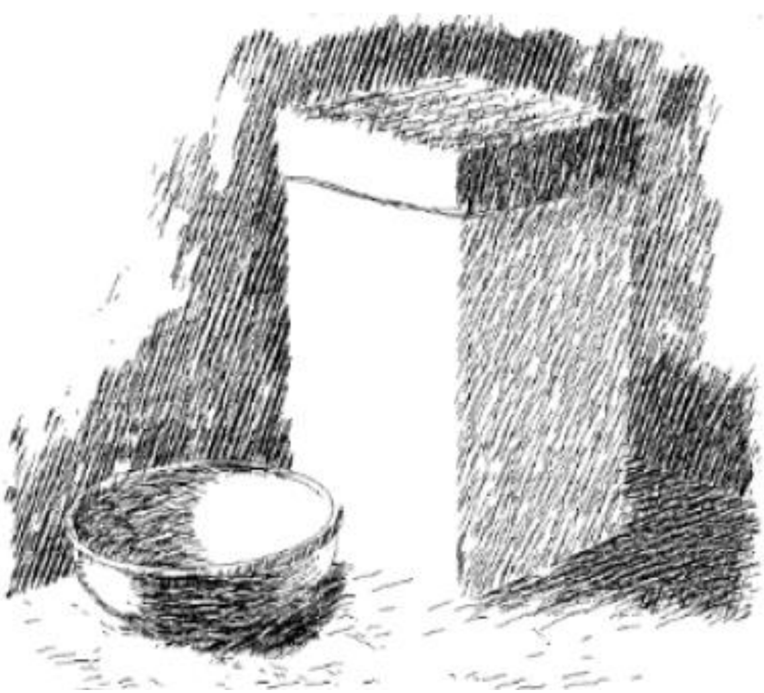
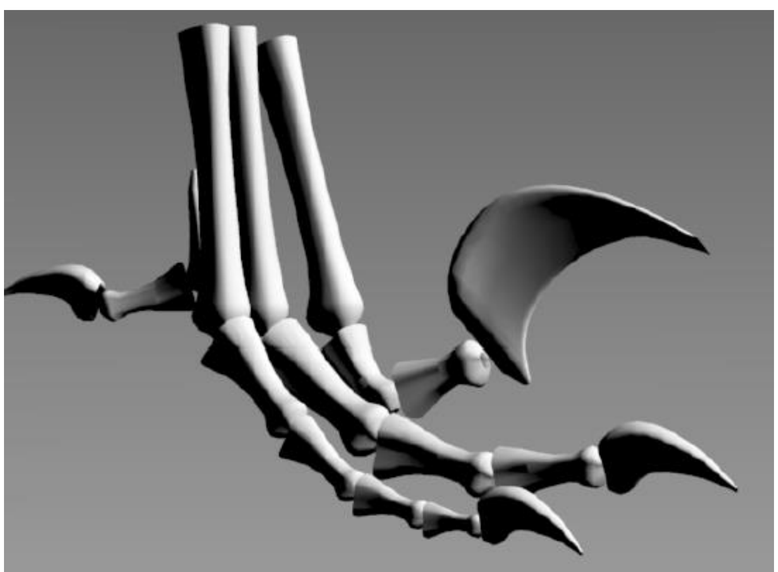
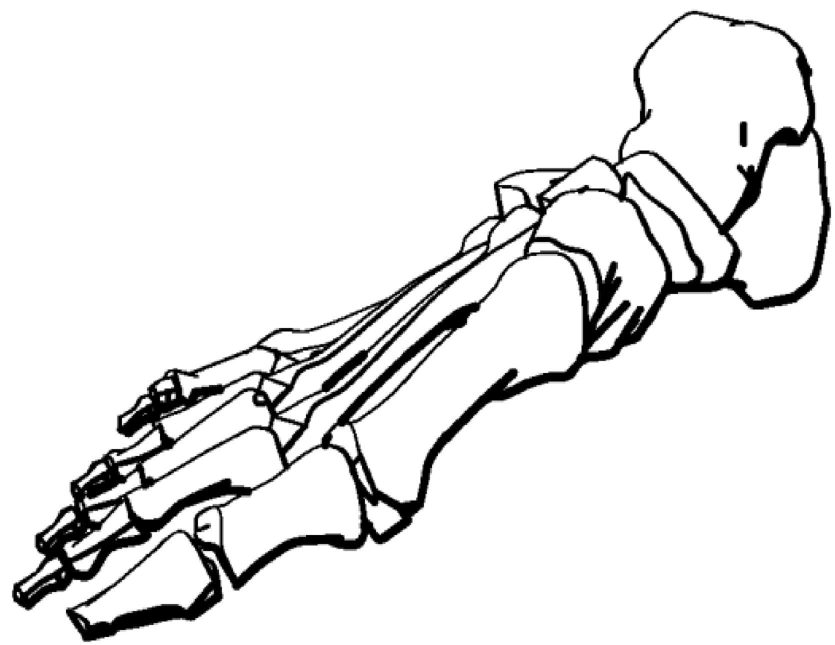
Non-Photorealistic Rendering (NPR)



Non-photorealistic Rendering

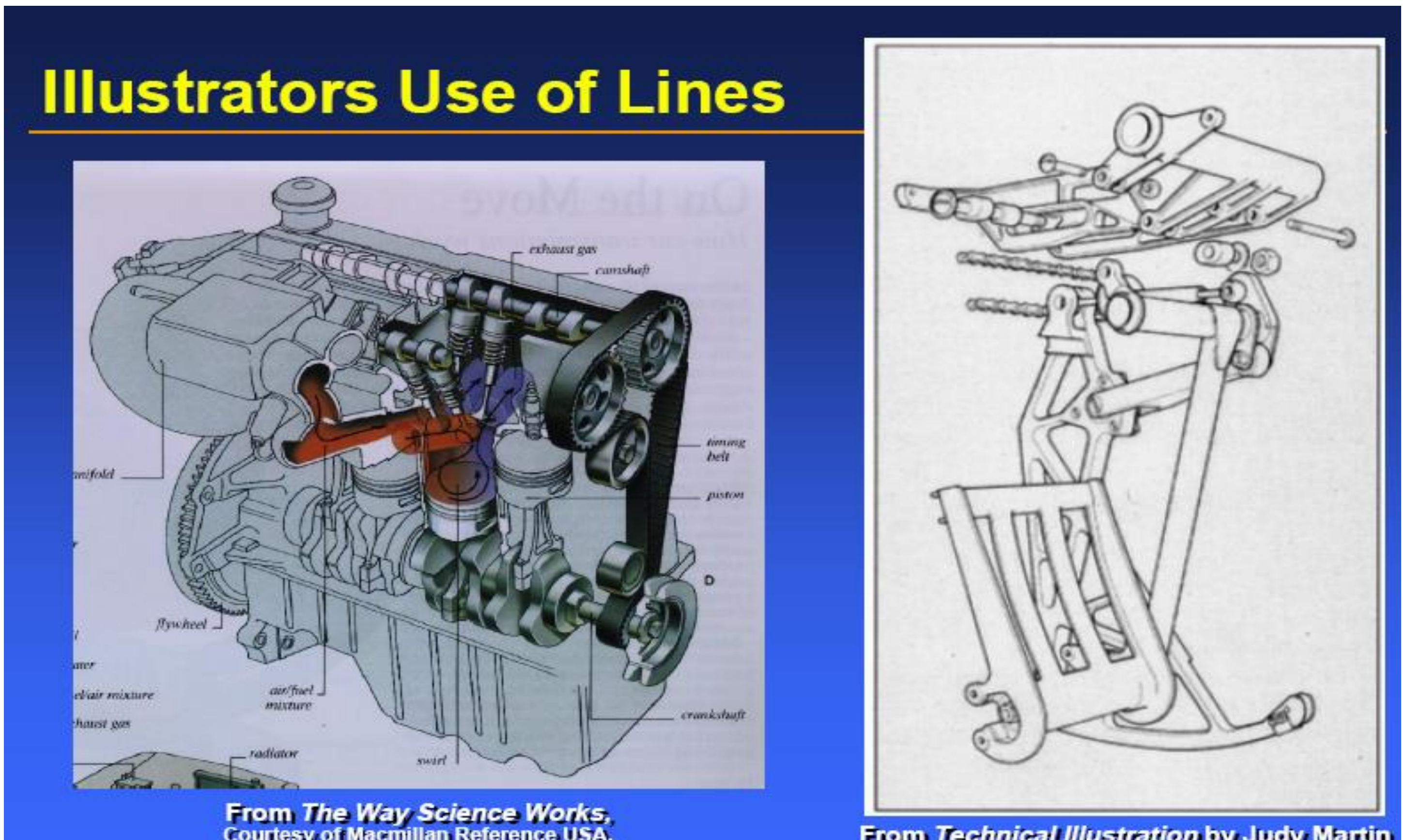
- Most computer graphics work strives for photorealism
- Other types of depiction can be more expressive or artistic





