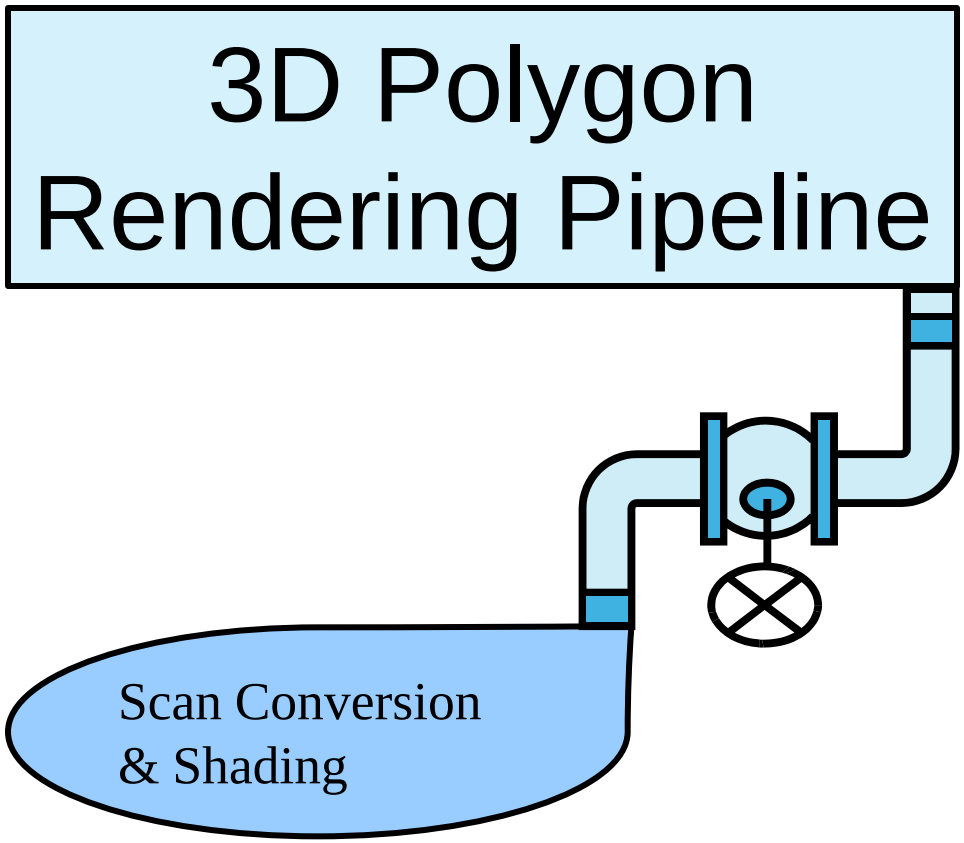
קורס גרפיקה ממוחשבת

2009 סמסטר ב'

ליאור שפירא

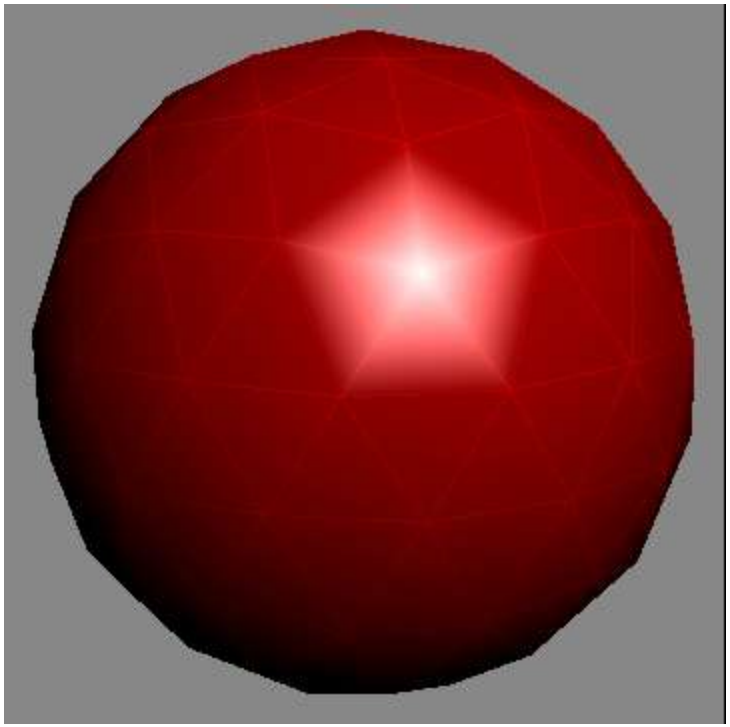


שיעור 8 חלק ראשון
השלמות...

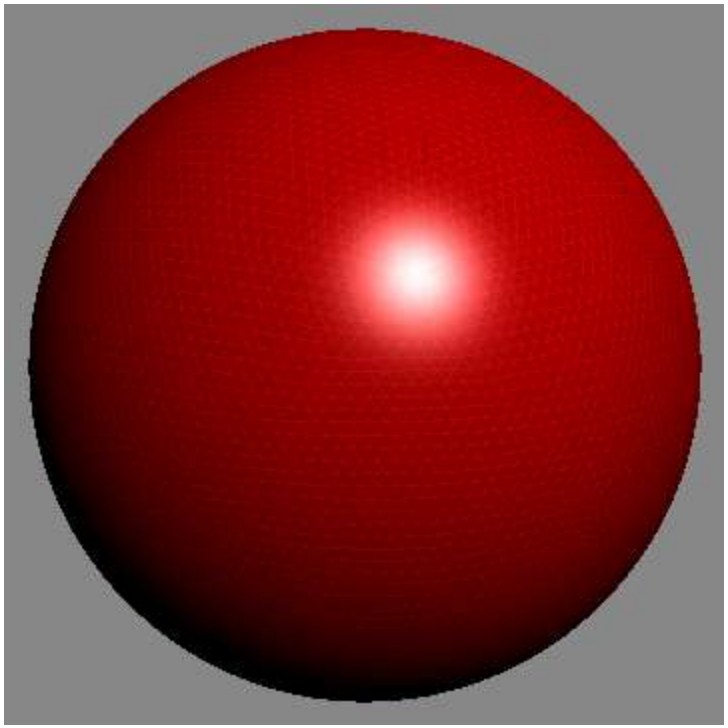
Thomas Funkhouser
Princeton University
C0S 426, Fall 1999

Gouraud Shading

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



Poor behavior of specular light



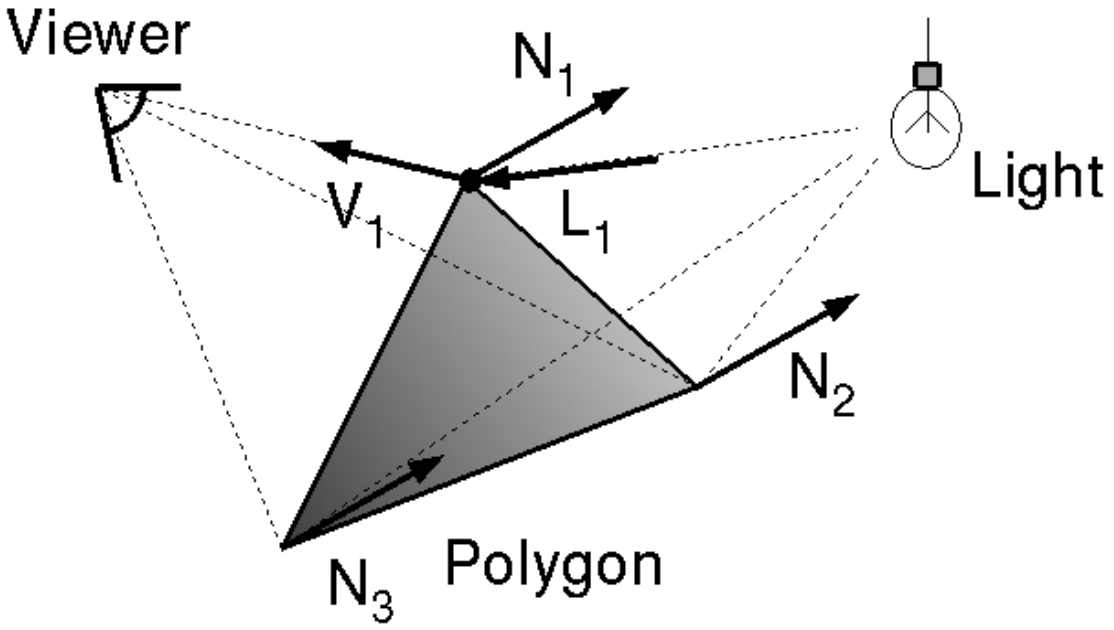
Same sphere – high polygon count

Polygon Shading Algorithms

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

Phong Shading

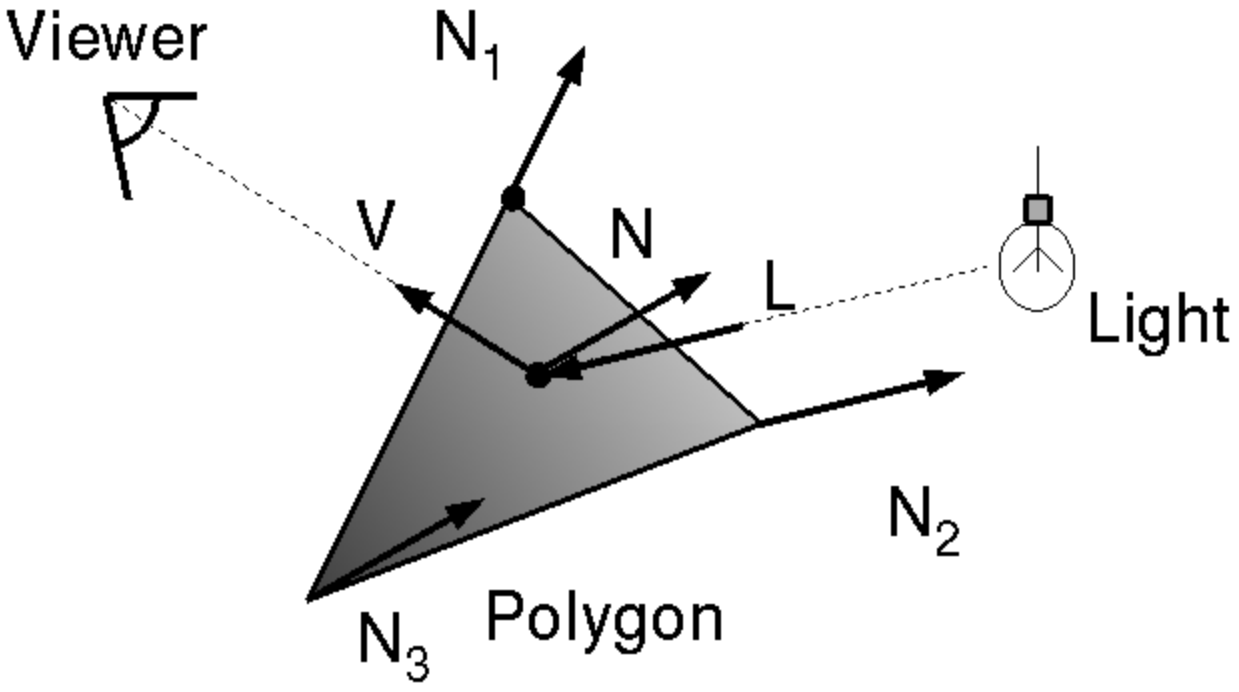
- What if polygonal mesh is too coarse to capture illumination effects in polygon interiors?



