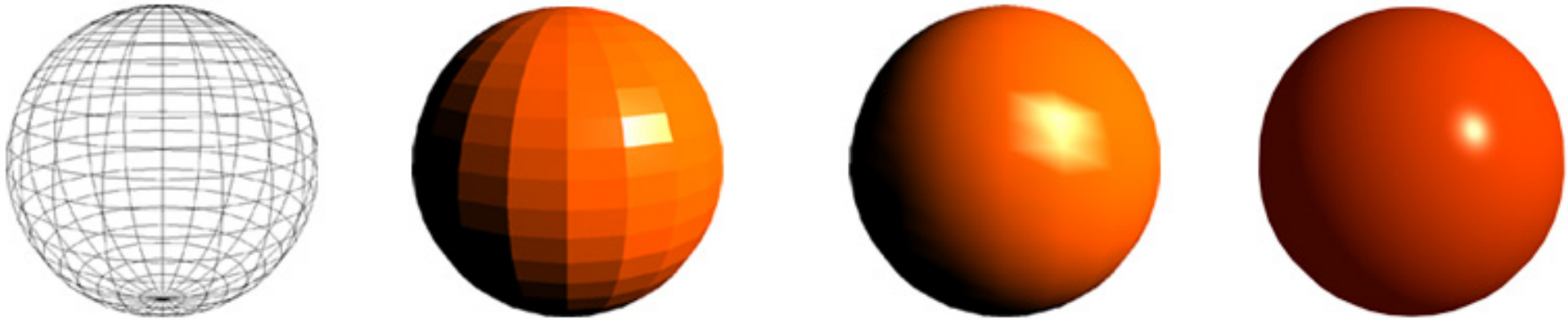


Shading and Lighting

also today: GDC Vault advanced lighting

CS 4300/5310

Computer Graphics

ANNOUNCEMENTS

Upcoming Deadlines

- HW3: Particle Systems
  - March 2nd

QUICK REVIEW

Review: The Graphics Pipeline

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Clipping
- Projection to 2D space
- Rasterization
- Pixel Shading
- Frame Buffer

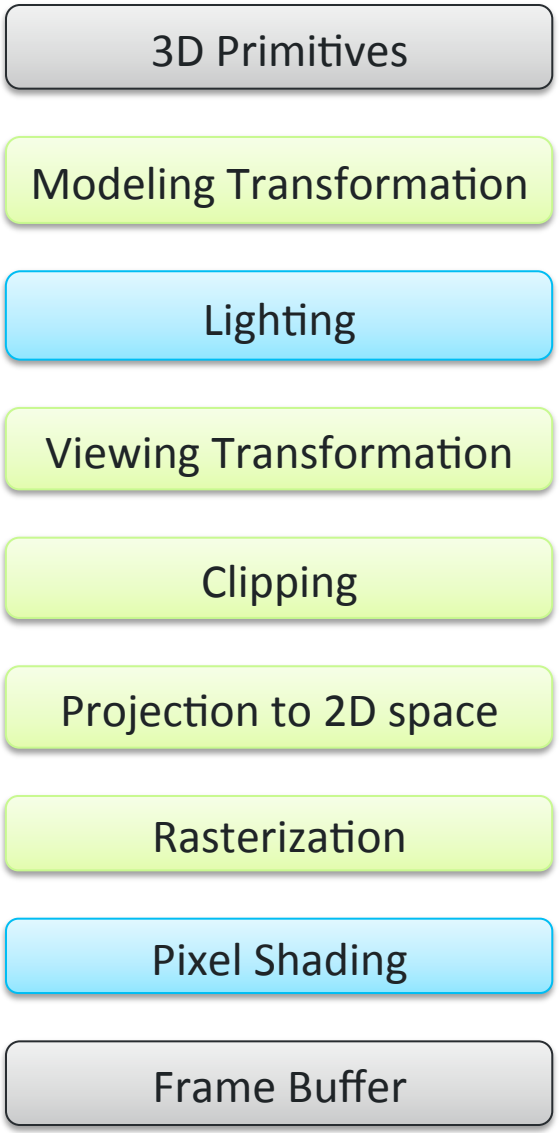
What happens in each of these stages?

Review: Lighting

- What kinds of lights are there?

- Why do we use each one?

Lighting & Shading



- Shading
  - Vertex lighting
  - Color interpolation
  - Texturing
- Effects
  - Bump mapping
  - Displacement mapping
  - Shadow mapping

part one

LIGHTING & SHADING

Kinds of Lights

- Ambient
- Point
- Directional
- Spot

Calculating Color Per Vertex

- How can we do this?