NPR – Technical Illustration

- Illustrate important features

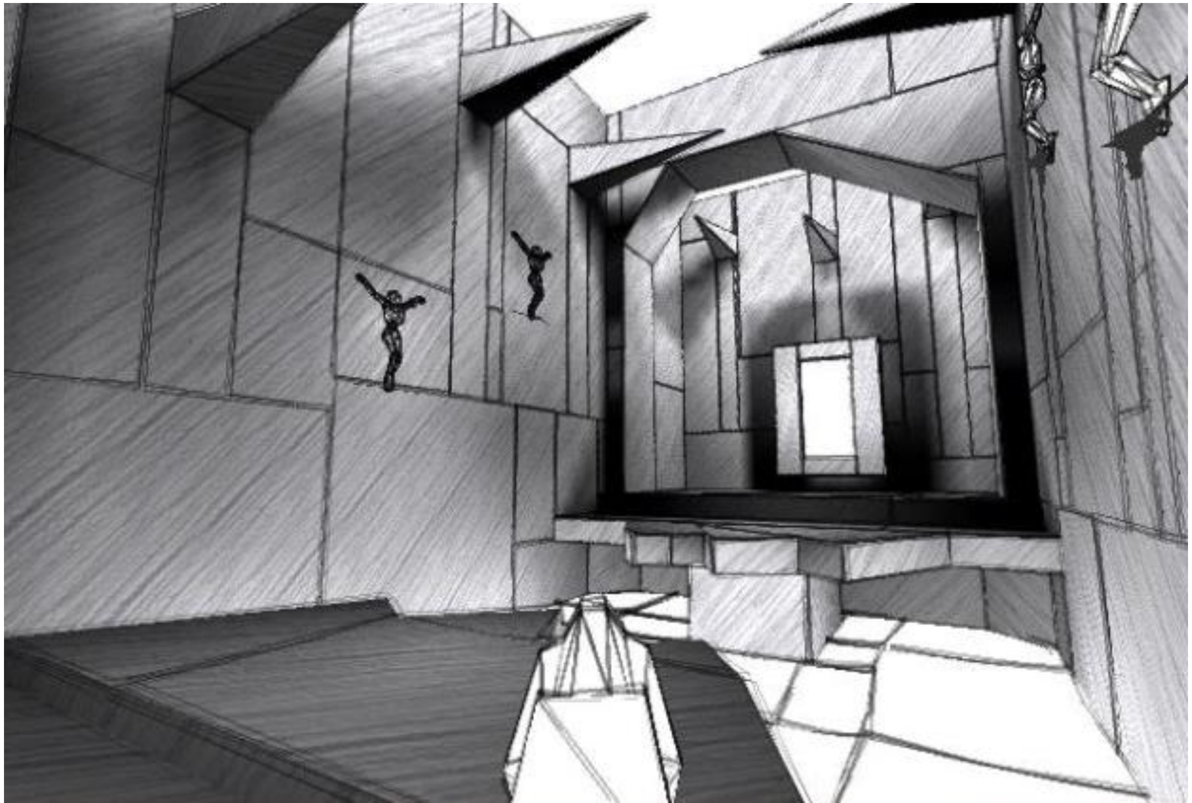


NPR – Painterly Rendering

- Make it look like being created using brush strokes and paint

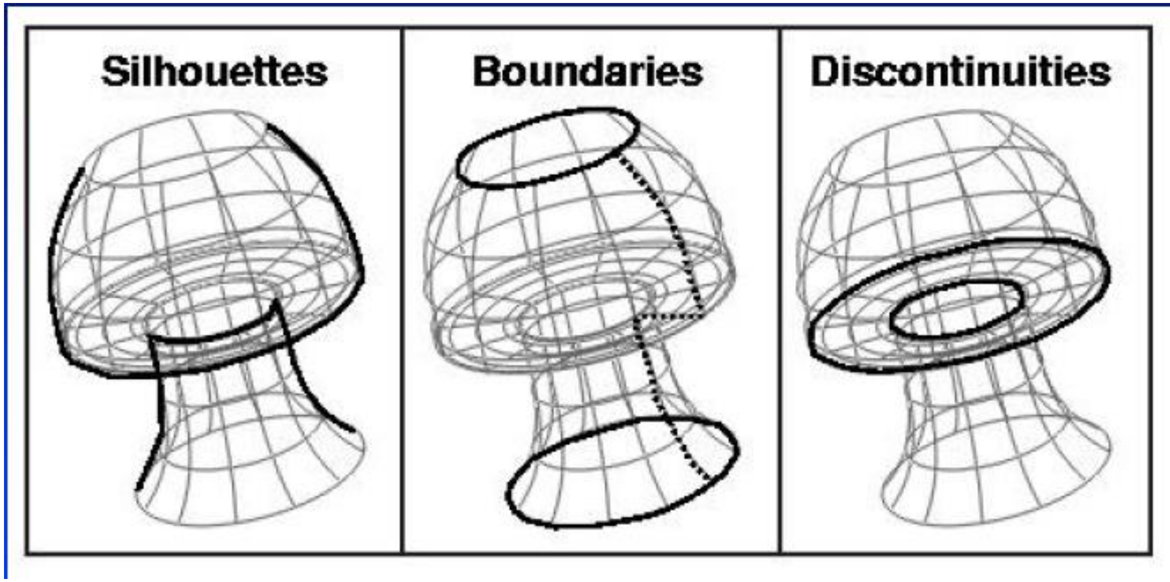
NPR – Sketchy Rendering

- Make it look like being created with pencil sketch



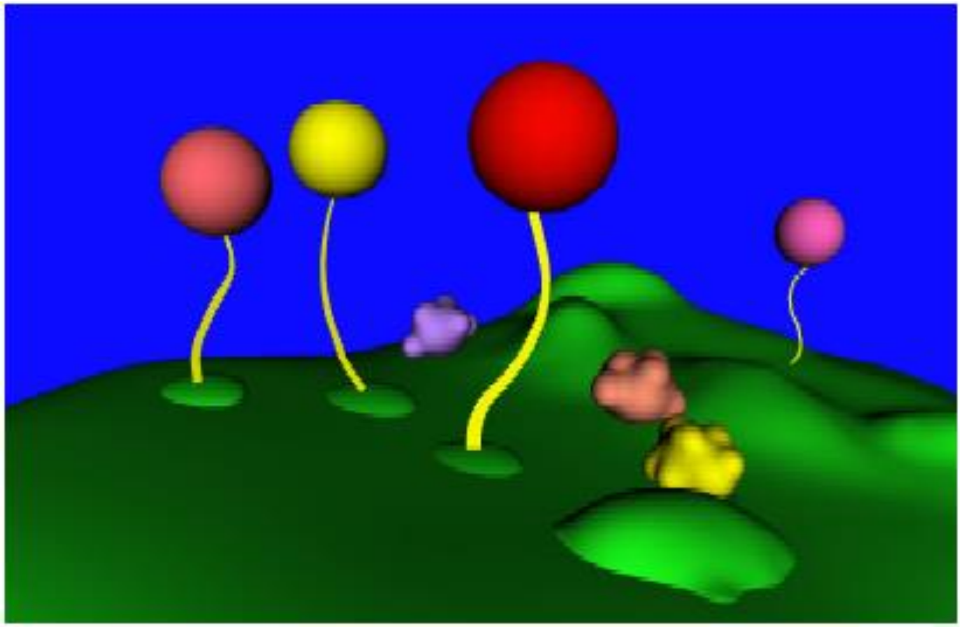
Shape Abstraction by Lines

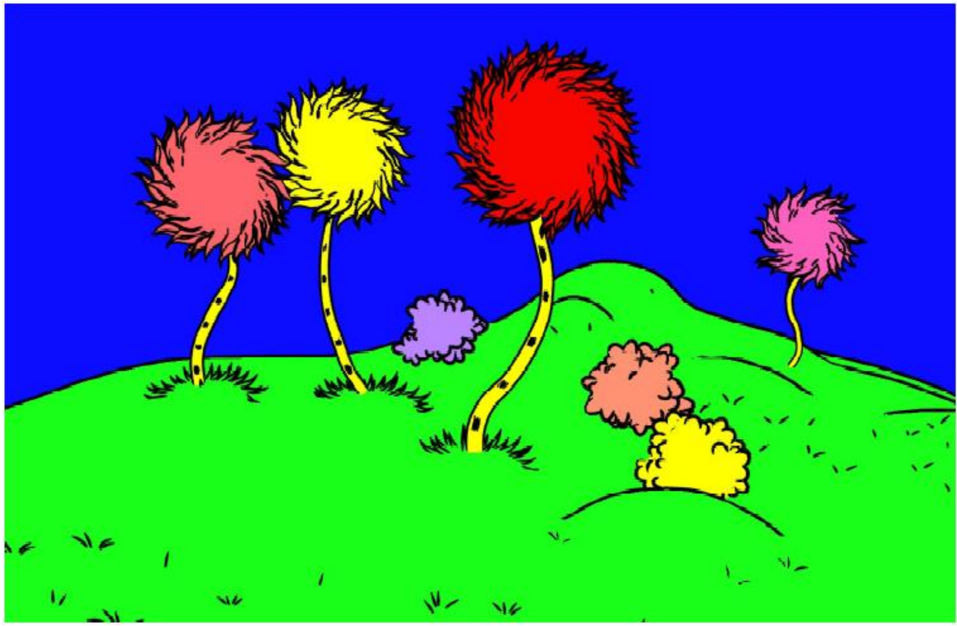
- Boundary Lines
- Silhouette Lines
- Creases
- Material Edges