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

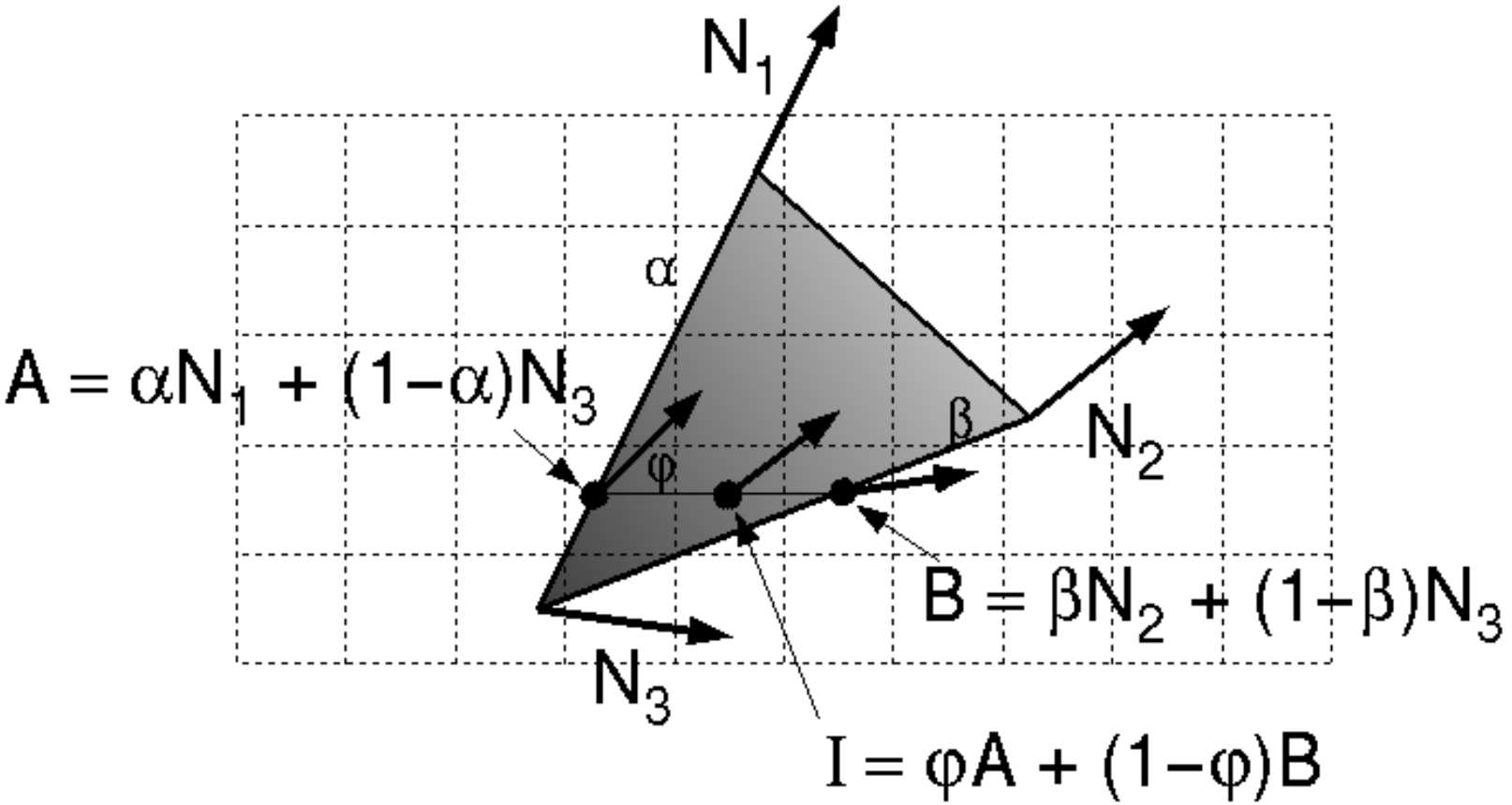
Phong Shading

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



Phong Shading

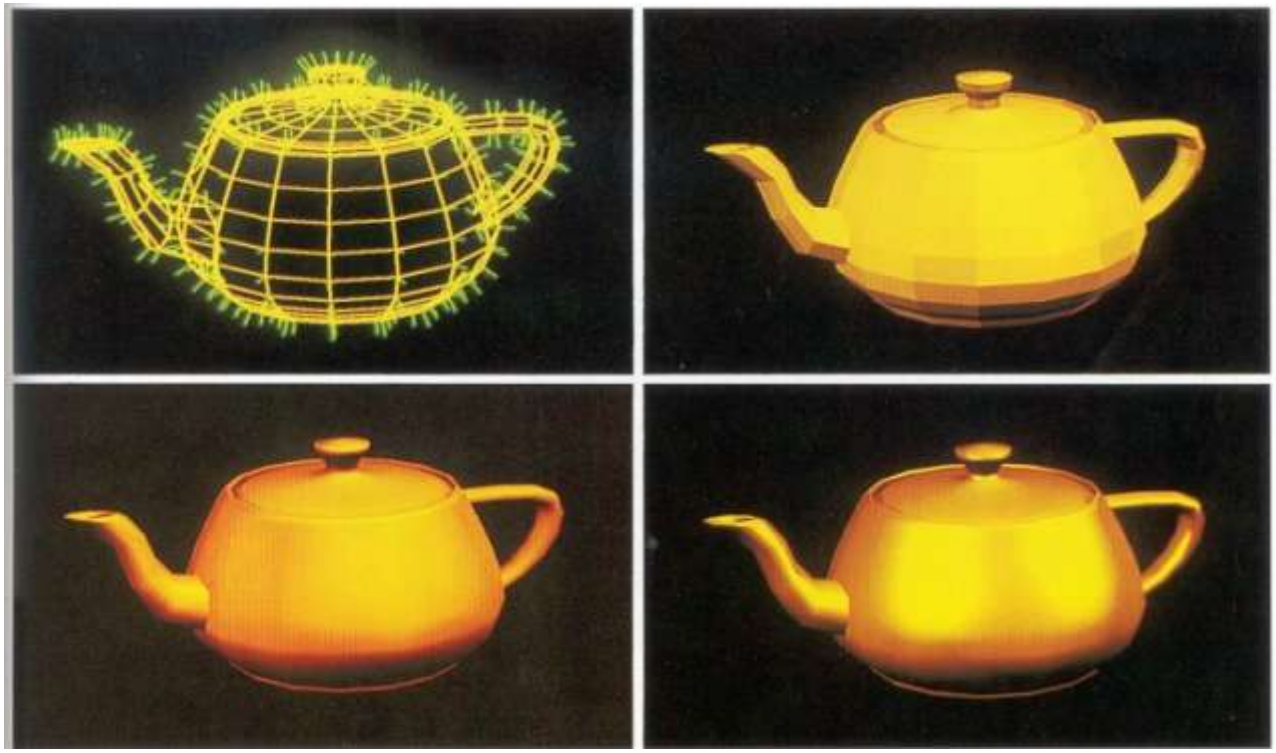
- Bilinearly interpolate surface normals at vertices down and across scan lines



Polygon Shading Algorithms

Wireframe

Flat

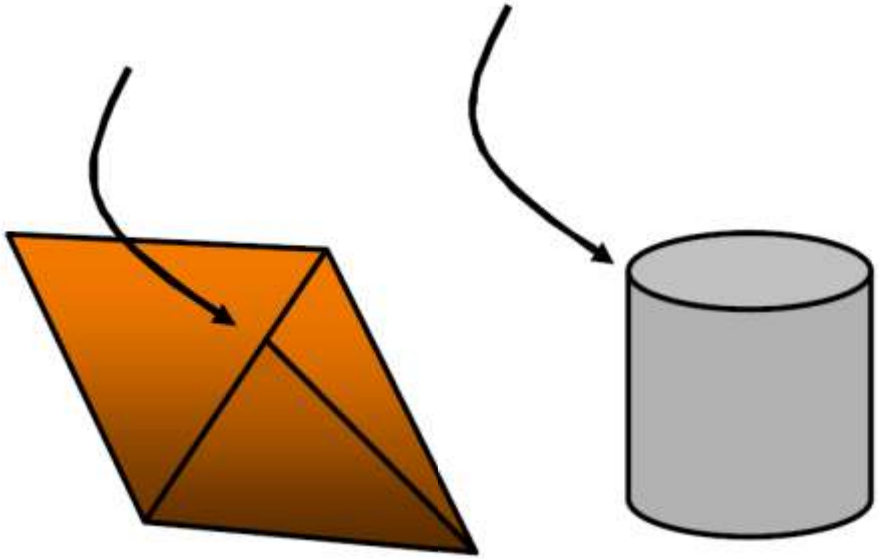


Gouraud

Phong

Shading Issues

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



Overview

- Scan conversion
  - Figure out 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

