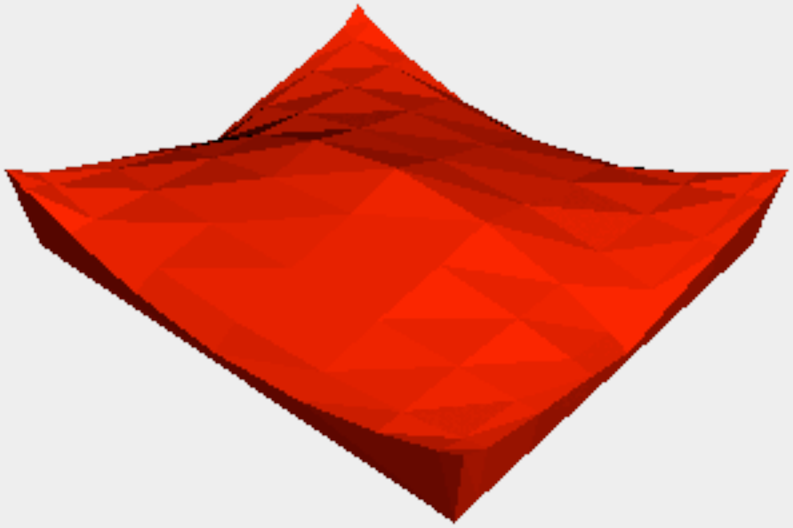
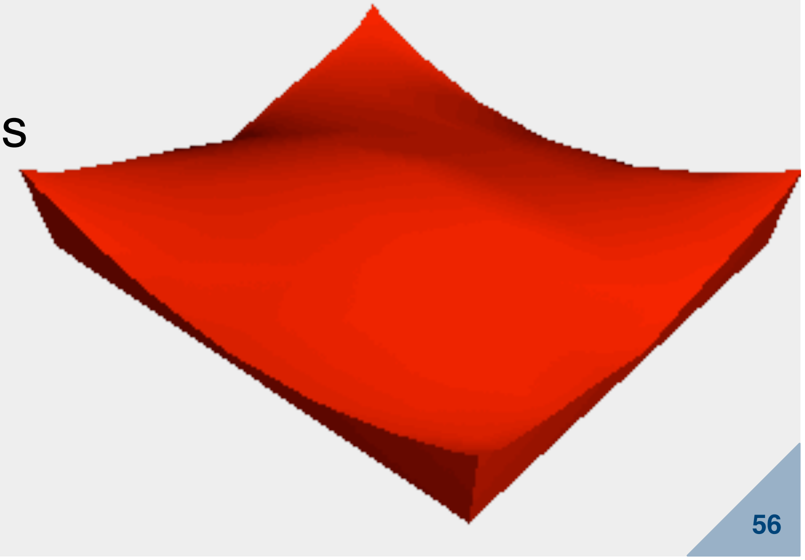
Flat shading

- Compute shading at a representative point and apply to whole polygon
  - OpenGL uses one of the vertices
- Advantages:
  - Fast - one shading computation per polygon, fill entire polygon with same color
- Disadvantages:
  - Inaccurate
  - Faceted

