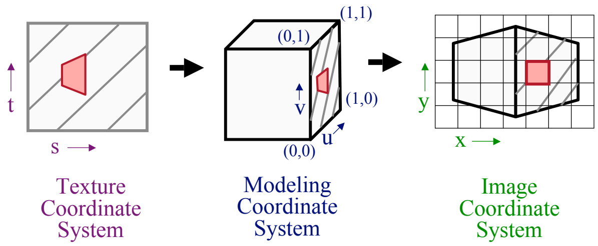
Texture Mapping

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



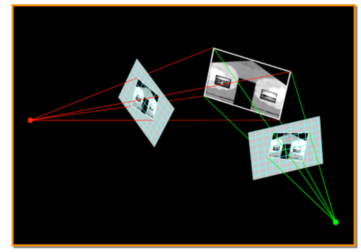
Texture Mapping

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



Texture Mapping

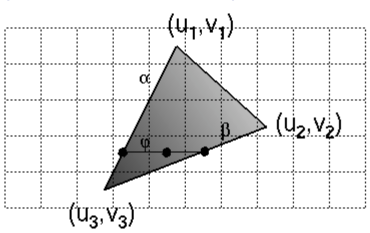
- Texture mapping is a 2D projective transformation
  - o texture coordinate system: (t,s) to
  - o image coordinate system (x,y)



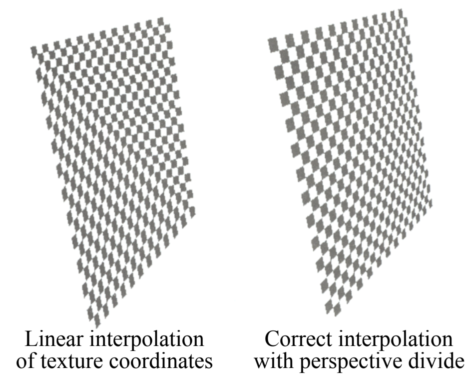
[Allison Klein]

Texture Mapping

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



Texture Mapping



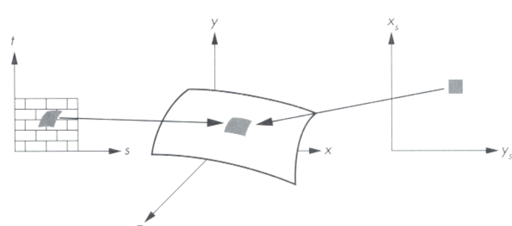
Hill Figure 8.42

Overview

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

Texture Filtering

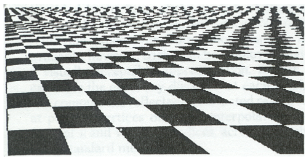
- Must sample texture to determine color at each pixel in image



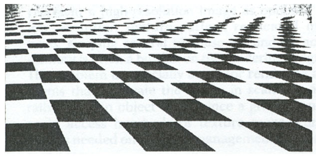
Angel Figure 9.4

Texture Filtering

- Aliasing is a problem



Point sampling

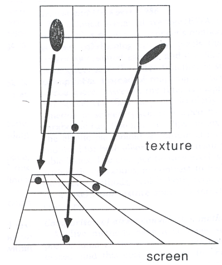


Area filtering

Angel Figure 9.5

Texture Filtering

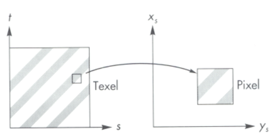
- Ideally, use elliptically shaped convolution filters