Surface Textures

Surface Textures

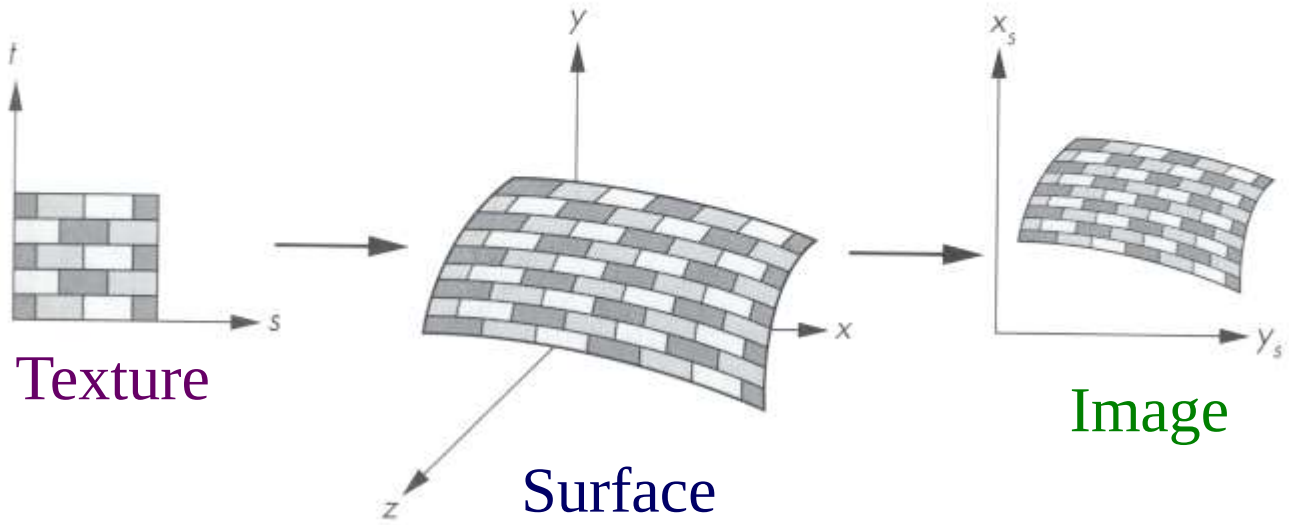
- Add visual detail to surfaces of 3D objects

Polygonal model

With surface texture

Textures

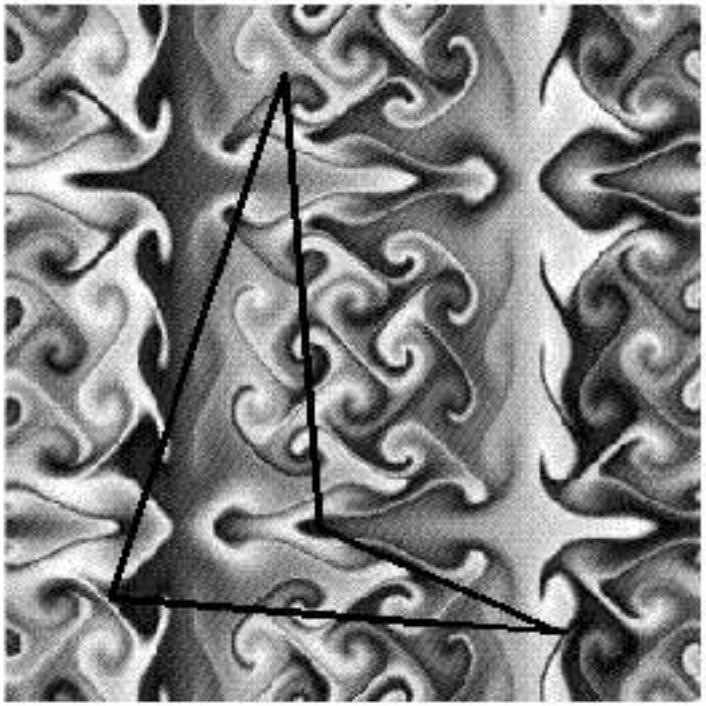
- Describe color variation in interior of 3D polygon
  - When scan converting a polygon, vary pixel colors according to values fetched from a texture

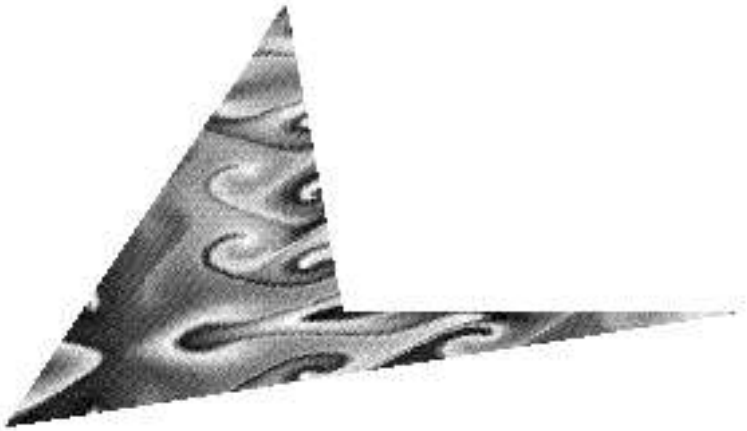


Angel Figure 9.3

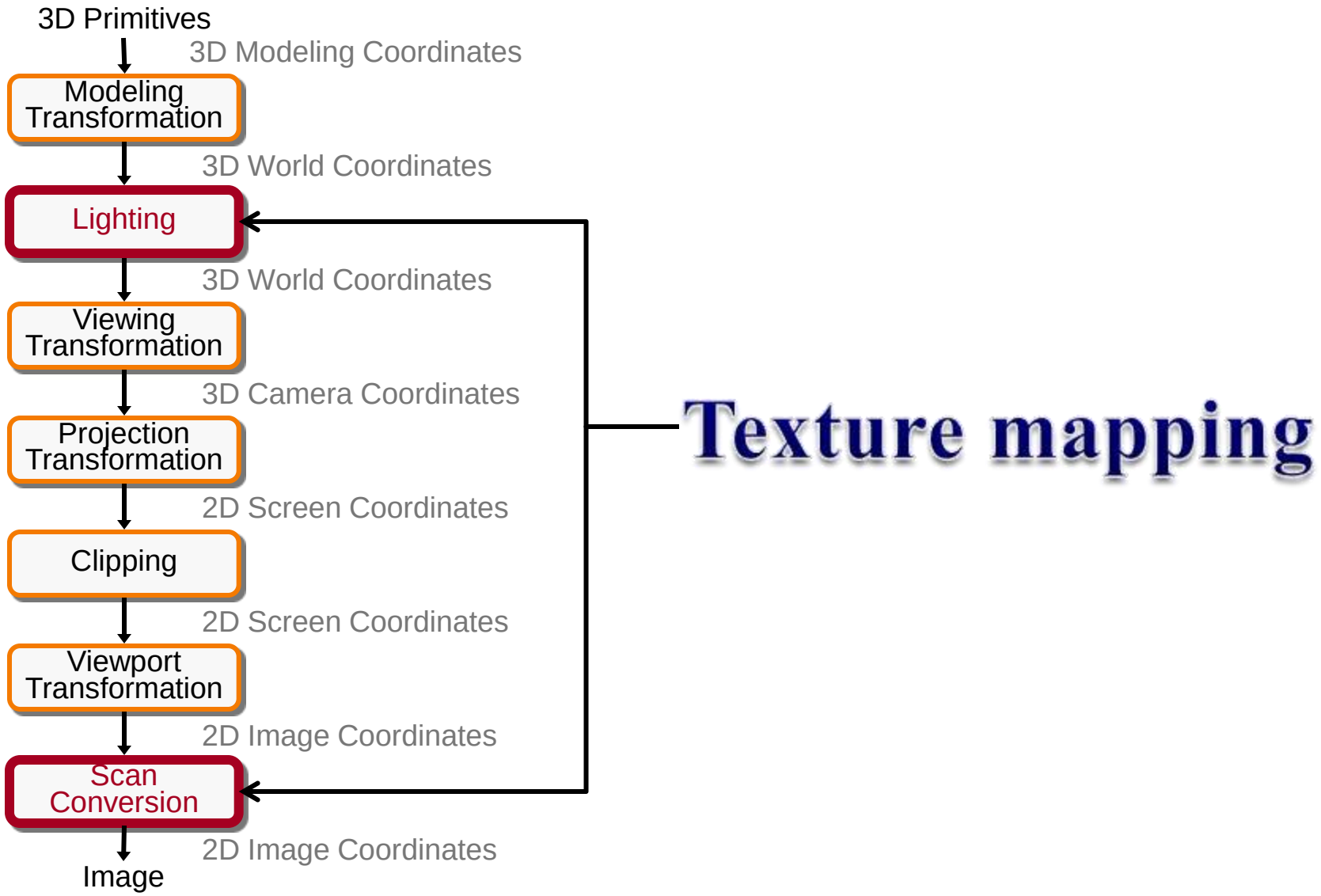
Textures

- We map coordinates in the 3D world to a T**exture**





3D Rendering Pipeline (for direct illumination)