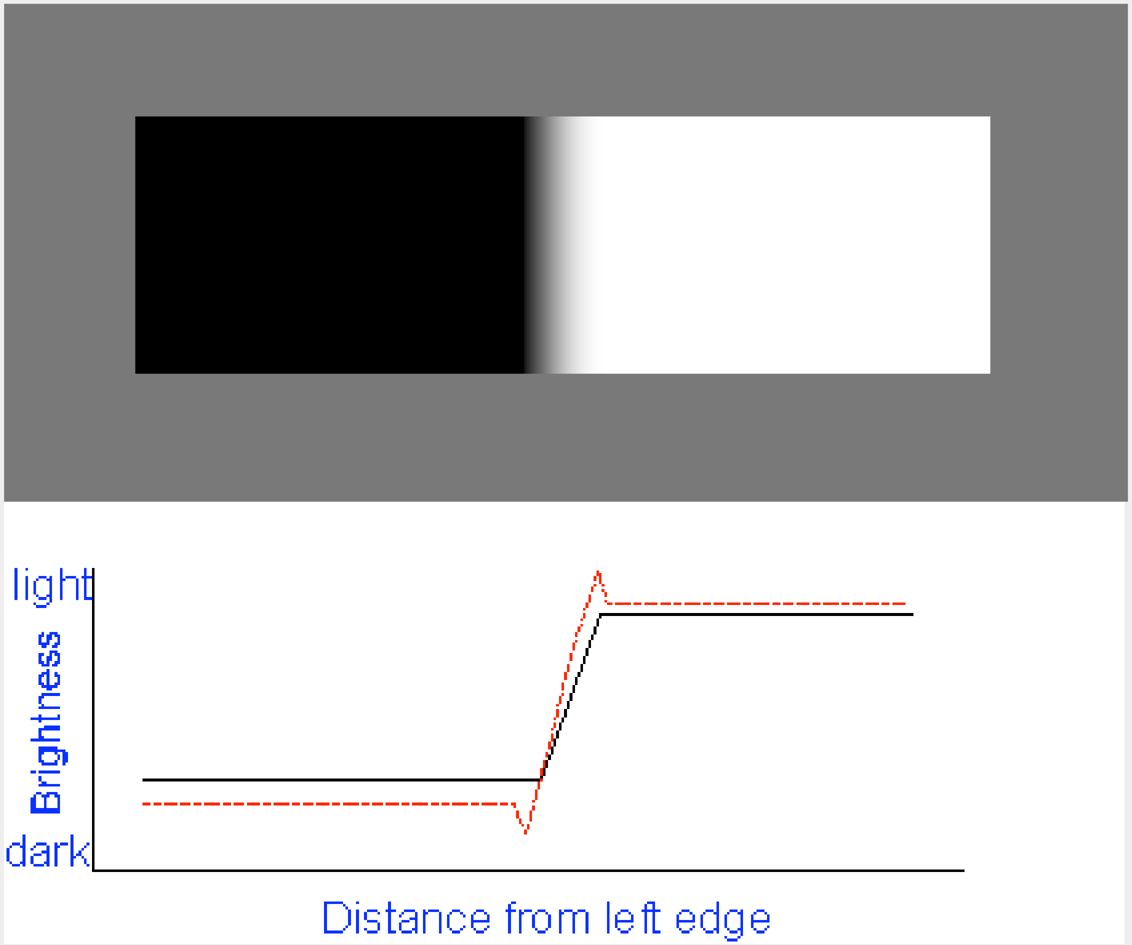


Gouraud Shading

- Light each vertex with its own location and normal
  - Result is a color at each vertex
- Interpolate the colors across each triangle
- Default mode for most hardware
- Advantages:
  - Fast: incremental calculations when rasterizing
  - Much smoother - use one normal per shared vertex to get continuity between faces
- Disadvantages:
  - Don’t get smooth specular highlights
  - C^1-discontinuities in color cause *Mach bands*: perceptual illusions of edges