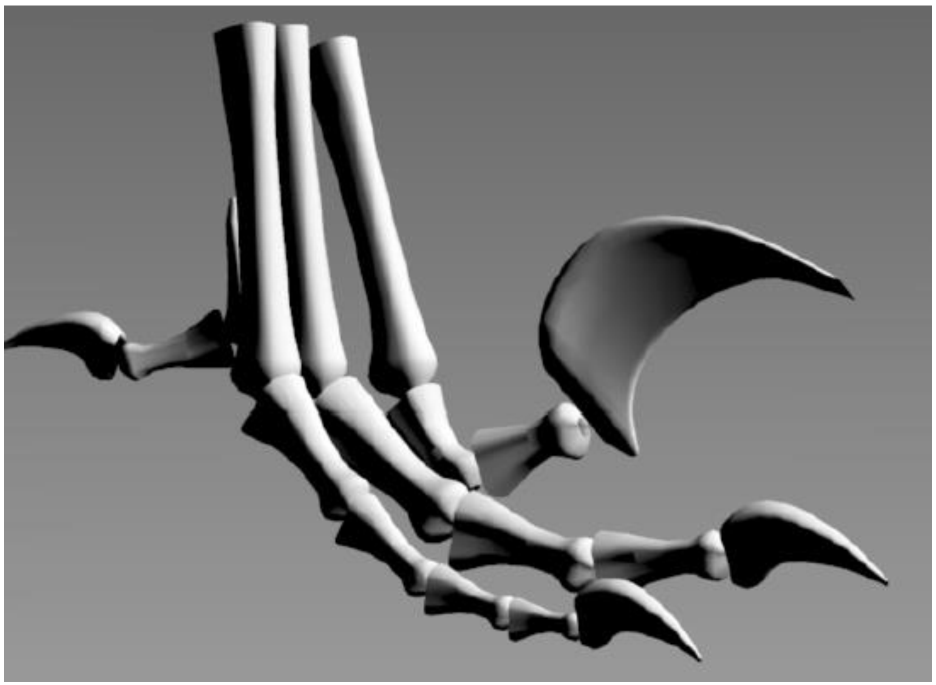
Shape Abstractions by Lines

- Various line styles can be used

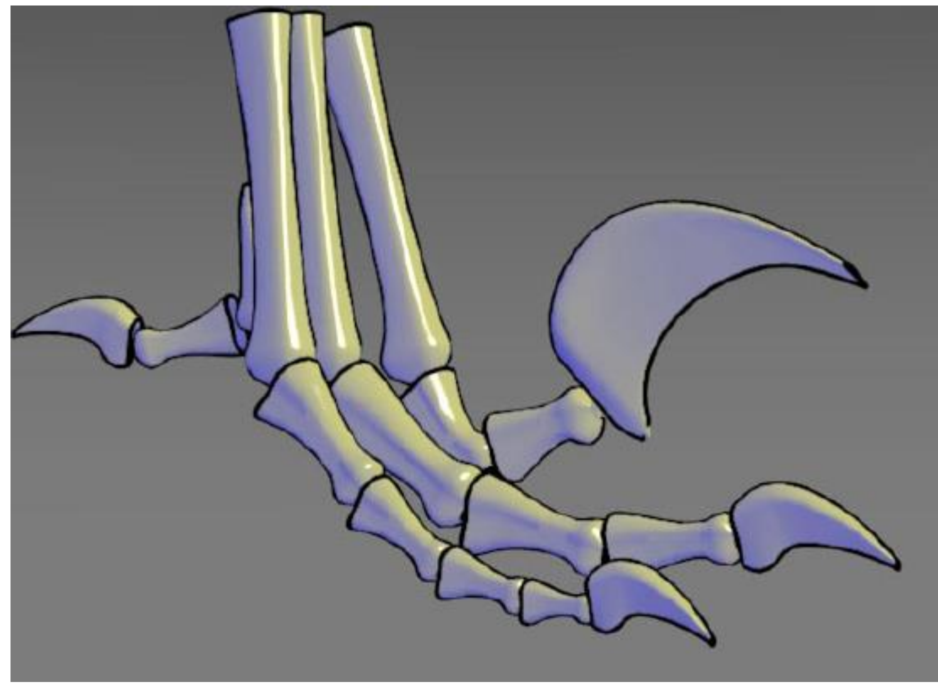




Shape Abstraction by Shading



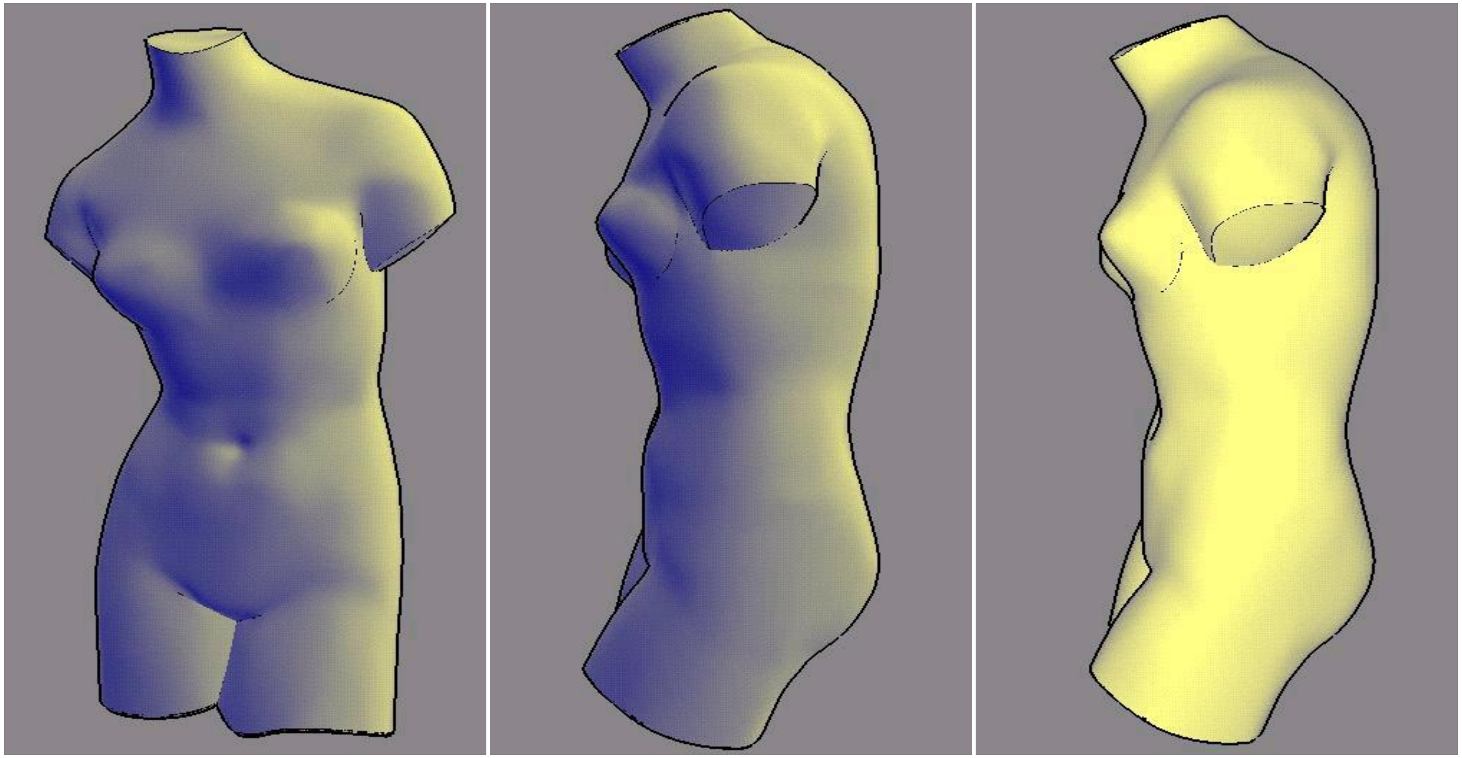
Regular OpenGL Gouraud Shading



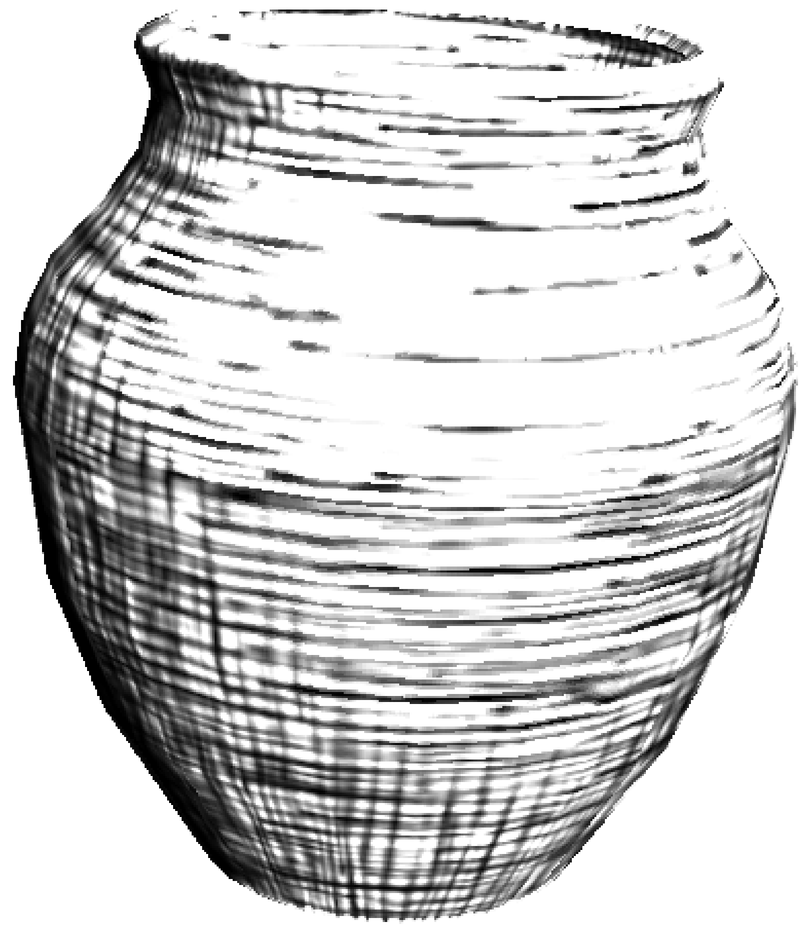
Tone Shading

Shape Abstraction by Shading

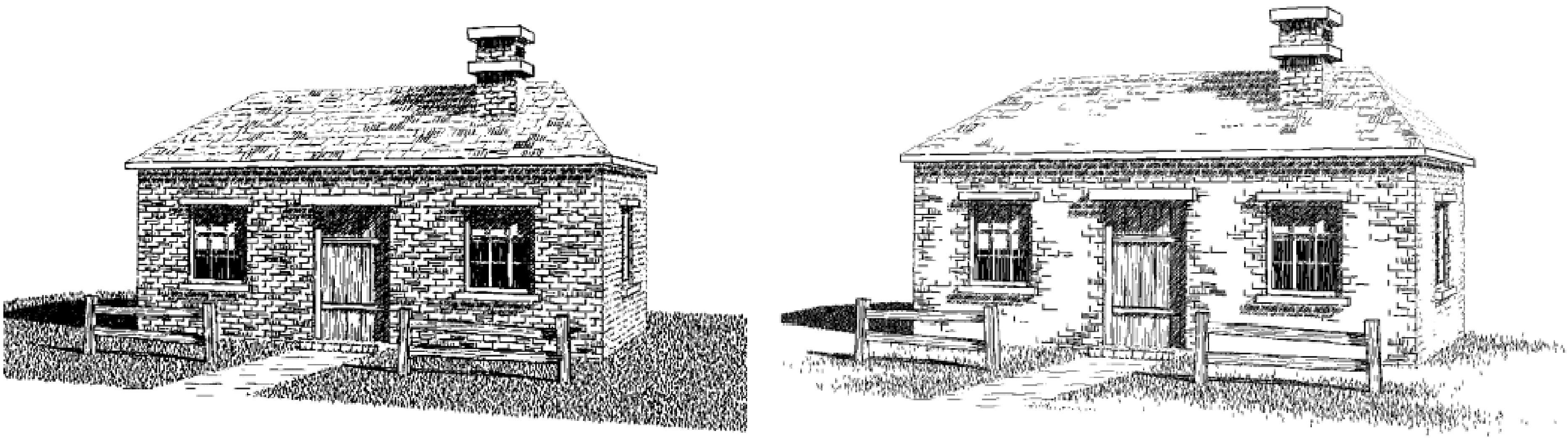
- More effective when combined with lines



Shape Abstraction by Textures



Shape Abstraction by Textures

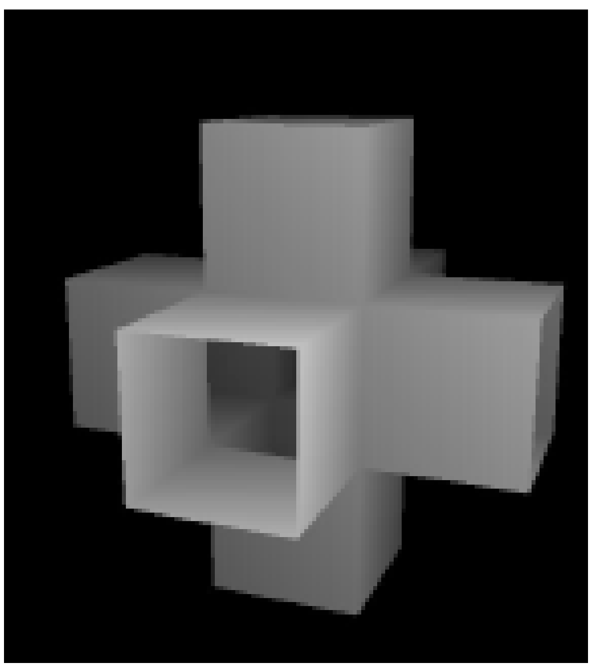


Feature Line Detection

- Image space method – analyze the rendered images
- Object space method – analyze the mesh