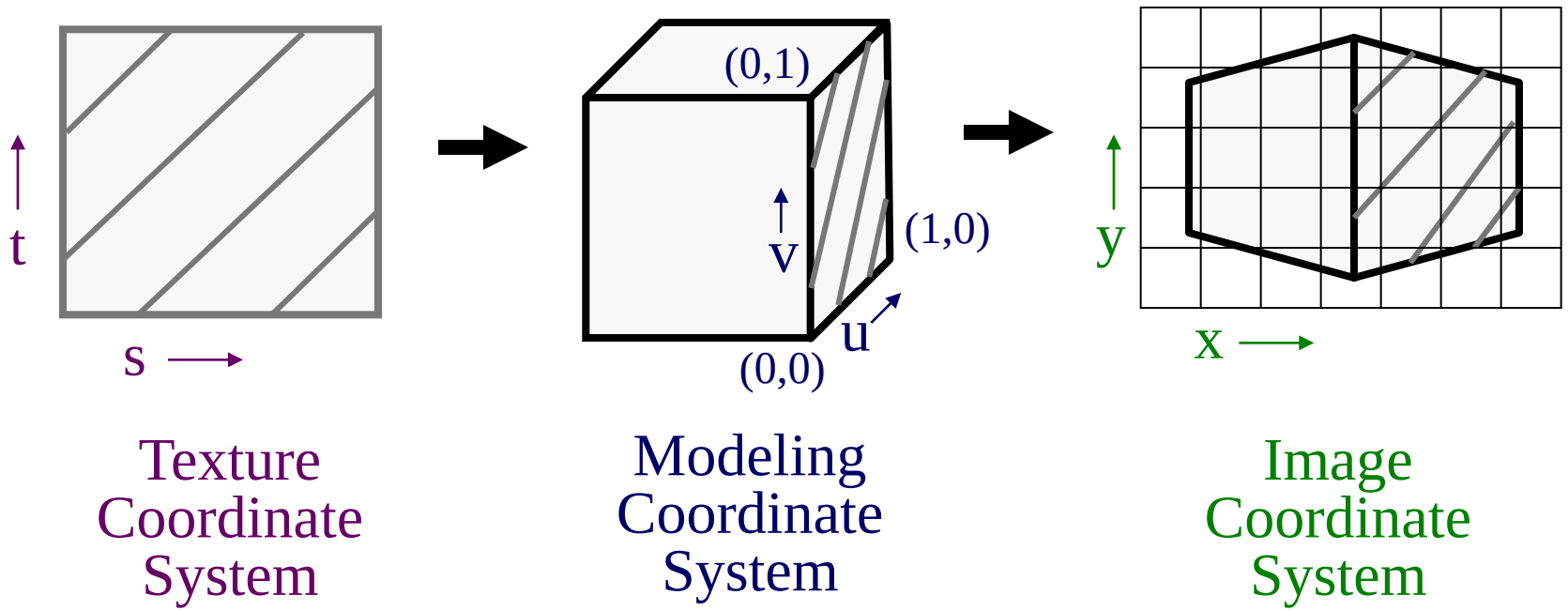


Overview

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

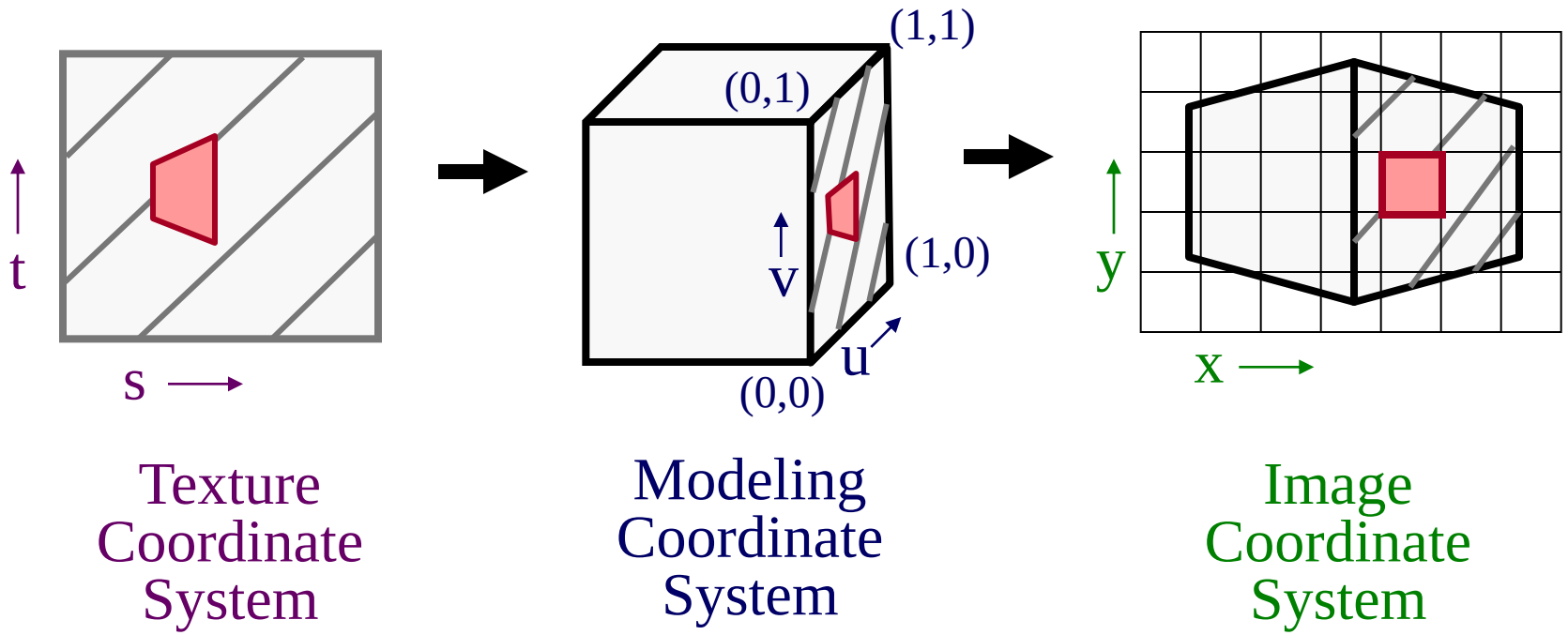
Texture Mapping

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



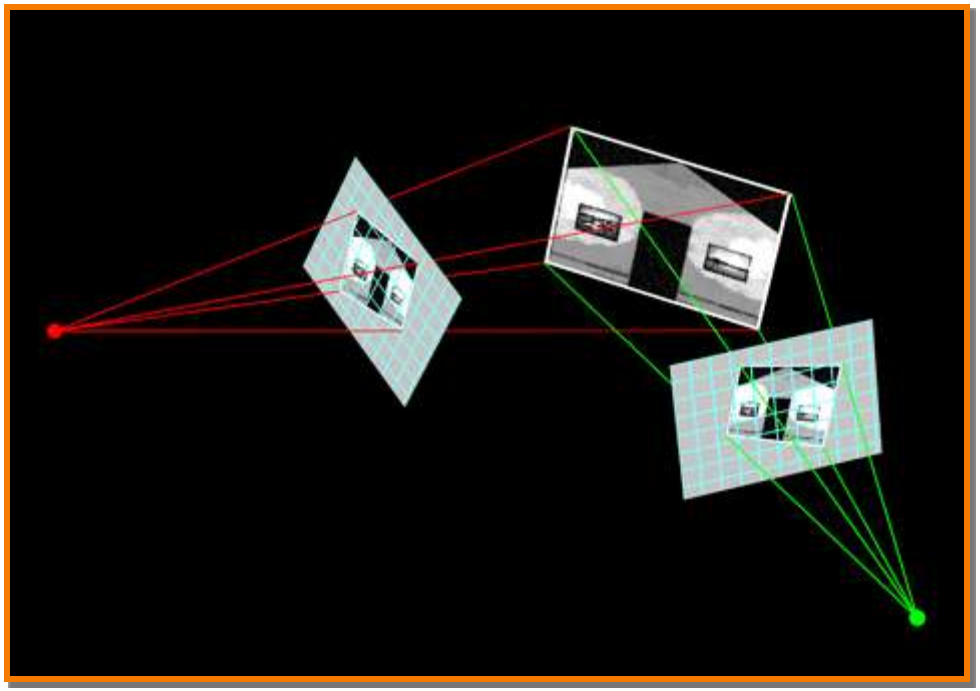
Texture Mapping

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



Texture Mapping

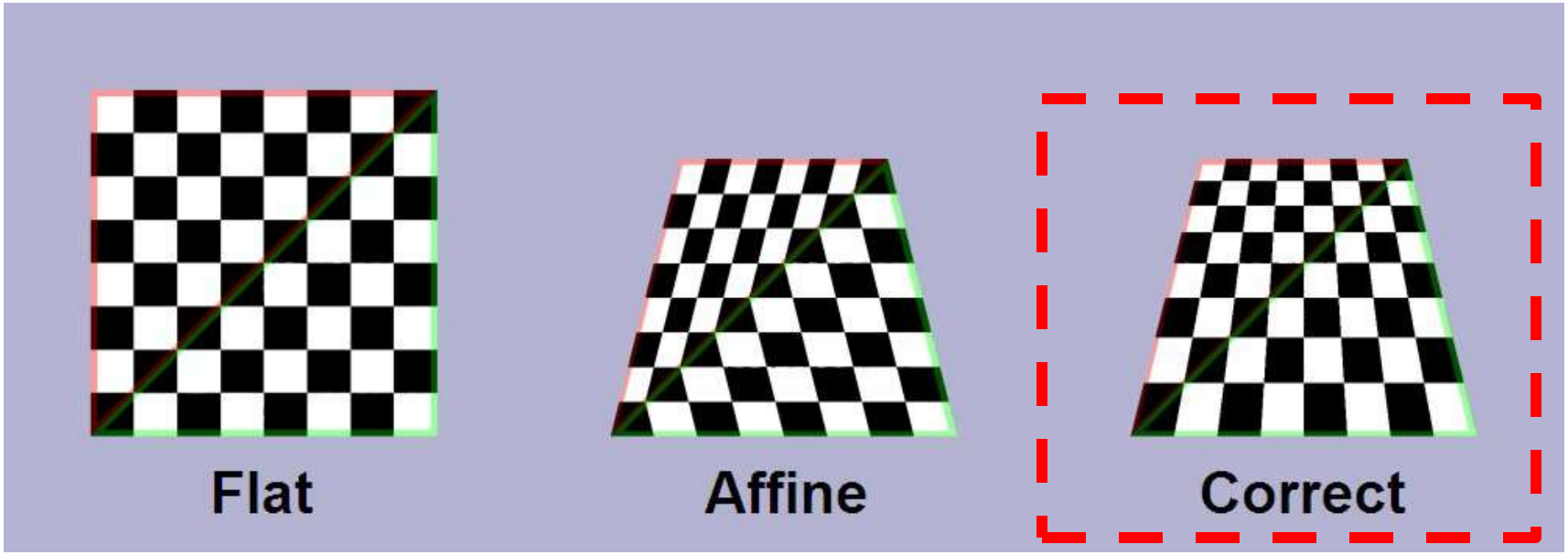
- Texture mapping is a 2D projective transformation
  - texture coordinate system: (t,s) to
  - image coordinate system (x,y)



Chris Buehler & Leonard McMillan, MIT

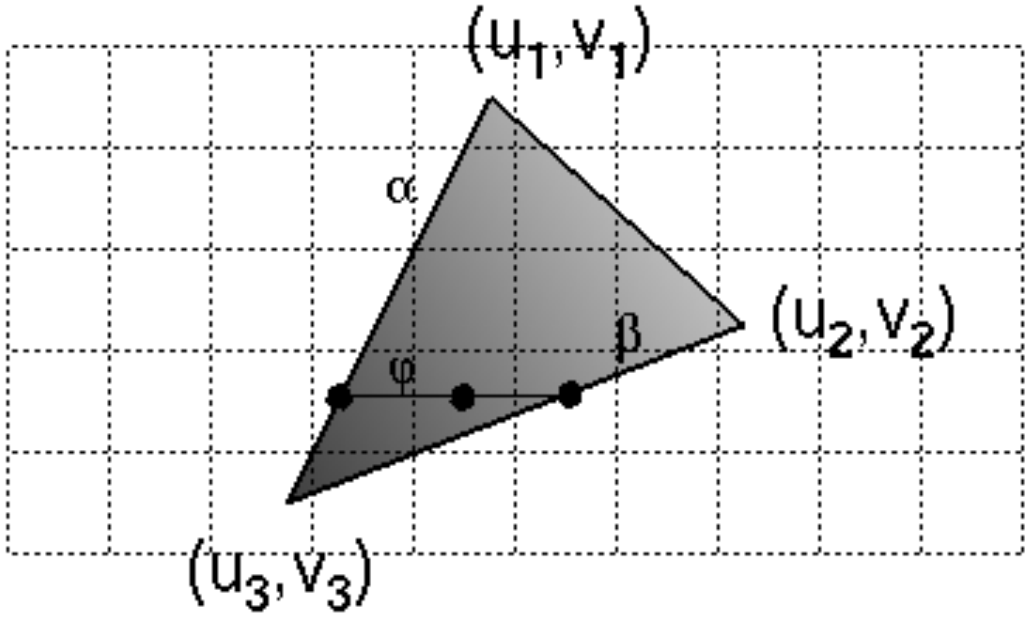
Texture Mapping

- Scan conversion
  - Interpolate texture coordinates down/across scan lines
  - Distortion due to bilinear interpolation approximation