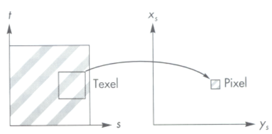
In practice, use rectangles

Texture Filtering

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



Magnification

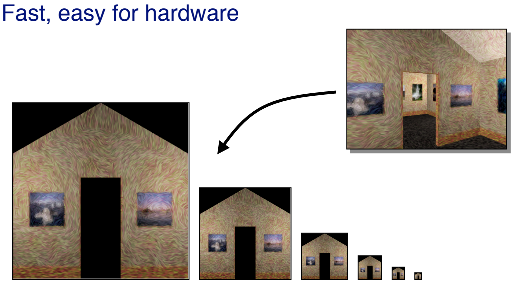


Minification

Angel Figure 9.14

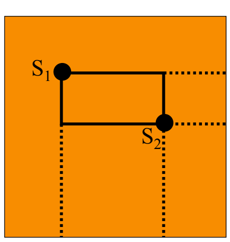
Mip Maps

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



Summed-area tables

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

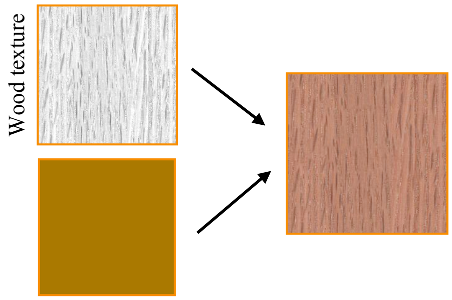


Overview

- Texture mapping methods
  - o Parameterization
  - o Mapping
  - o Filtering
- Texture mapping applications
  - o Modulation textures
  - o Illumination mapping
  - o Bump mapping
  - o Environment mapping
  - o 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

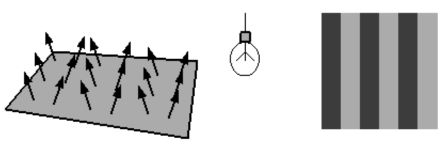
- K_A
- K_D
- K_S
- K_T
- n

$K_T = T(s,t)$

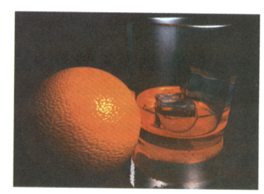
$$I = I_E + K_A I_A + \sum_L (K_D(N \bullet L) + K_S(V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S$$

Bump Mapping

Texture values perturb surface normals



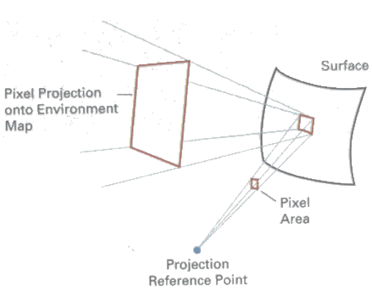
Bump Mapping



H&B Figure 14.100

Environment Mapping

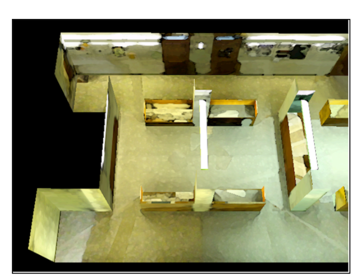
Texture values are reflected off surface patch



H&B Figure 14.93

Image-Based Rendering

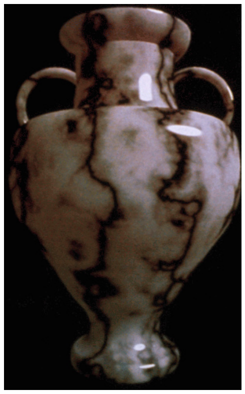
Map photographic textures to provide details for coarsely detailed polygonal model



Solid textures

Texture values indexed by 3D location (x,y,z)

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

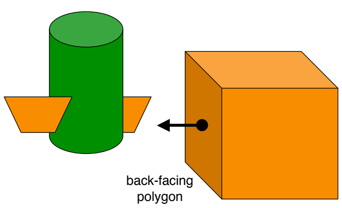


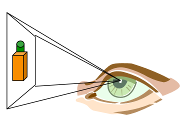
Texture Mapping Summary

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

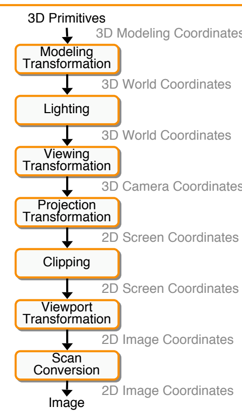
Hidden Surface Removal (HSR)

- Surfaces may be back-facing.
- Surfaces may be occluded.
- Surfaces may overlap in the image plane.
- Surfaces may intersect.



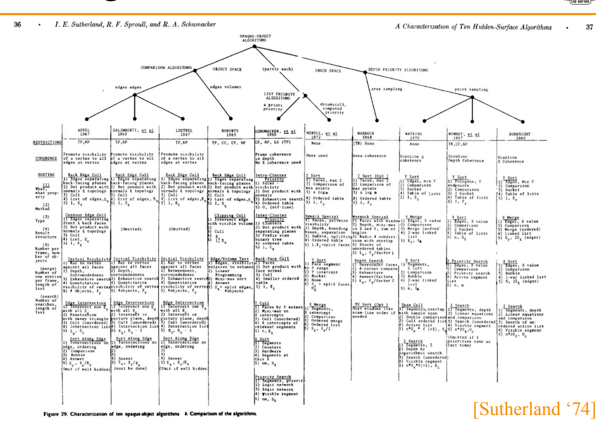


3D Rendering Pipeline



Somewhere in here we have to decide which objects are visible, and which objects are hidden.