Image Space Method

- Analyze the depth buffer – look for depth discontinuity using edge detector



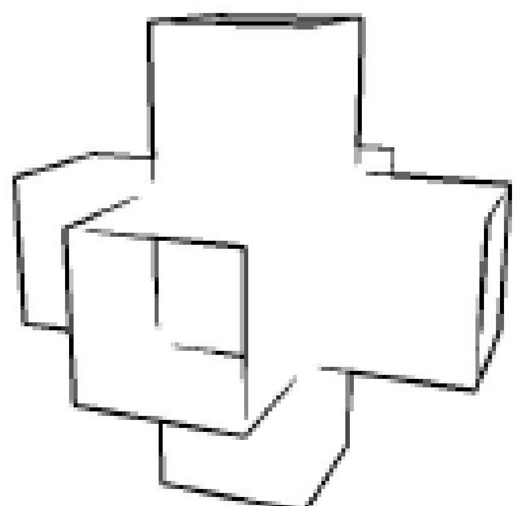
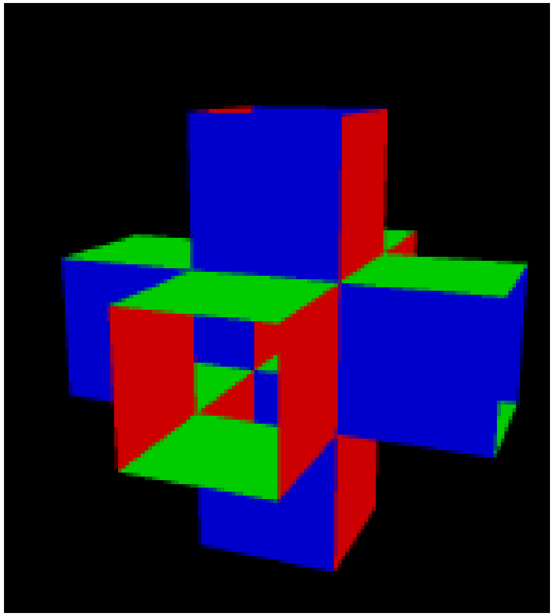


Image Space Method

- Analyze the Normal Map – convert surface normal (x,y,z) to (R,G,B) and then detect the color discontinuity



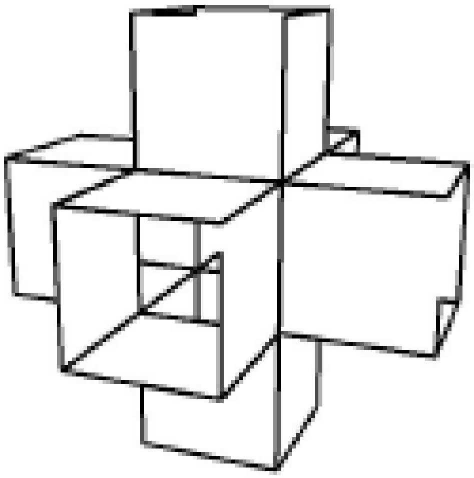
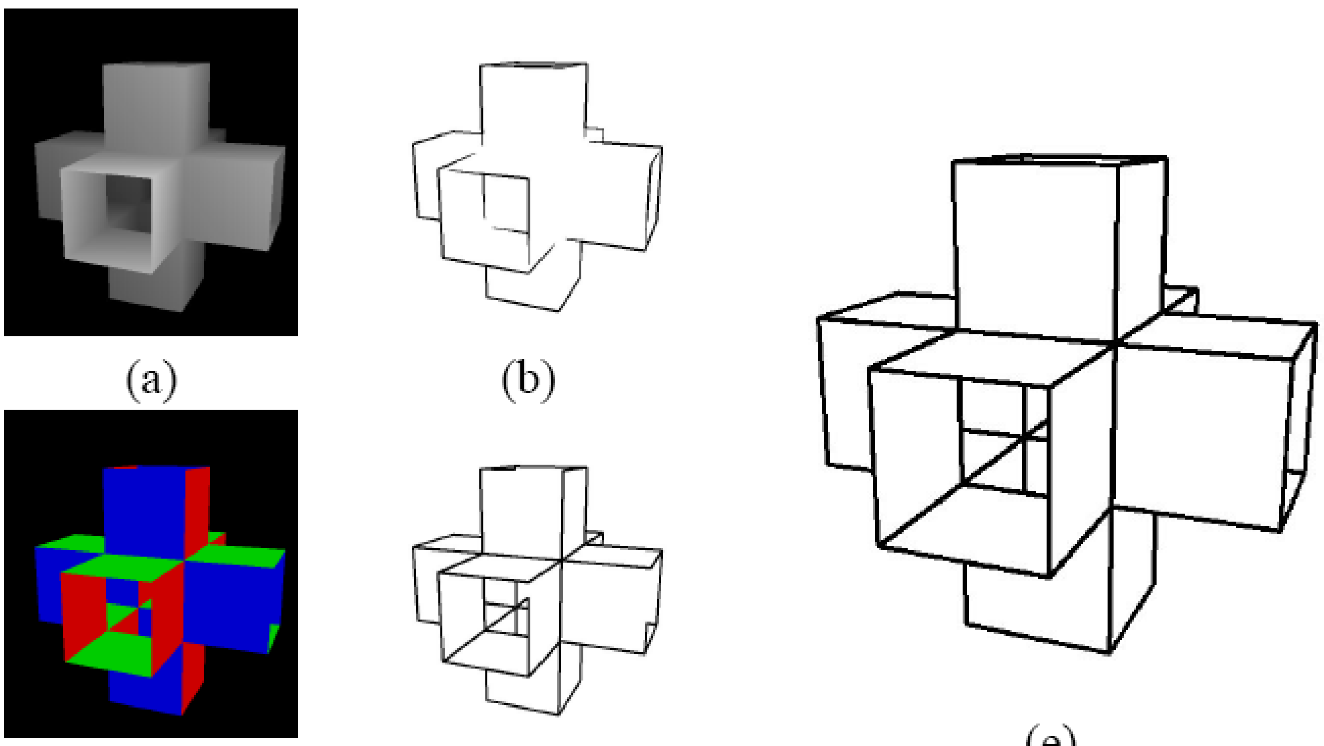


Image Space Method

- Better result can be obtained if both edges are combined



Edge Detector

- Discontinuity in depth map or normal map can be detected using edge detector

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

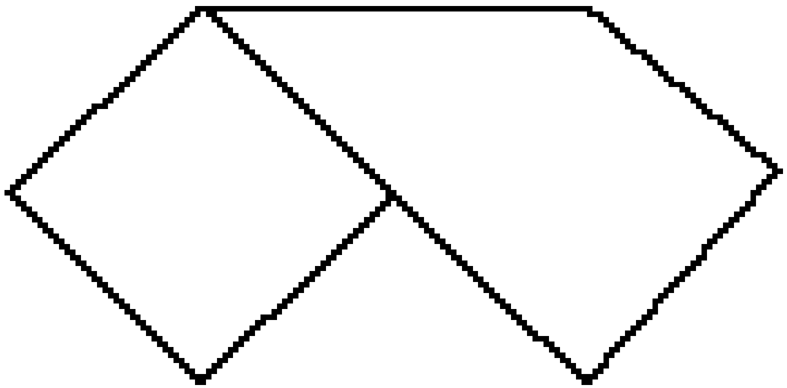
$Ix(x,y) = I(x,y) \otimes Sx$; $Iy(x,y) = I(x,y) \otimes Sy$

$IM = \mathrm{sqrt}\left(Ix(x,y)^2 + Iy(x,y)^2 \right)$

Get edge by thresholding IM

Image Space Method Problem

- For a folded piece of a paper, the edge cannot be detected

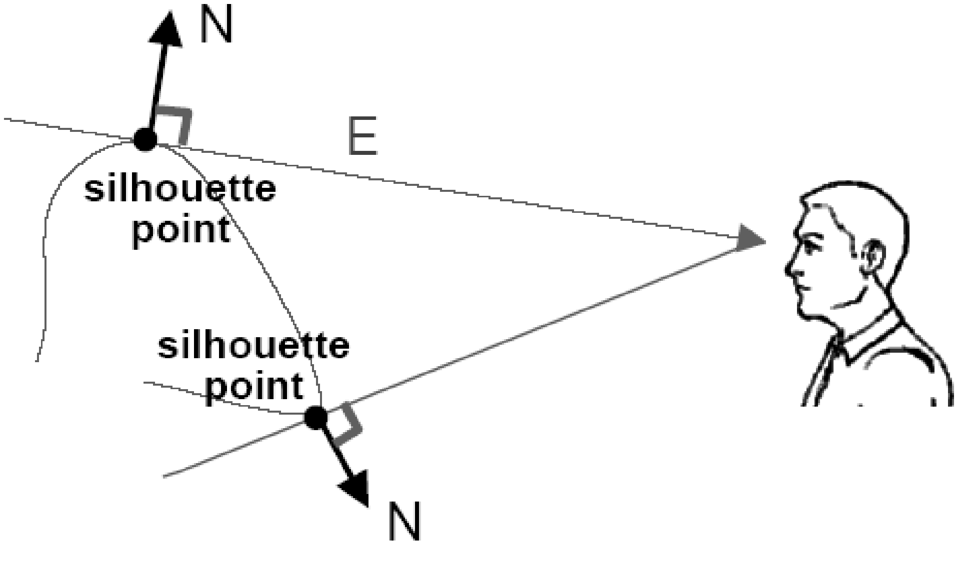


Object Space Method

- Mainly used to detect silhouettes and creases
  - Silhouettes: edges that connect front and back faces
  - Creases: A discontinuity on an otherwise smooth edges

Silhouette

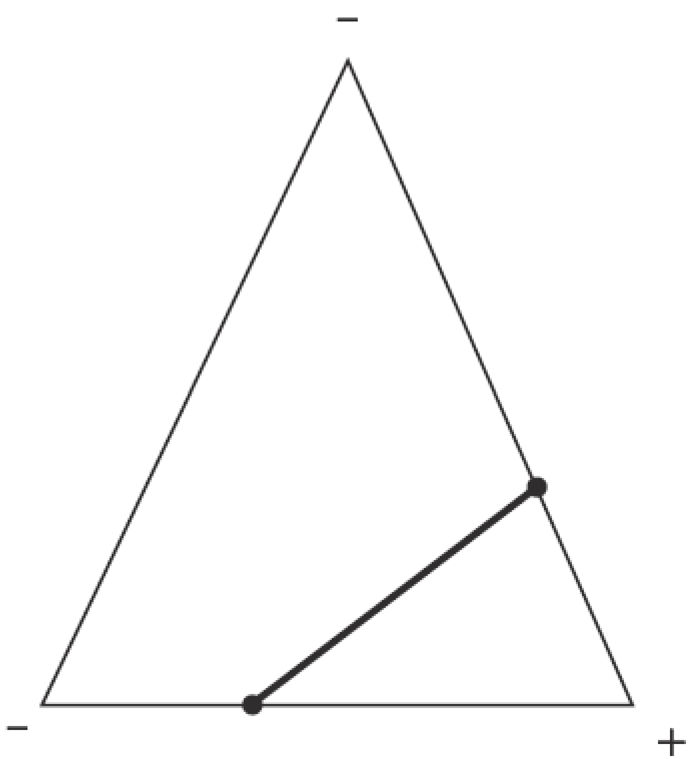
- For a smooth surface, a silhouette can be defined as:
  - N. (X-E) = 0 ; N: normal, X: silhouette point; E: camera