Calculating Color Per Vertex

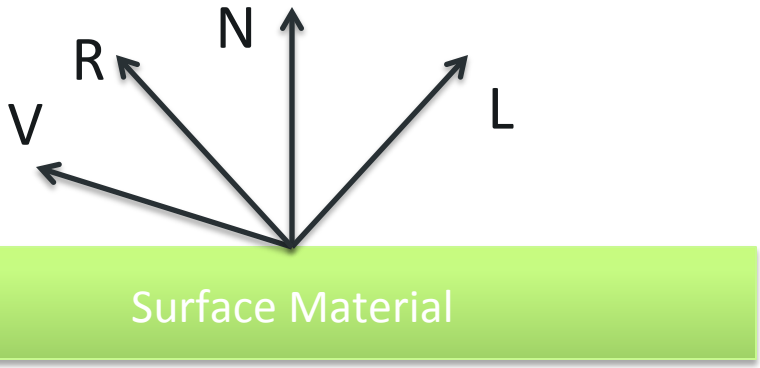
- How can we do this?

- **The same way we calculate color for the raytracer!**

Ambient Light

- There is usually only one ambient light in a scene

I

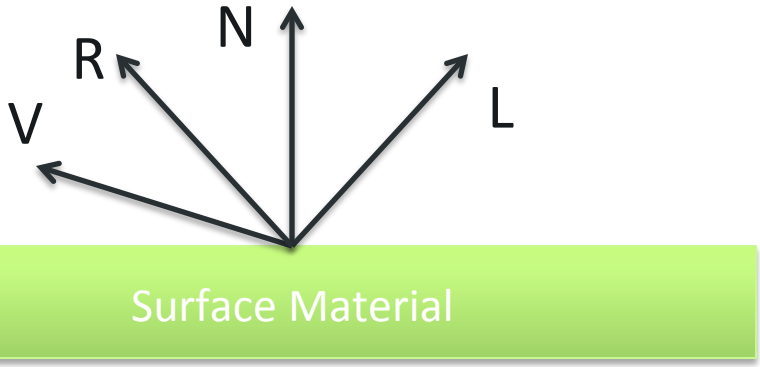


color = $M_A I_A$

Lambertian Shading

- Diffuse shading: matte color
  - Assume material reflects light evenly in all directions

I

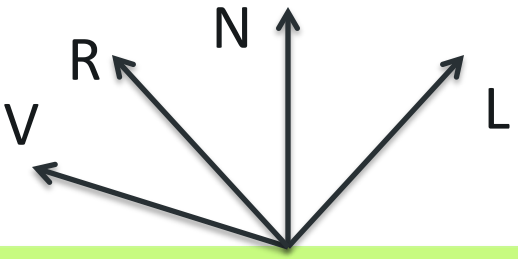


color = $M_A I_A + \Sigma_L M_D (N \bullet L) I_L$

Blinn-Phong Shading

- Specular light: idealized reflection
  - Depends on how much is seen by viewer

I



$$\text{color} = M_A I_A + \Sigma_L (M_D (N \bullet L) I_L + \mathbf{M_S (V \bullet R)^n I_L})$$

But what about the pixels?

- We know the color at every vertex of an object

- **What are our options for the pixels?**

Flat Shading

Pre-Rasterization

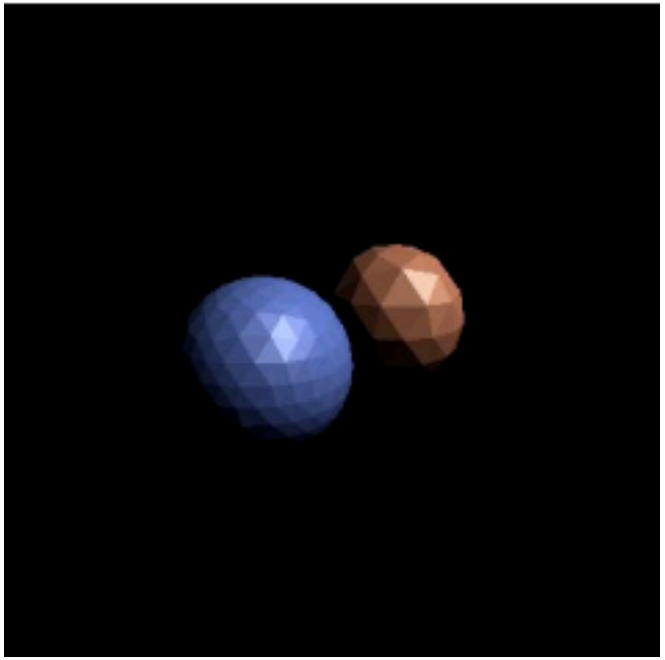
- transform position and normal (object to eye space)
- compute shaded color per triangle using normal
- transform position (eye to screen space)