HSR Algorithms

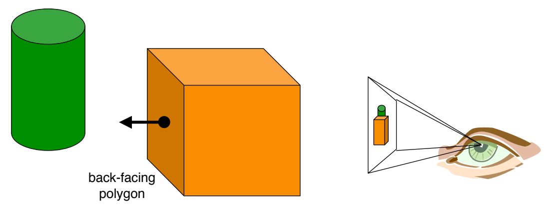


[Sutherland '74]

Hidden Surface Removal Algorithms

- Object space
  - o Back-face detection
  - o Depth sort
- Screen space
  - o Ray casting
  - o Scan-line
  - o Z-buffer
  - o Area subdivision

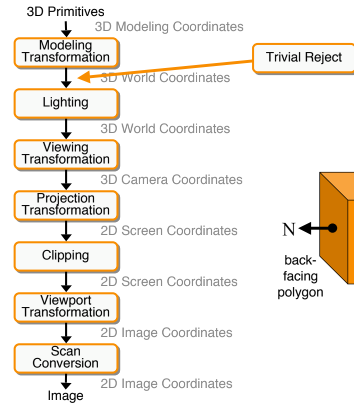
Back-face detection



Q: How do we test for back-facing polygons?

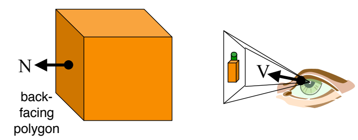
A: Dot product of the normal and view directions.

3D Rendering Pipeline



A polygon is backfacing if

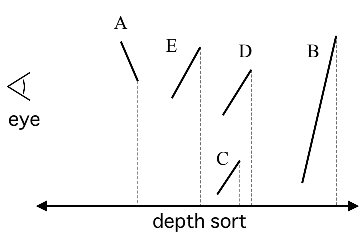
$$V \cdot N > 0$$

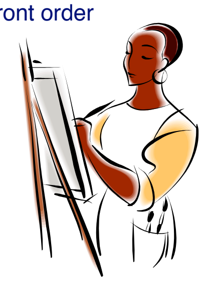


Depth sort

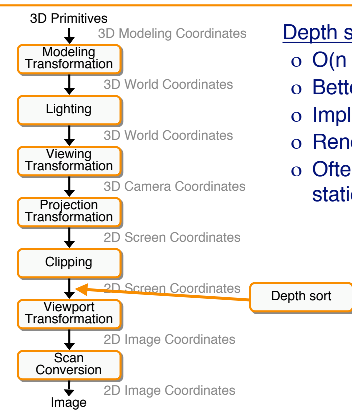
"Painter's algorithm"

- Sort surfaces in order of decreasing maximum depth
- Scan convert surfaces in back-to-front order





3D Rendering Pipeline

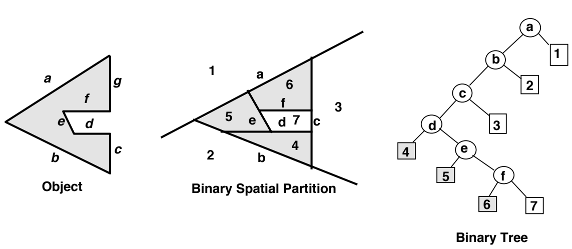


Depth sort comments

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

BSP Tree

- Binary space partition with solid cells labeled
  - Constructed from polygonal representations
  - Provides linear-time depth sort for arbitrary view



(We'll come back to this…)

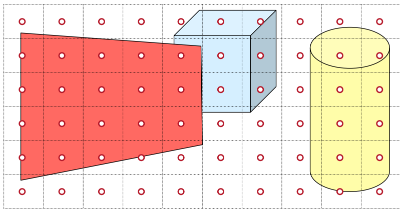
Naylor

Hidden Surface Removal Algorithms

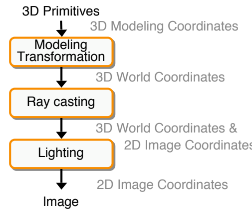
- Object space
  - Back-face detection
  - Depth sort
- Screen space
  - Ray casting
  - Scan-line
  - Z-buffer
  - Area subdivision