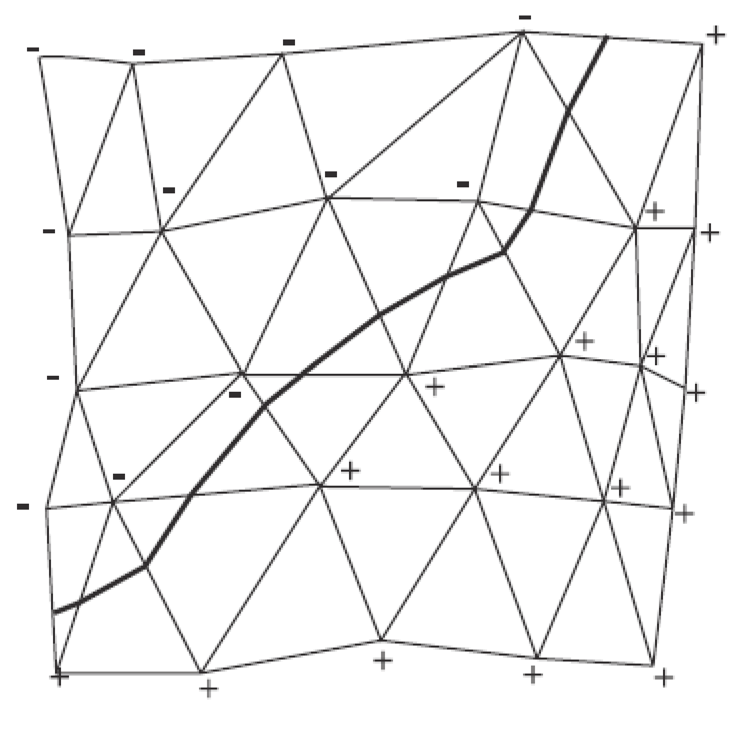


Software Method

- Detect Silhouettes from all triangle edges
- For each vertex, evaluate:
  - d = n. (x-e) / |n|*|(x-e)|
  - s = + if d >0; else –
- Find s = 0 along face edges

Software Method





Hardware Method

- Use OpenGL to draw silhouette edges (no explicit search)
- Pseudo code (a three pass method)

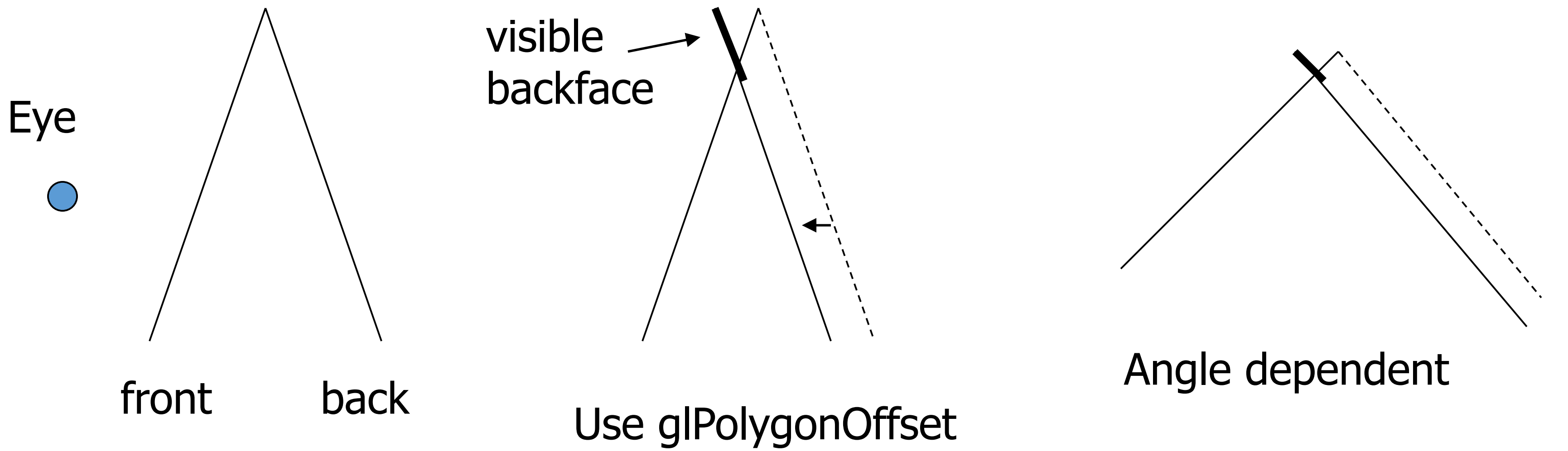
  draw shaded front faces

  draw front faces in line mode, set stencil

  draw back farces in line mode at where stencil was set; decrementing stencil

Hardware Method

- Reduce to 2 pass by push the backface forward (z bias)



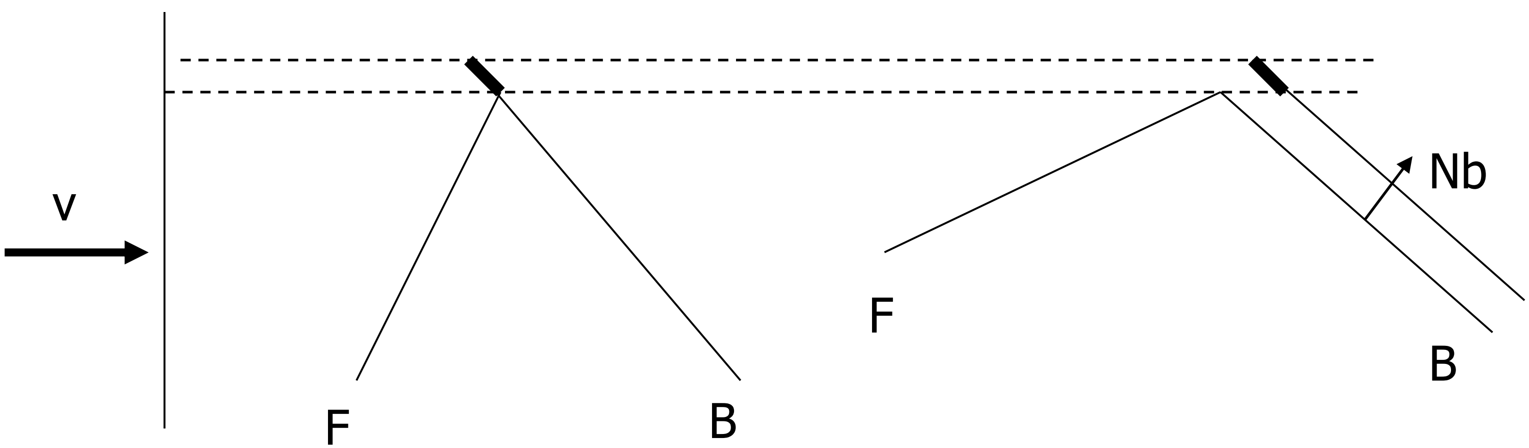
Issues of the Previous Method

- Non-uniform z resolution needs to be taken care of – translate by k*z;
  - K: a scaling factor, z: the polygon distance
- The width of the line width will depend on the orientation of the back-facing polygon and front-facing polygon
- Raskar and Cohen – fatten the back-facing polygons

Raskar and Cohen's Fix

- The back-facing polygon edge is pushed outwards

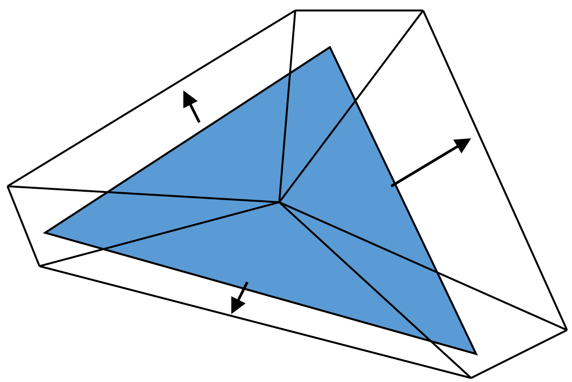
By Offset = K * z/V.Nb



The distance to push only depends on the orientation of back-facing polygon

Raskar and Cohen’s Fix

- In fact, each of the polygon edges needs to be pushed by a different amount: z*sin(α)/V.Nb; where cos(α) = v.e, e is the polygon edge vector



Charcoal Effect

- Tessellate the polygon to smaller pieces
- Also fatten front-facing polygons with $0<N.V<0.1$
- Assign color $I = (1+V.N)/3$