Rasterizer

- interpolated parameters: z’ (screen z)
- pass through color

Fragment stage

- write to color planes only if interpolated z’ < current z



Gouraud Shading

Pre-Rasterization

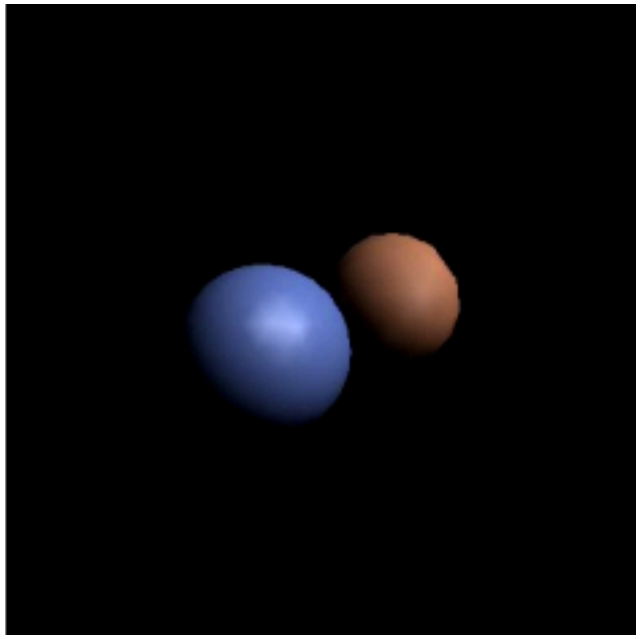
- transform position and normal (object to eye space)
- compute shaded color per vertex
- transform position (eye to screen space)

Rasterizer

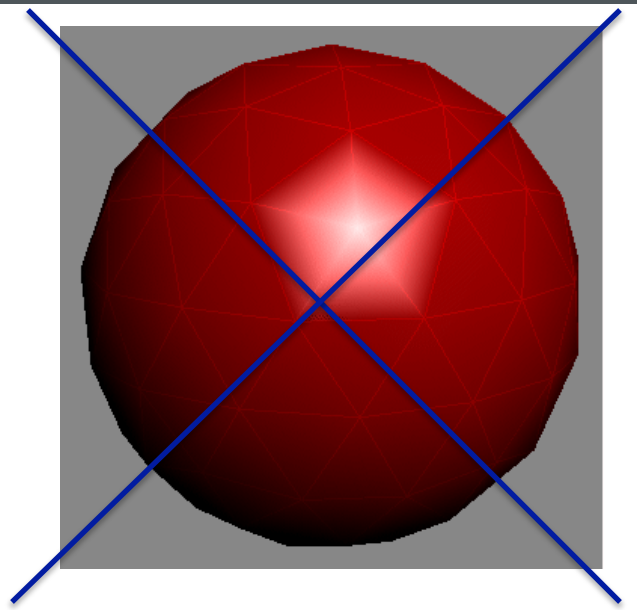
- Interpolated parameters: z’ (screen z)
- Interpolated r, g, b color

Fragment stage

- write to color planes only if interpolated z’ < current z



Phong Shading (Per-Pixel Shading)

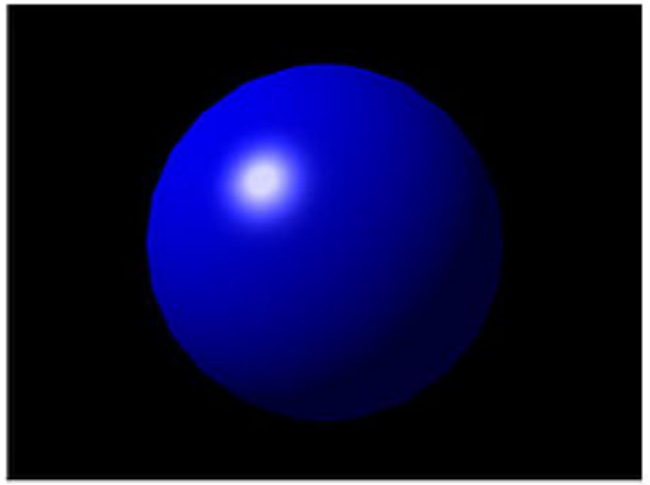


Pre-Rasterization

- transform position and normal (object to eye space)
- transform position (eye to screen space)
- Pass through color

Rasterizer

- Interpolated parameters: z' (screen z)
- Interpolated r, g, b and surface normal (x, y, z)



Fragment stage

- Compute shading using interpolated color & normals
- write to color planes only if interpolated z' < current z

Lighting color is computed per pixel rather than per vertex

Raytracing vs. Rasterization

- Shading is very similar between these two approaches to rendering
- What's missing so far?
- Why is rasterization considered faster?

Limitations

- What effects from the real world can we not capture with this model?

- What special effects (artistic effects) can we not capture?

ADVANCED LIGHTING