Ray Casting

- Fire a ray for every pixel
  - If ray intersects multiple objects, take the closest



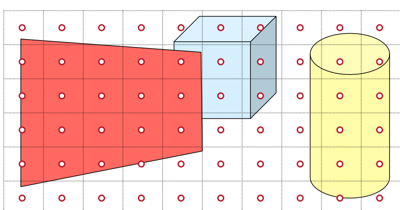
Ray Casting Pipeline



Ray casting comments
- O(p log n) for p pixels
- May (or not) use pixel coherence
- Simple, but generally not used

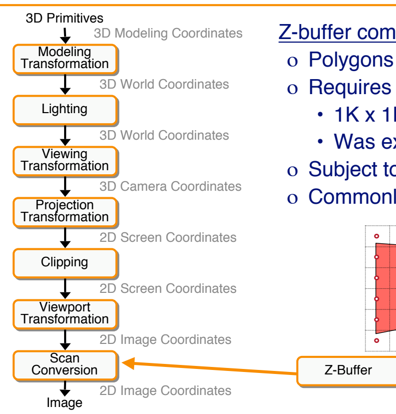
Z-Buffer

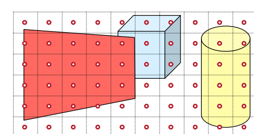
- Color & depth of closest object for every pixel
  - Update only pixels whose depth is closer than in buffer
  - Depths are interpolated from vertices, just like colors



3D Rendering Pipeline

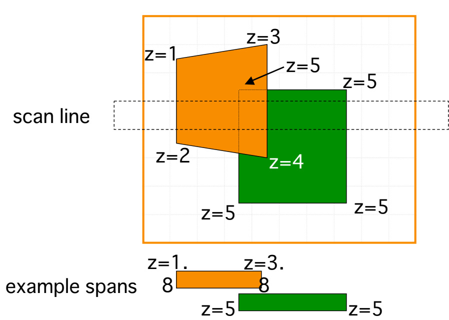
Z-buffer comments
- Polygons rasterized in any order
- Requires lots of memory
  - 1K x 1K x 24bits
  - Was expensive, cheap now
- Subject to aliasing (A-buffer)
- Commonly in hardware



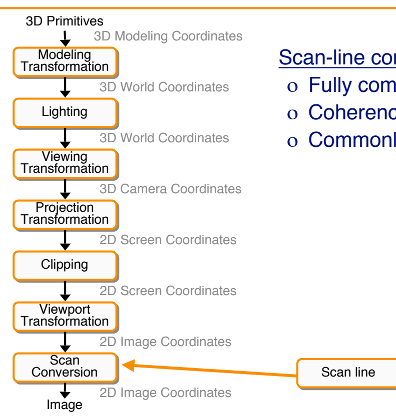


Scan-Line Algorithm

- For each scan line, construct spans
  - Sort by depth



3D Rendering Pipeline

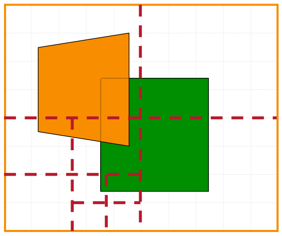


Scan-line comments
- Fully compute only visible pixels
- Coherence among along scans
- Commonly in software

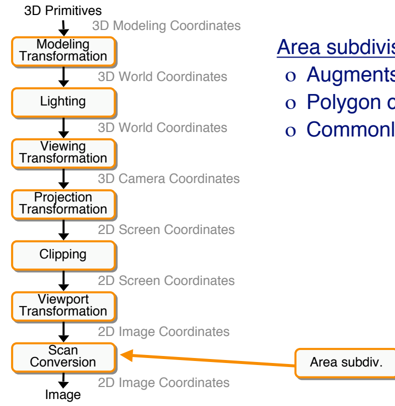
Area Subdivision

Warnock's algorithm

- Fill area if:
  - » All surfaces are outside area, or
  - » Only one surface intersects area, or
  - » One surface occludes other surfaces in area
- Otherwise, subdivide



3D Rendering Pipeline



Area subdivision comments

- Augments scan conversion
- Polygon coherence
- Commonly in software

Summary

- Texture Mapping
  - Add detail during scan conversion
- Hidden surface removal
  - Find visible surfaces