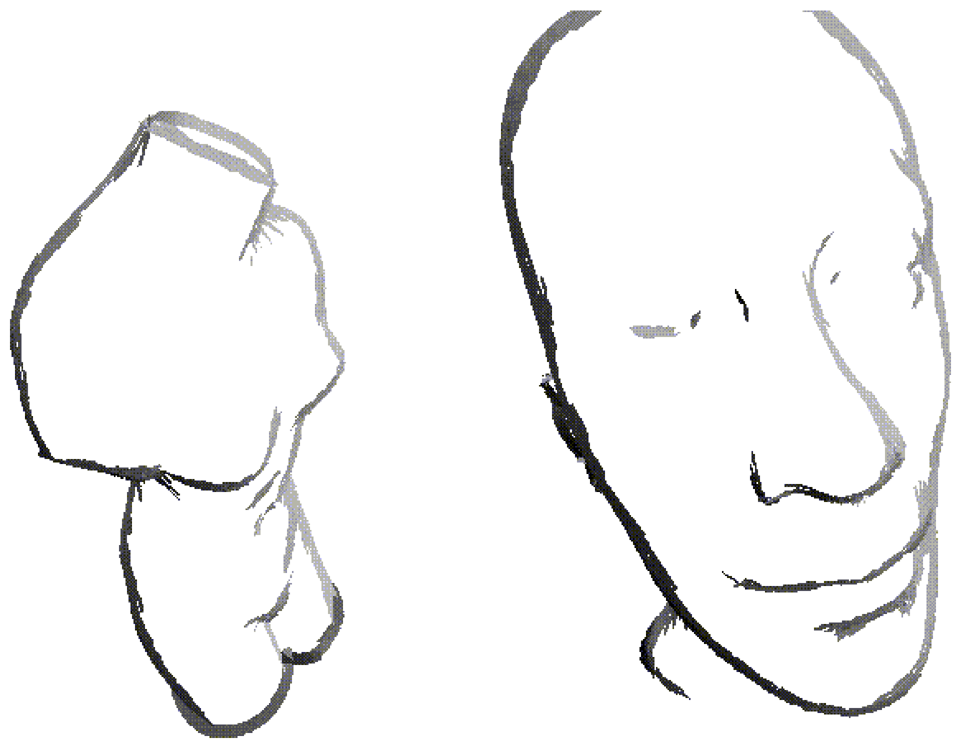
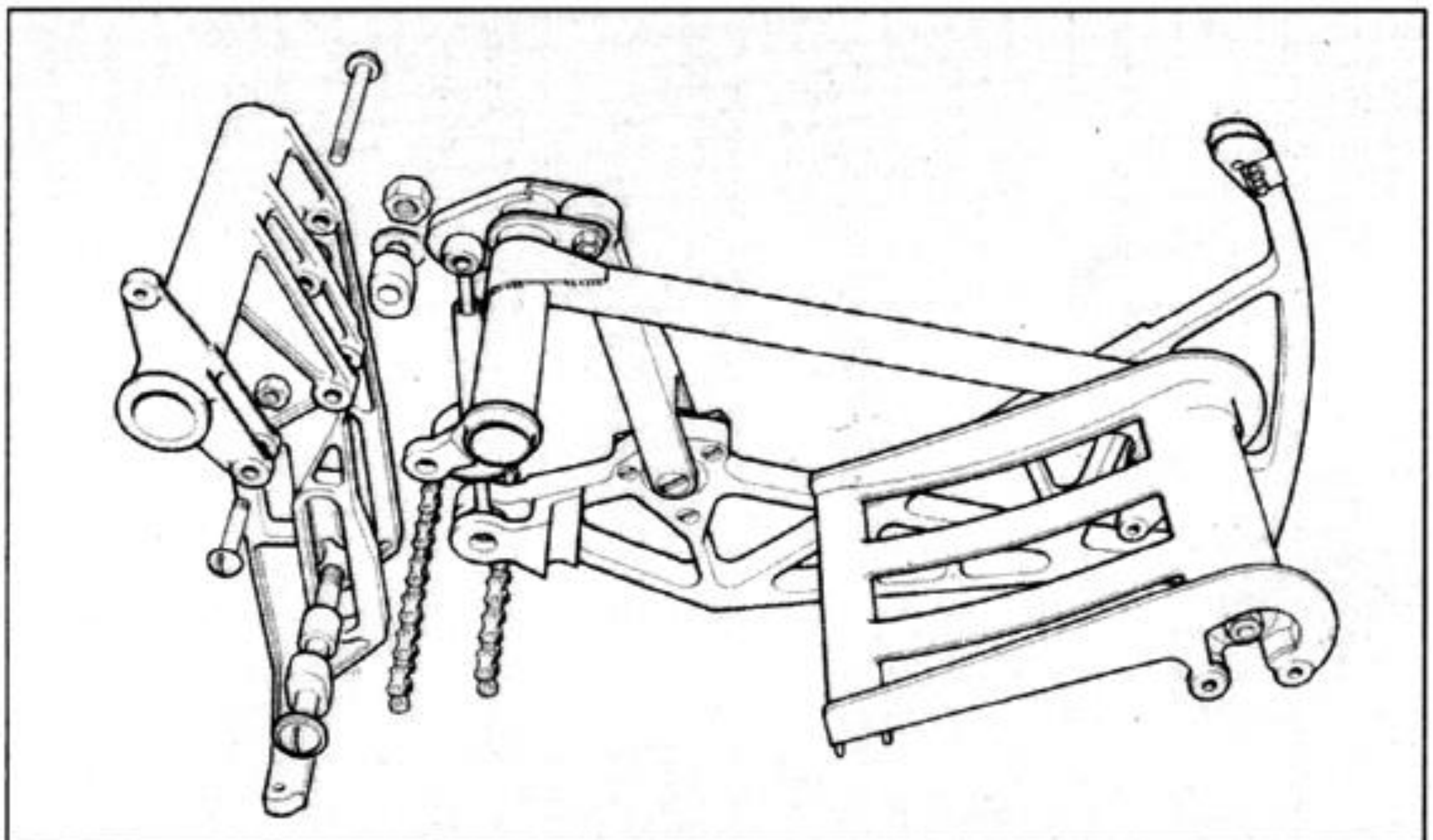


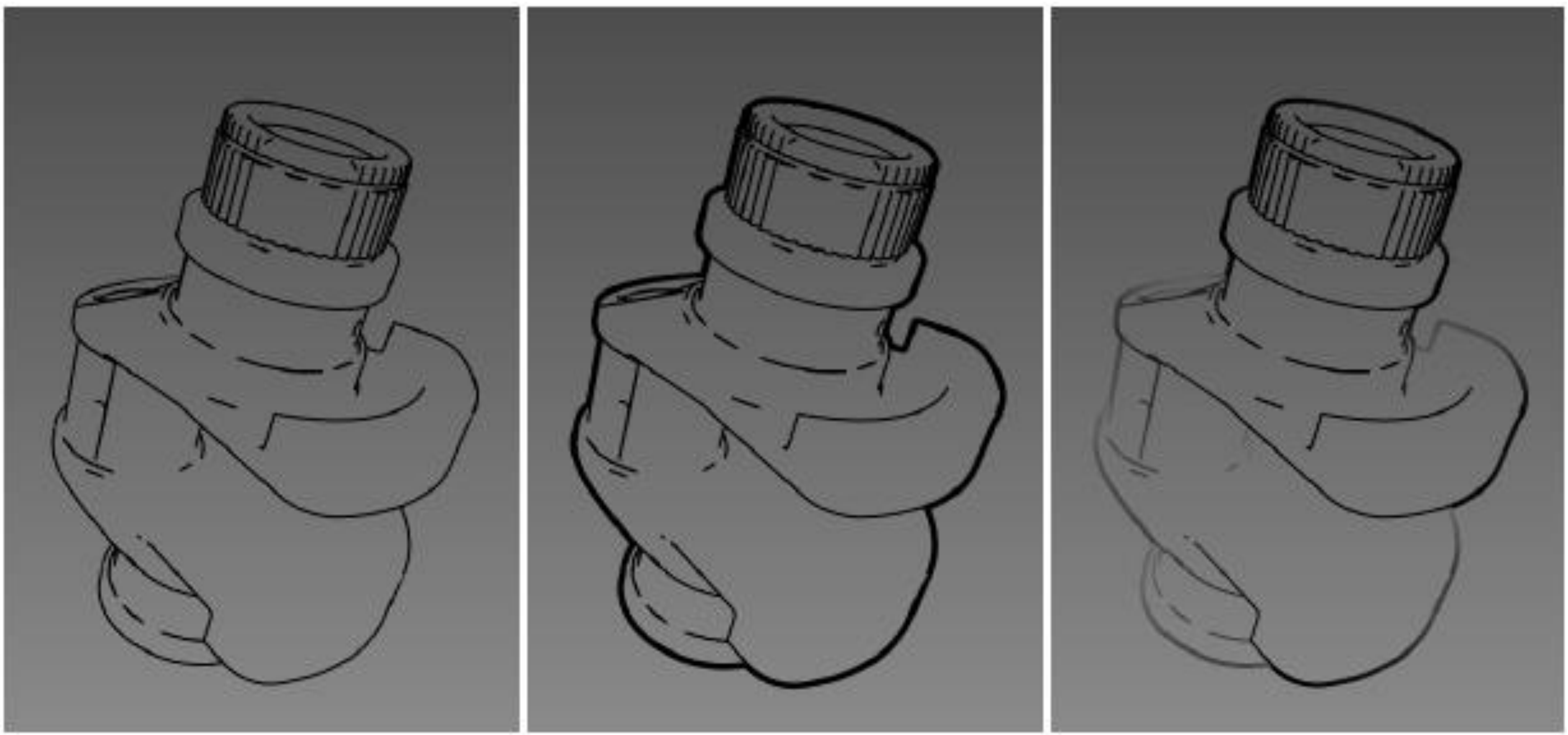
Illustration Example



Line Weight

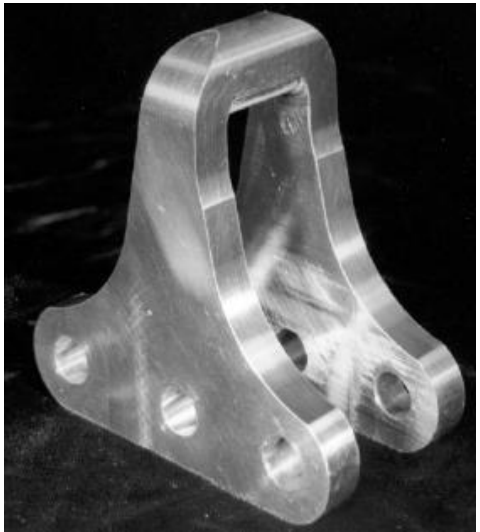
- Some possible choices:
  - Single line weight used throughout the image
  - Two line weights, with heavier describing the outer edges (boundary and silhouette)
  - Various light weight along a single line, emphasizing perspective effect (heavy lines in the foreground, tapering toward the farther part of the object)

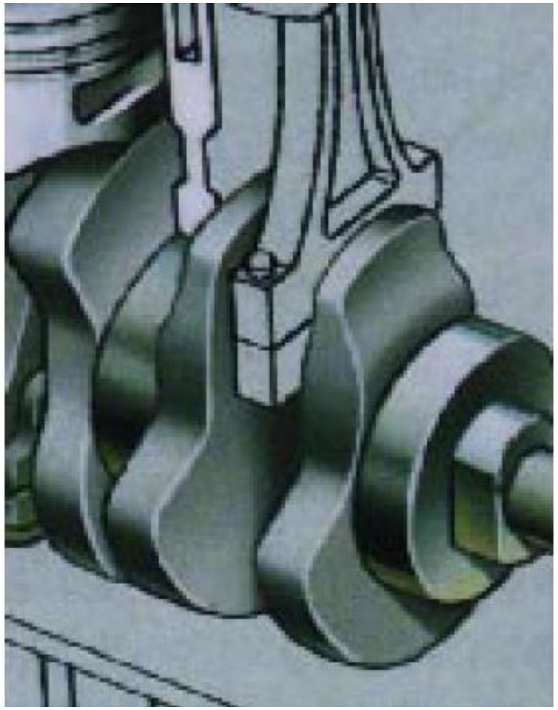
Line Weight

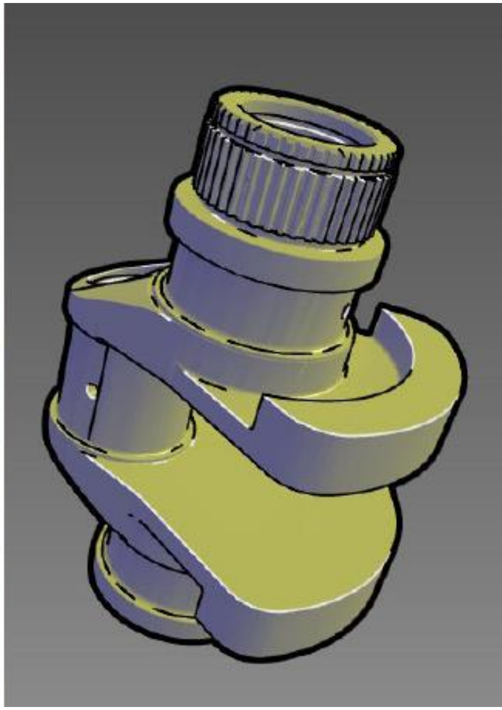


Line Color

- Attempt to incorporate shading
- Interior lines can be drawn in white, simulating highlight