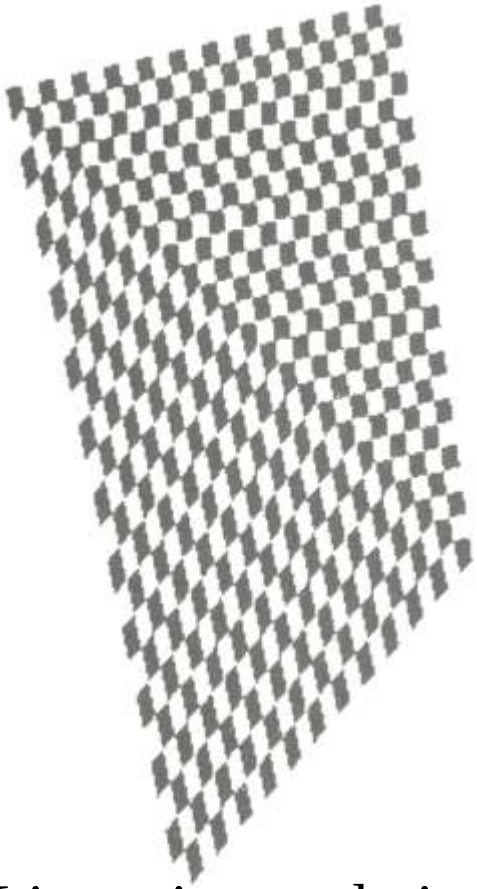


Texture Mapping

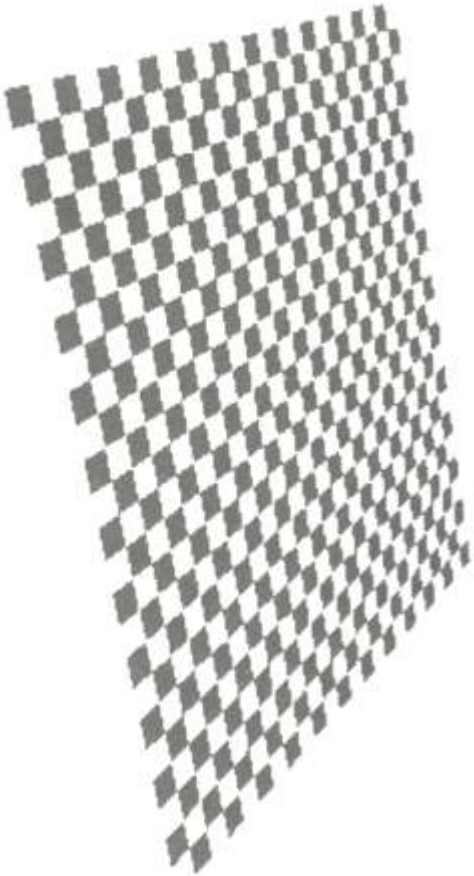
- Scan conversion
  - Interpolate texture coordinates down/across scan lines
  - Distortion due to bilinear interpolation approximation
    - Cut polygons into smaller ones, or
    - Perspective divide at each pixel



Texture Mapping



Linear interpolation
of texture coordinates



Correct interpolation
with perspective divide

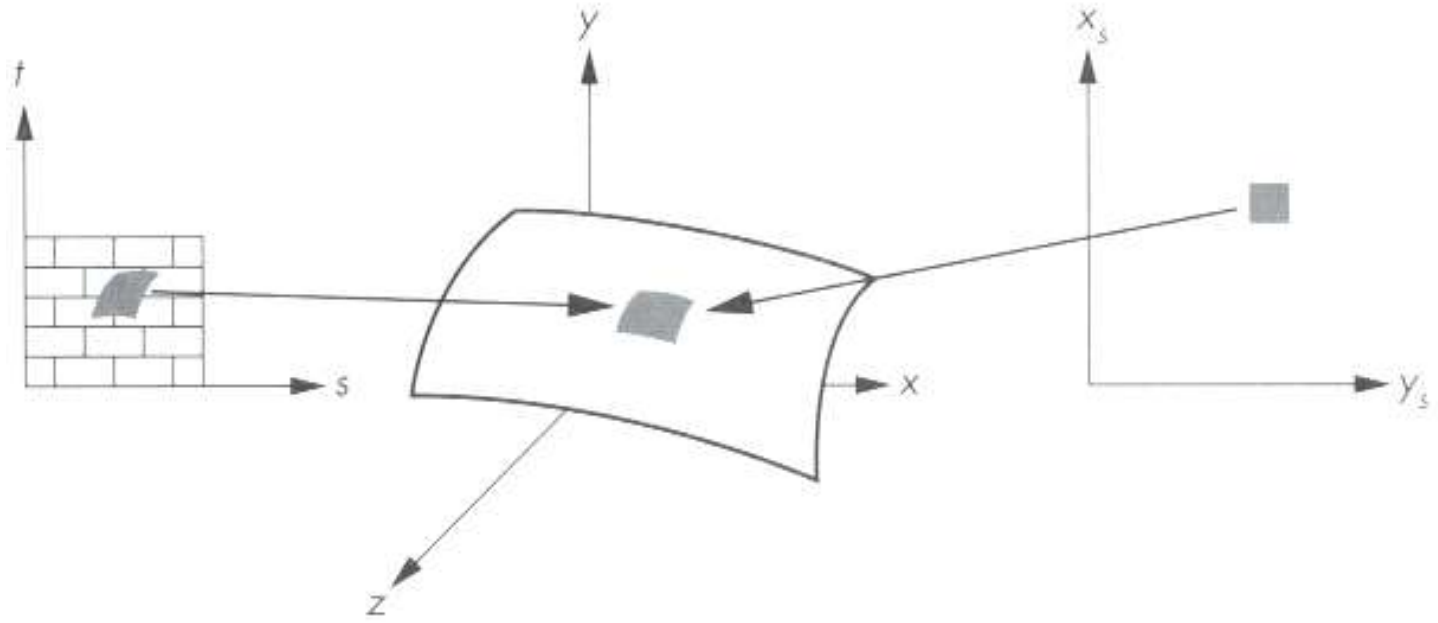
Hill Figure 8.42

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

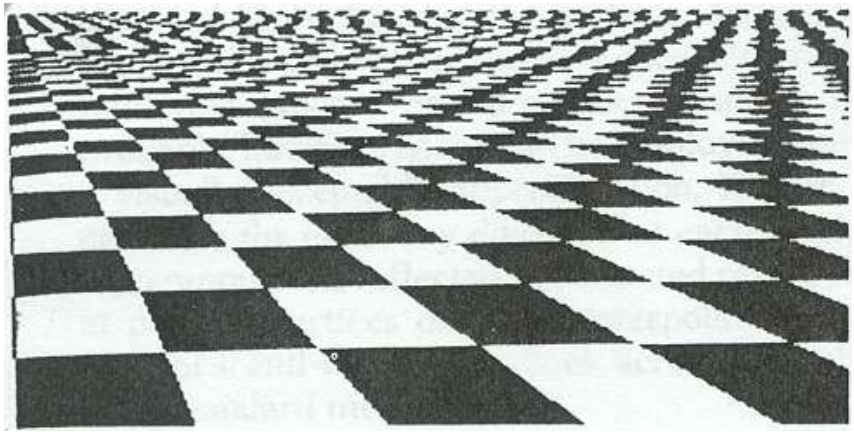
- Must sample texture to determine color at each pixel in image



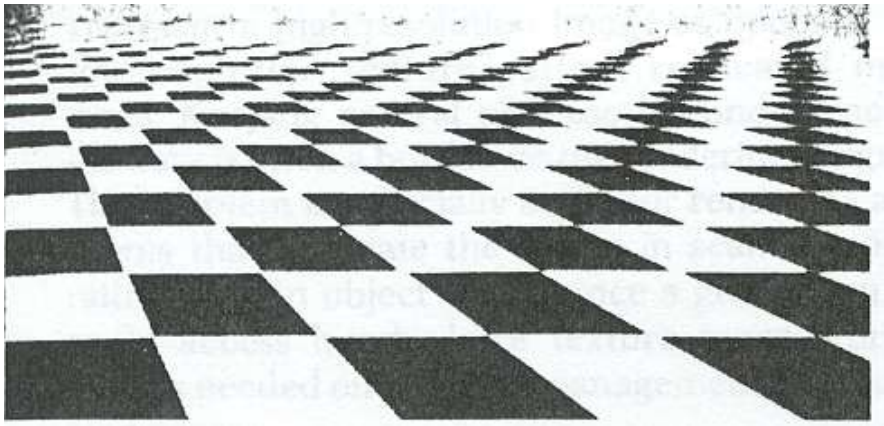
Angel Figure 9.4

Texture Filtering

- Aliasing is a problem



Point sampling

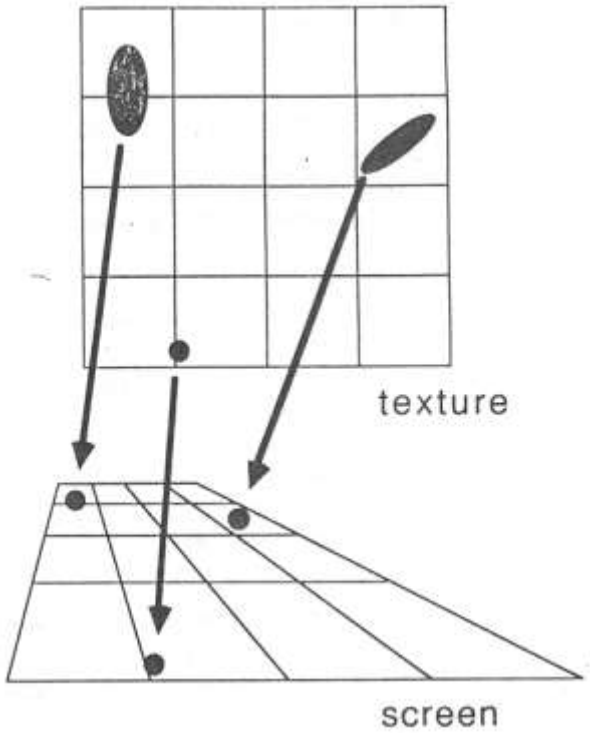


Area filtering

Angel Figure 9.5

Texture Filtering

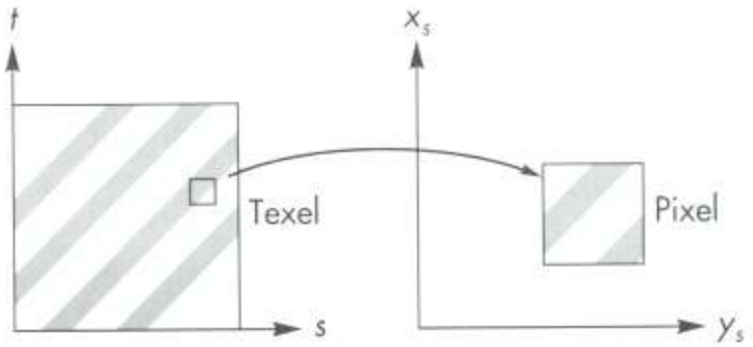
- Ideally, use elliptically shaped convolution filters