Mach bands

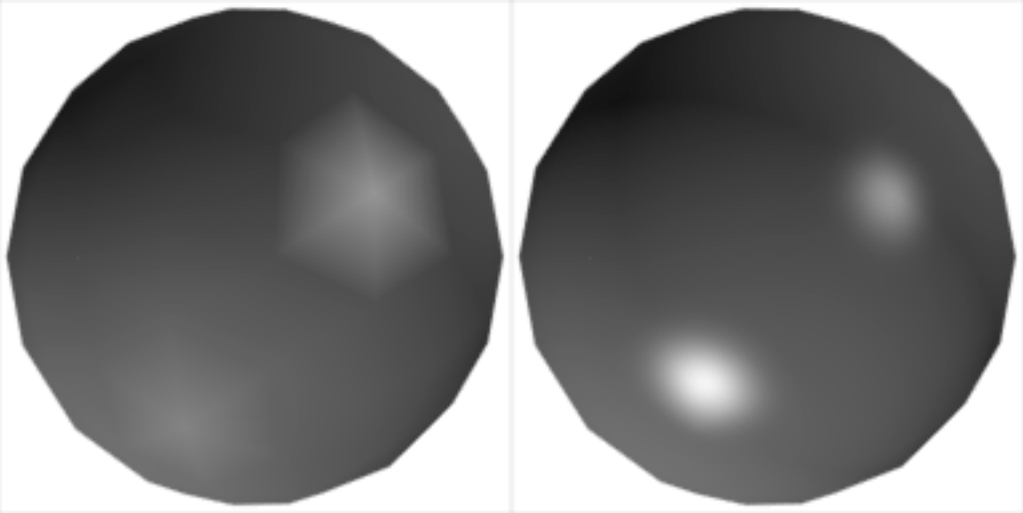


Phong Shading

- Want to recompute lighting at every pixel
  - Need normal vector at each pixel
- Interpolate vertex normals
  - Interpolate the normals while scan-converting
  - Typically, bilinearly interpolate x,y,z; then renormalize
  - Also interpolate or transform to get position in world/camera space
- Known as *Phong Shading* or *Phong Interpolation*
  - Not to be confused with Phong Lighting Model
  - (though both are often used at the same time)
- Advantages:
  - More accurate
  - Better images
- Disadvantages:
  - Slow
  - Still not completely accurate
- Modern GPUs can perform Phong shading via pixel shaders

Gouraud vs. Phong shading

- Gouraud misses specular hilights

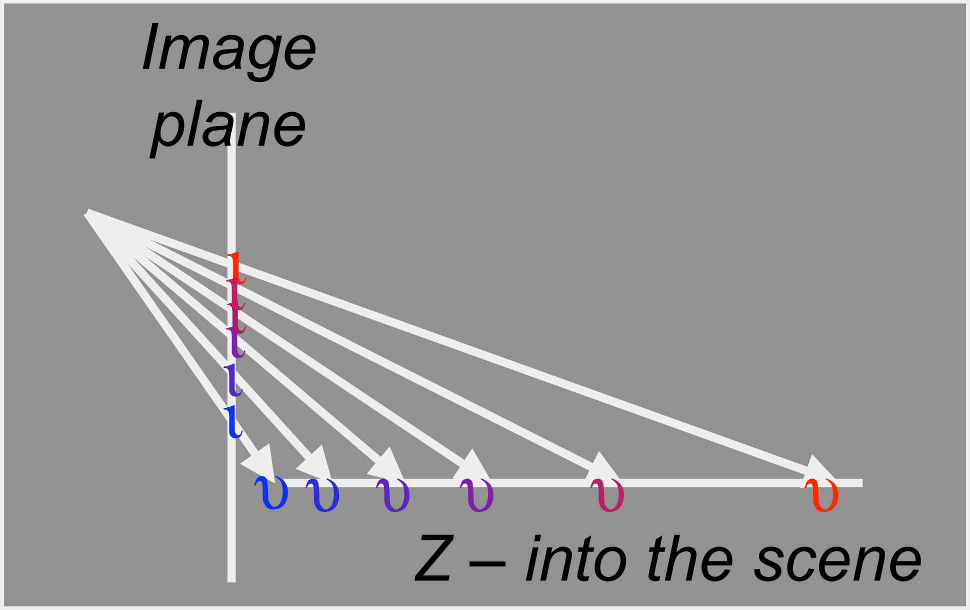


Gouraud *Phong*

- solution:
  - tessellate more finely: use smaller triangles
  - that would help with the polygonal artifacts on the silhouettes too

Note: Perspective Distortion

- Linear interpolation in screen space doesn't align with linear interpolation in world space



- Solutions:
  - Do hyperbolic interpolation (see Buss)
  - Break up into smaller triangles

Lighting vs. Shading

- *Lighting*: compute the result of light illuminating surfaces
- *Shading*: assign colors to pixels

- For photorealistic rendering:
  - in principle, shading==lighting: perform lighting at every pixel

- In practice:
  - may take shortcuts
  - may include non-lighting effects
    - fog
    - illustration
    - cartoon shading

Vertex Lighting

- Each vertex goes through lighting process
- Lighting computation determines final color at the vertex
  - Based on initial “unlit” vertex color
  - Based on lights in the scene
  - Based on material properties of the surface
  - Based on surface normal $\vec{\mathbf{n}}$
- Interpolate colors using Gouraud shading
- (Same lighting computation for per-pixel lighting)