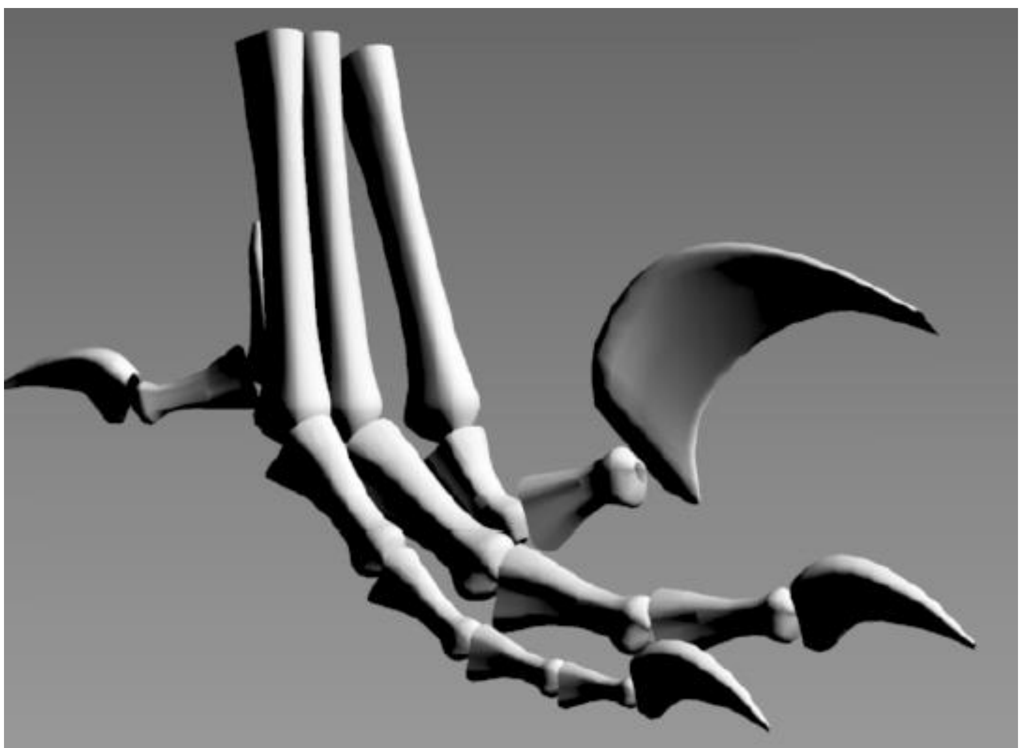






Tone Shading

- The standard Phong Shading model is not always satisfactory

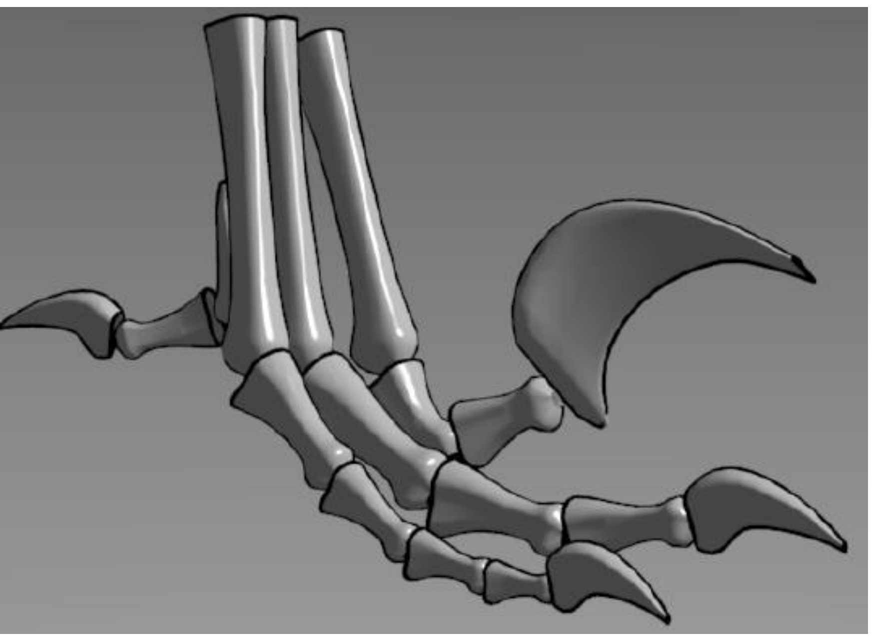


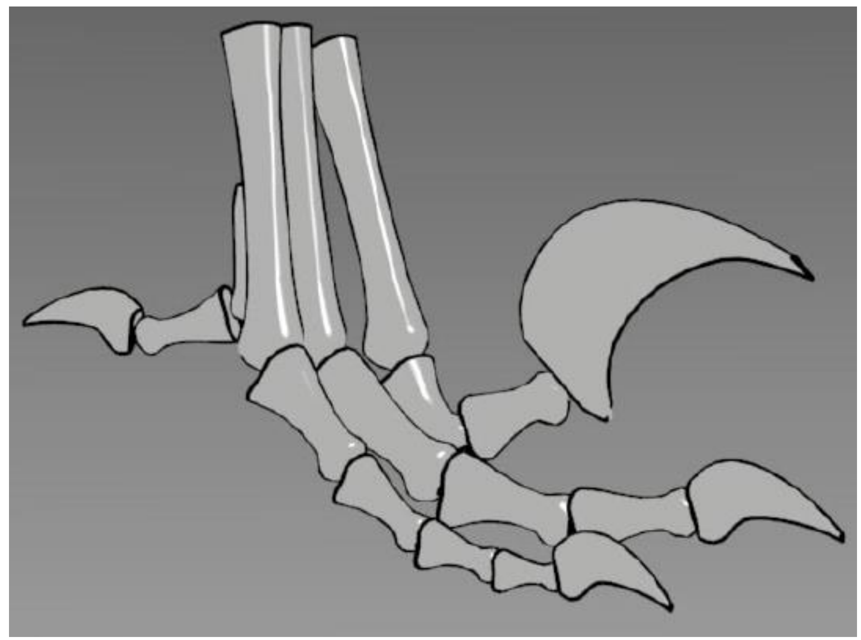
Problems in regions where
N.L < 0

-Only Ambient Colors are seen
-Difficult to deduce shapes
-Object outlines cannot be seen

Two ad hoc Solutions

- Hand-tuned ambient color
- Just highlights and edge lines





Ambient is only a constant ☹

Not enough surface detail ☹

Effective Shading Model Needed

- Shading Model is insufficient
- Lost shape information
  - Especially in the areas of subtle curvature (small claws above)
- Not automatic, lots of hand-tuning

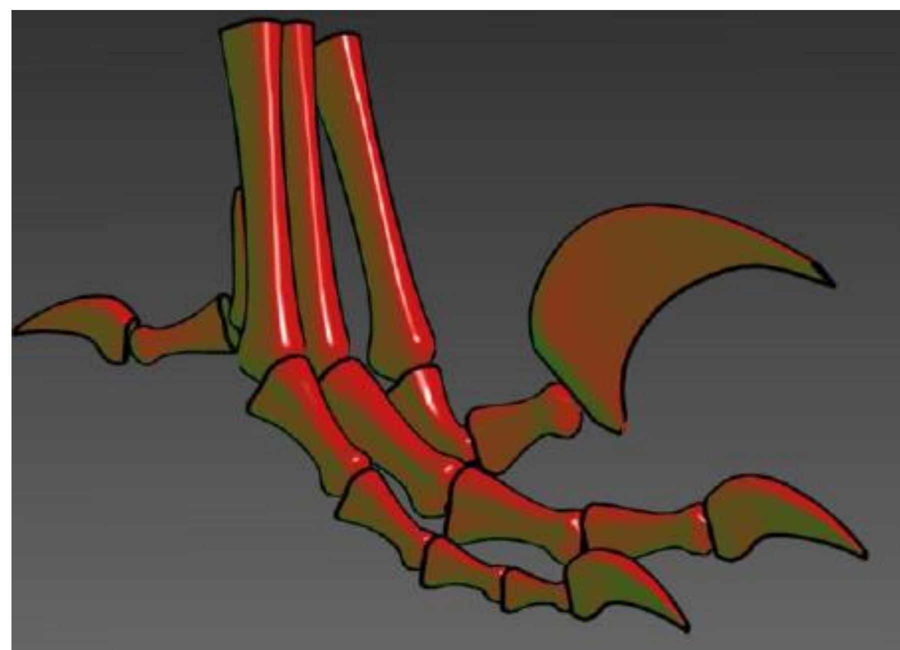
Tone Shading Goals

- To include shading in an image with back edge lines and white highlights visible
  - Use a compressed dynamic range for shading
  - Use color visually distinct from black and white

Reduce Dynamic Range

- One way to compress dynamic color range is to use colors of different tones
  - Add gray to a color to generate different tones