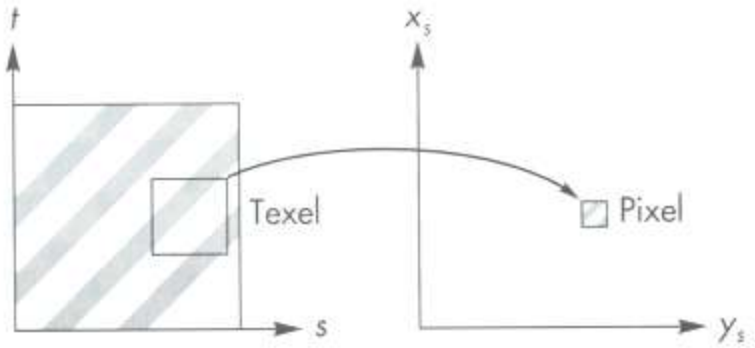
In practice, use rectangles

Texture Filtering

- Size of filter depends on projective warp
  - Can pre-filter images
    - Mipmapping
    - Summed area tables



Magnification



Minification

Angel Figure 9.14

Mipmapping

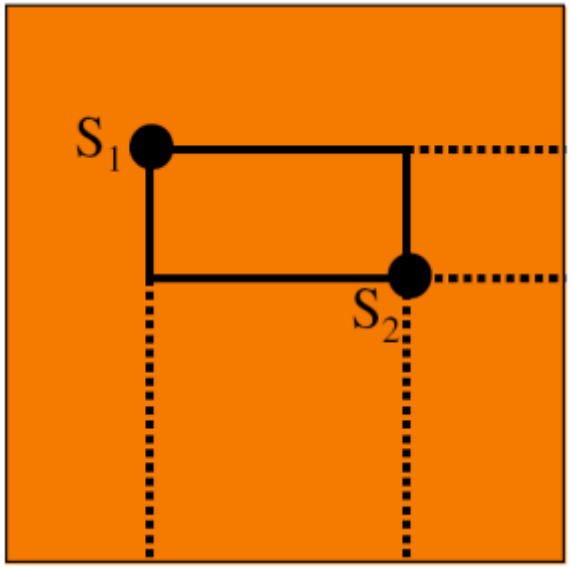
- Keep textures pre-filtered at multiple resolutions
  - How do we sample?
    - Nearest neighbor with Mipmapping
    - Bilinear filtering

Mipmapping

- Keep textures pre-filtered at multiple resolutions
  - For each pixel, linearly interpolate between two closest levels (e.g., **tri-linear** filtering)
  - Fast, easy for hardware

Summed-area tables

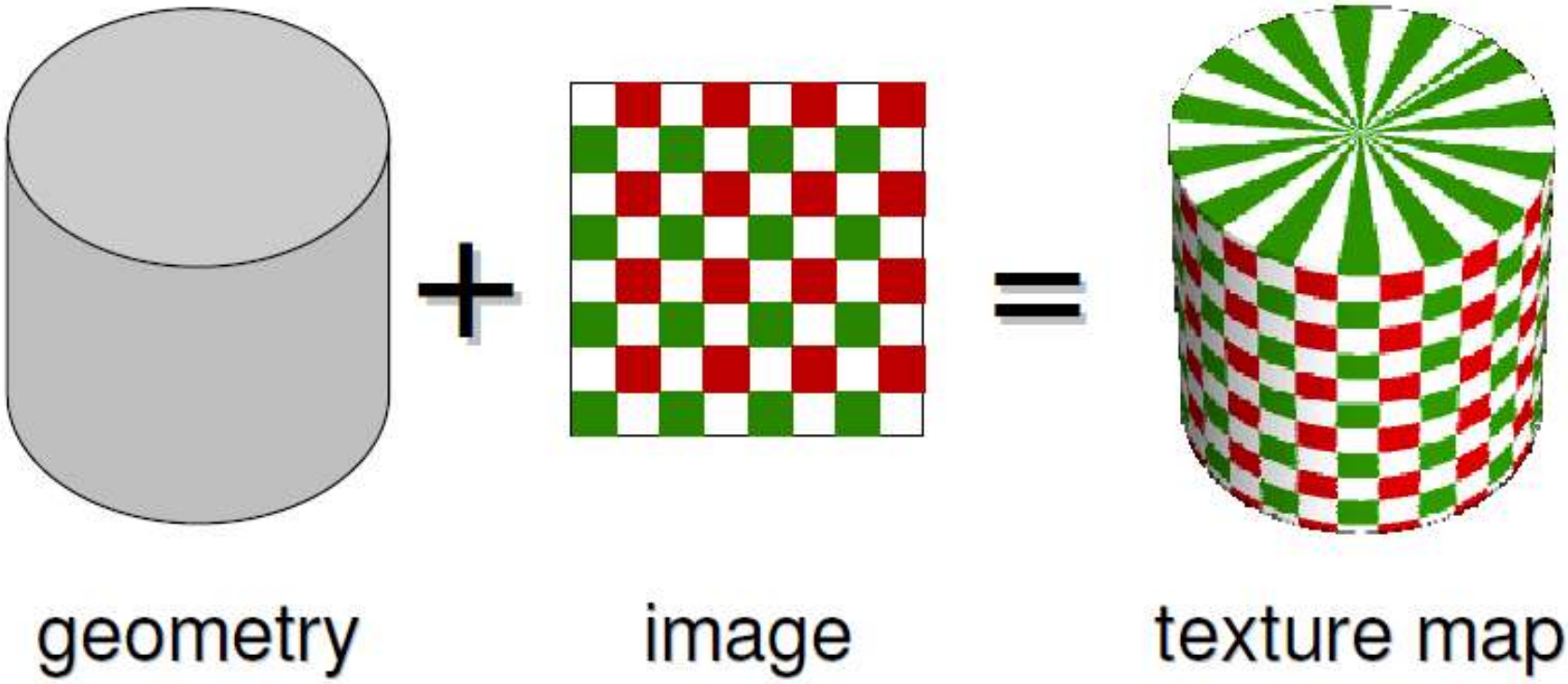
- At each Texel keep sum of all values down&right
  - We can compute sum of all values within a rectangle in constant time (four entries)
  - Better ability to capture very oblique projections
  - But, cannot store values in a single byte



Overview

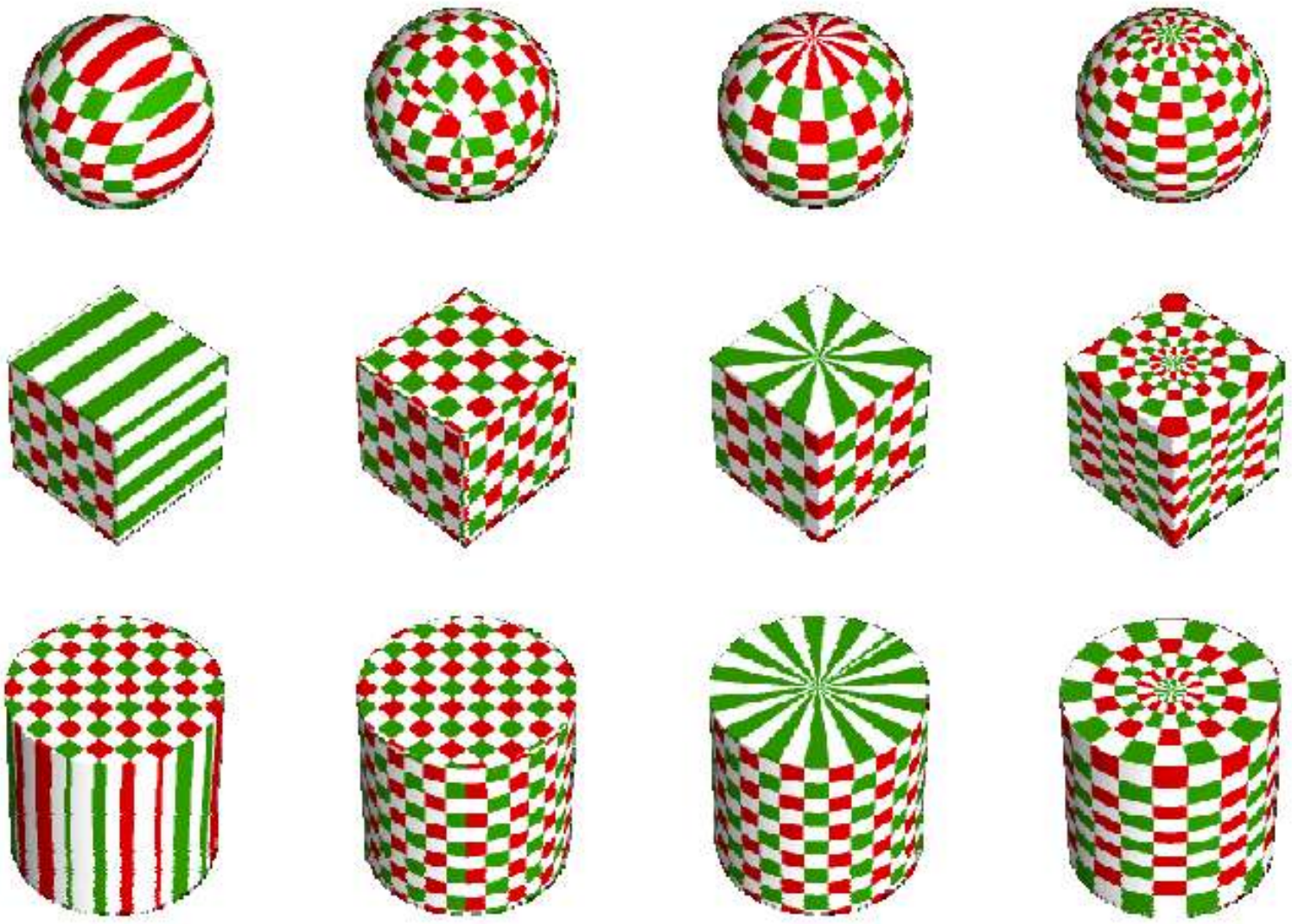
- Texture mapping methods
  - Mapping
  - Filtering
  - Parameterization
- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Non-photorealistic rendering