Unnatural color
Lack of luminance difference



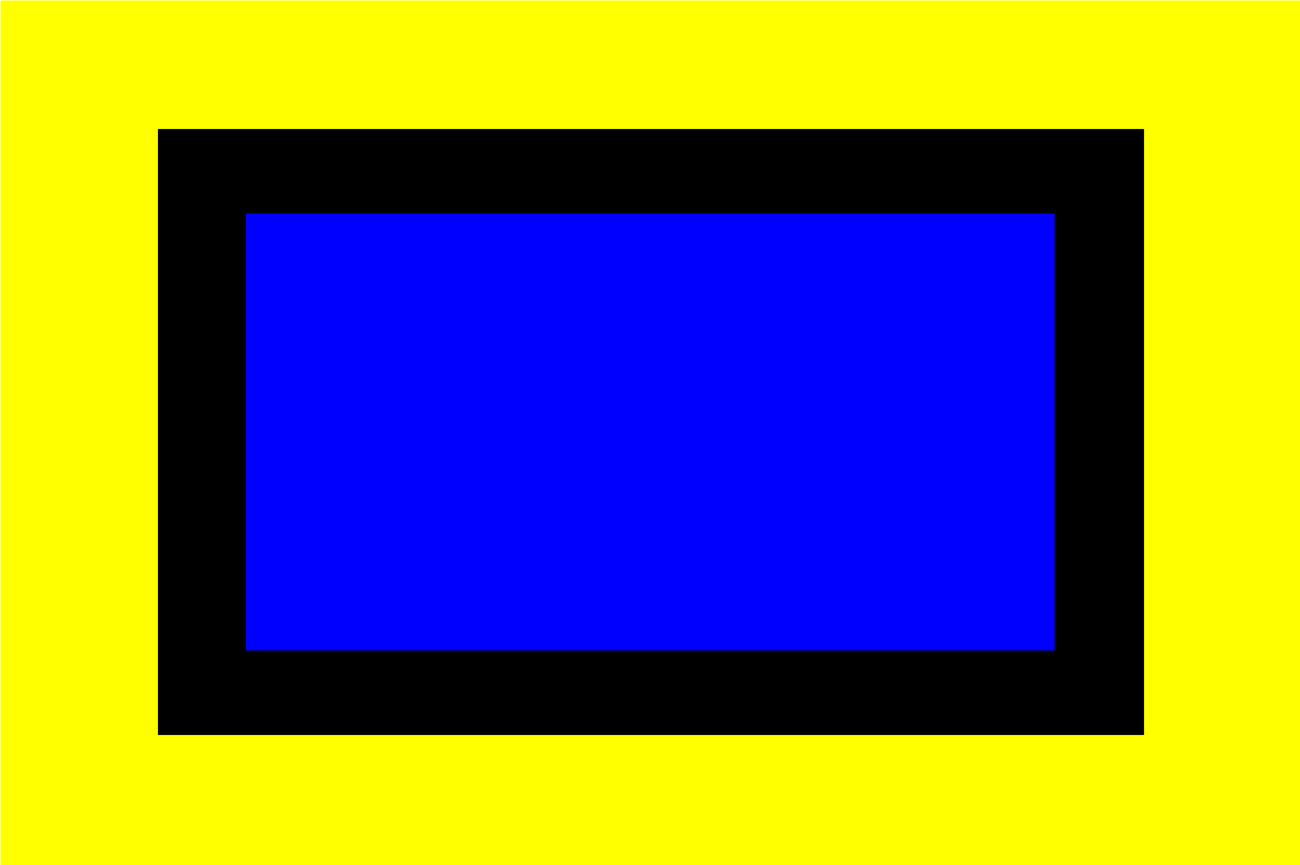
Create Undertone

- To further differentiate different surface orientations, we can use cool to warm color undertones
- Cool colors – blue, violet, green
- Warm colors – red, orange, yellow

warm

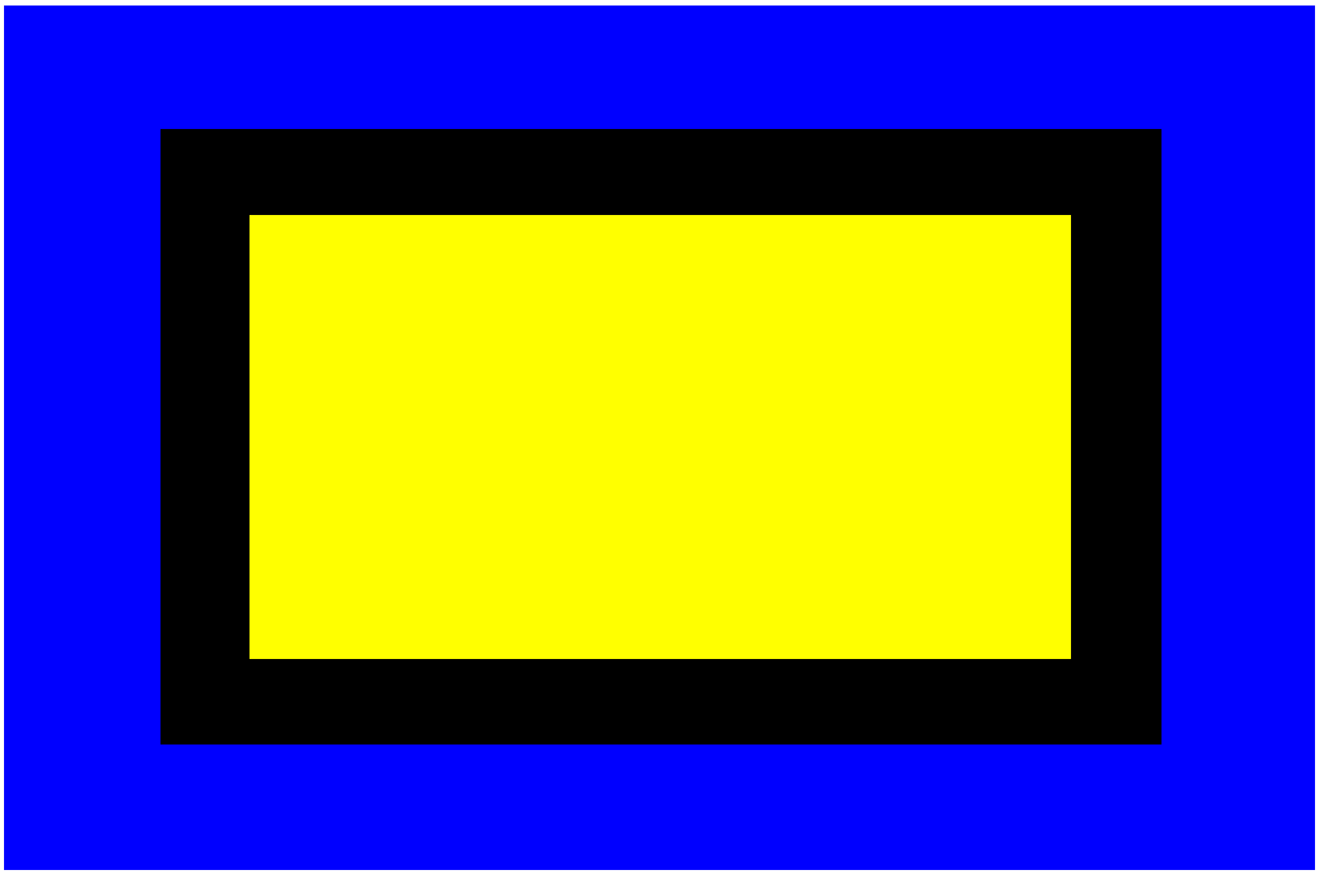
cold

Test Your Perception



Which color (yellow or blue) seems closer?

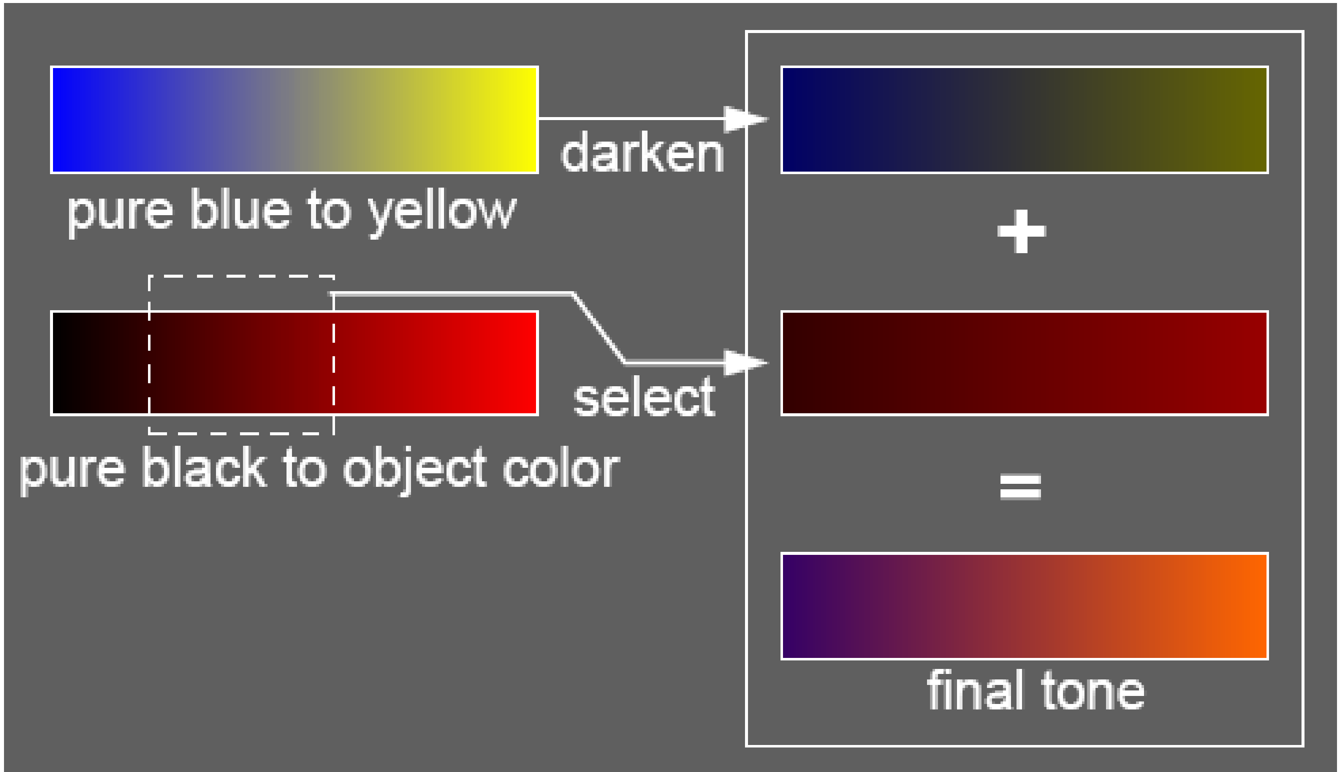
Test Your Perception



What about now?

Blend Tone and Undertone

- Add warm-to-cool undertone to a red object



Use Warm-to-cool Undertone

- We can modify the diffuse Phong Lighting Model (Blend cool and warm color)

$I = (1 + L.N)/2 * K_{cool} + (1- (1+L.N/2)) * K_{warm}$

The Light vector should be place in perpendicular to the gaze direction (usually place at up and to the right)

Tone Shading Equation

- $K_{cool} = K_{blue} + \alpha K_d$ (undertone and tone)
- $K_{warm} = K_{yellow} + \beta K_d$ (undertone and tone)

$K_{blue} = (0,0,b)$ b in [0,1]

$K_{yellow} = (\gamma,\gamma,0)$ γ in [0,1]

α and β are user-specified parameters

Kd is the object diffuse color