Parameterization

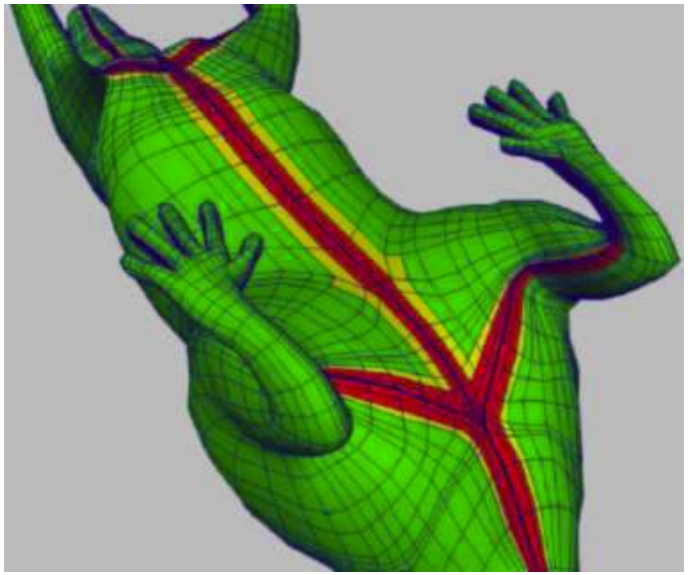


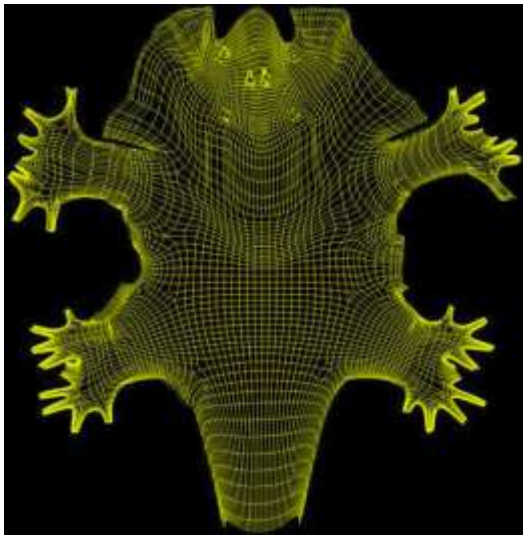
- Q: How do we decide *where* on the geometry each color from the image should go?

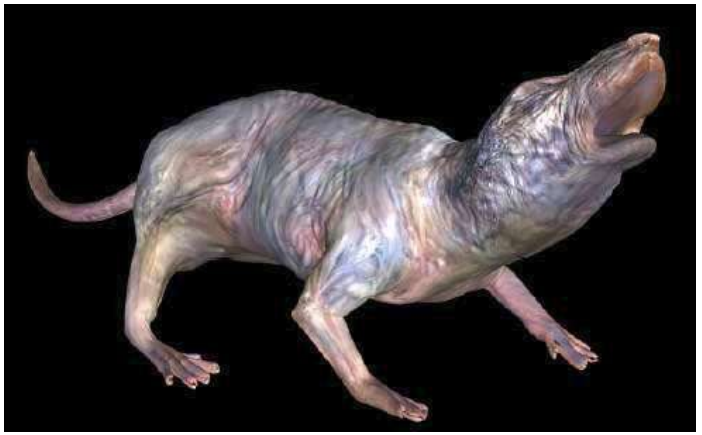
Varieties of projections



Option: unfold the surface



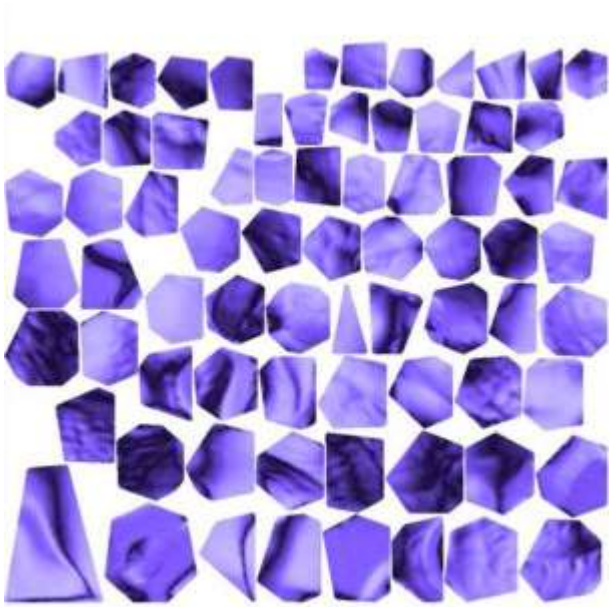




See interesting work here: http://www.cs.ubc.ca/~sheffa/proj.html

Option: Make an Atlas

charts



atlas

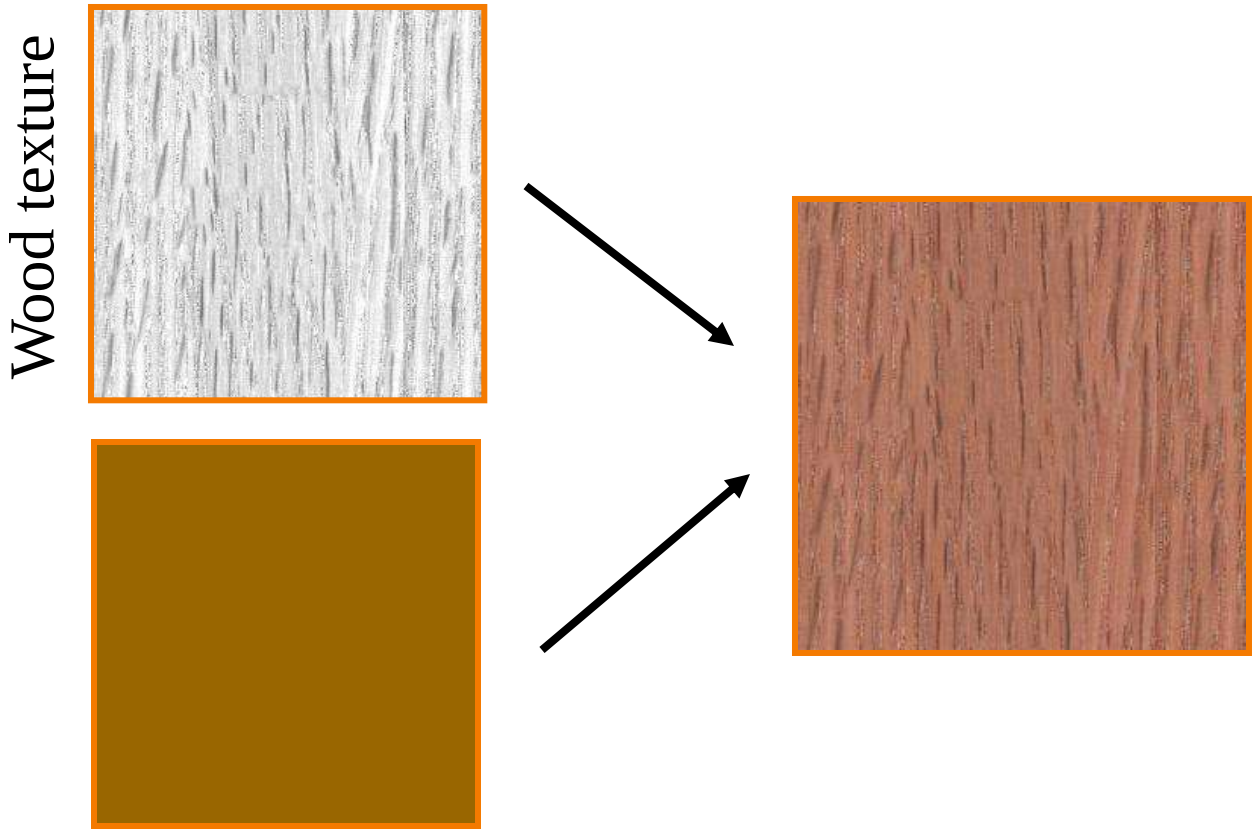
surface

Overview

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

Modulation textures

- Map texture values to scale factor



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

Illumination Mapping

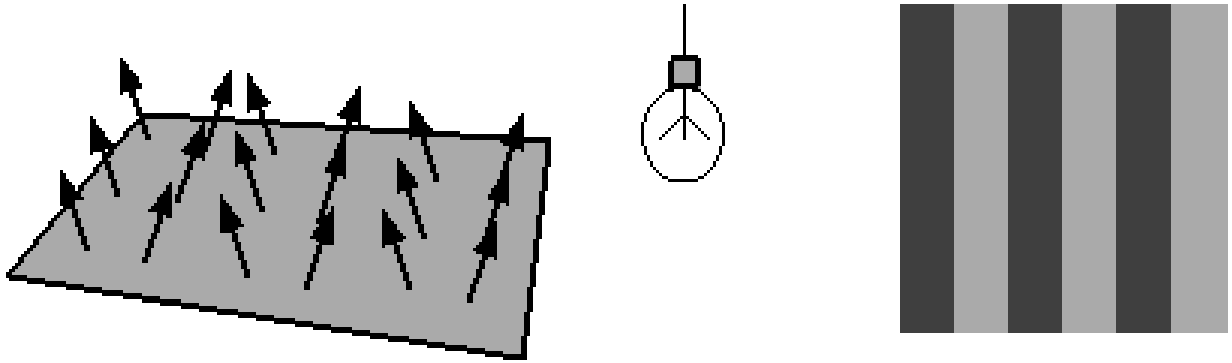
- Map texture values to surface material parameter
  - K_A
  - K_D
  - K_S
  - K_T
  - n

$K_T = T(s,t)$

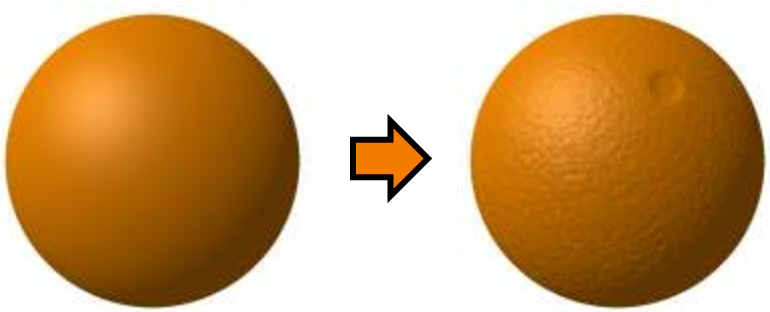
$$I = I_E + K_A I_A + \sum_L (K_D (N \bullet L) + K_S (V \bullet R)^n) S_L I_L + K_T I_T + K_S I_S$$

Bump Mapping

- Map texture values to perturbations of surface normals

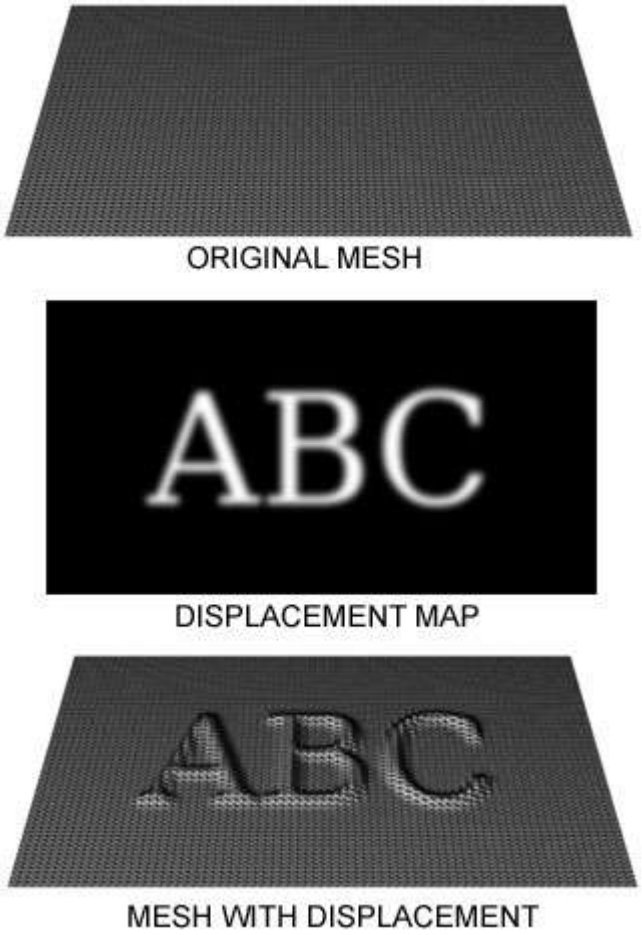


Bump Mapping



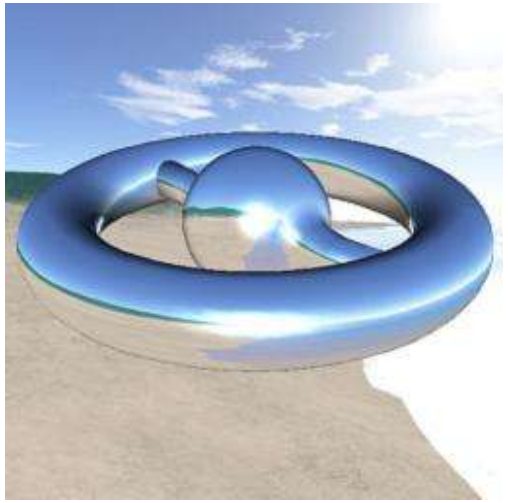
Displacement Mapping

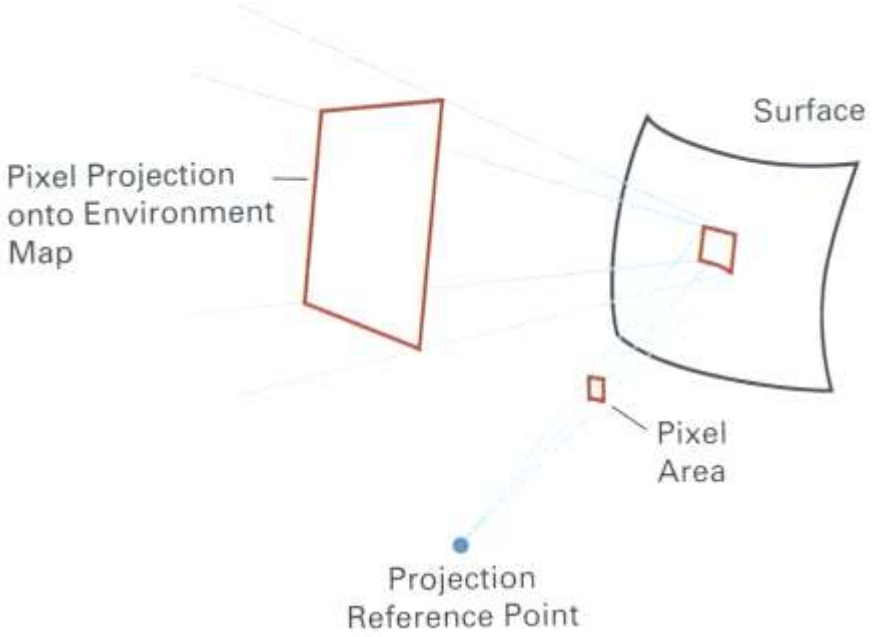
- Actually change geometric position of points over the textured surface
- Requires adaptive tessellation
- Until recently existed only in high-end rendering systems



Environment/Reflection Mapping

- Map texture values to perturbations of surface normals

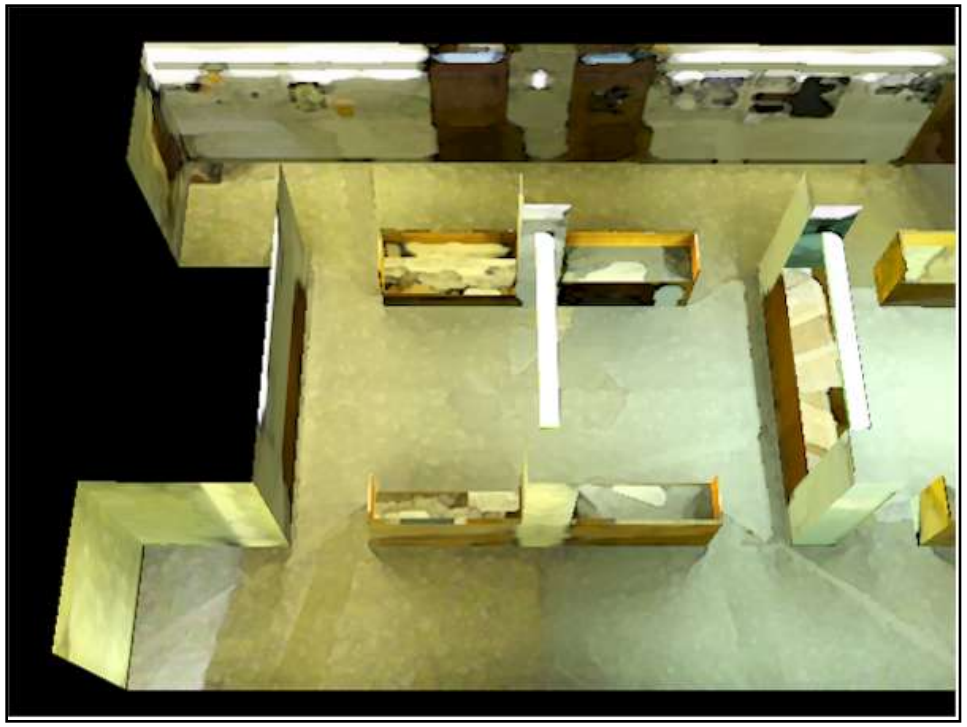




H&B Figure 14.93

Image-Based Rendering

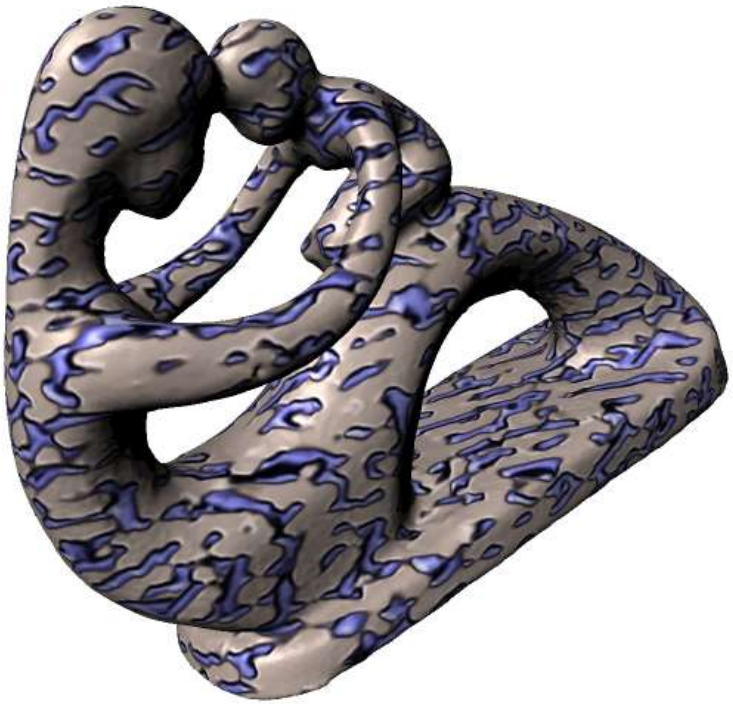
- Map photographic textures to provide details for coarsely detailed polygonal model



More info here: http://en.wikipedia.org/wiki/Image-based_modeling_and_rendering

Solid Textures

- Texture values indexed by 3D location
  - Expensive storage, or
  - Compute on the fly, E.g Perlin noise



See interesting results here: http://johanneskopf.de/publications/solid/

Solid Textures

- Texture values indexed by 3D location
  - Expensive storage, or
  - Compute on the fly, E.g Perlin noise