Shader Lighting, Textures and Shadows

CSCI 4239/5239
Advanced Computer Graphics
Spring 2025

Shader Lighting

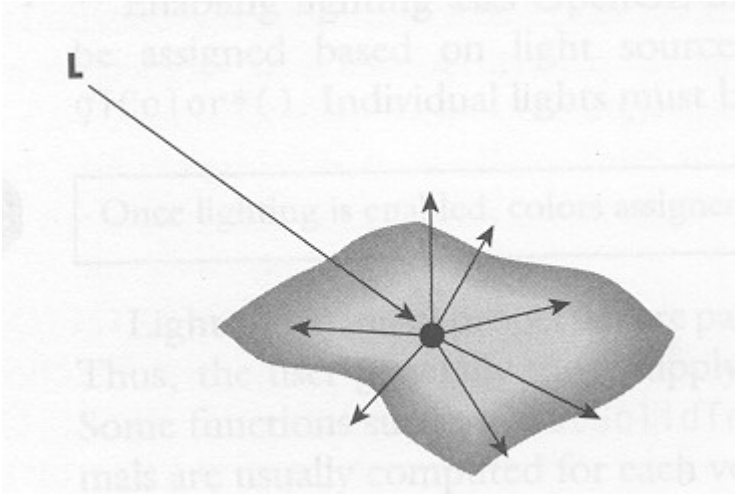
- Ultimate flexibility
  - Lighting method
    - Phong reflection
    - Blinn-Phong reflection
  - Lighting
    - Per vertex with Gouraud shading
    - Per pixel lighting
  - Special effects
    - High Dynamic Range lighting
- Ultimate responsibility
  - Nothing happens automatically

OpenGL Lighting Components

- $C = M_E + M_A(C_A+C_G) + (N \bullet L)M_D C_D + (N \bullet H)^S M_S C_S$
- C_X are light components
- M_X are material components
- Components
  - Emission
  - Ambient (also Global Ambient)
  - Diffuse
  - Specular
- Calculated for each light, vertex, RGBA
- Assumes values in the range 0-1

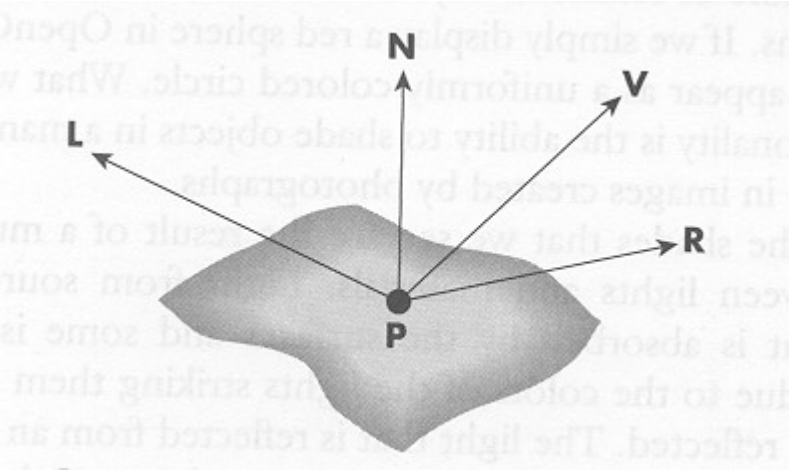
Diffuse Reflections

- Diffuse light scatters in all directions
  - Lambertian reflection
- Intensity depends on cosine of the angle of incidence
- Intensity $(N \bullet L) M C_D$



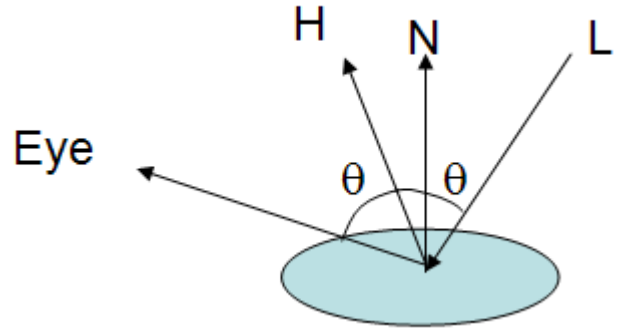
Phong Reflection Model

- *L* light source
- *N* normal vector for surface
- *R* reflected light
  - $R = 2(L \bullet N)N - L$
- *V* viewer (eye)
- Intensity $(V \bullet R)^S MC$
  - *S* shininess
  - *M* material reflection coefficient
  - *C* color if light source
- Calculated independently for R,G,B



Blinn-Phong Reflection Model

- Also called modified Phong or Fast Phong
- Simpler and faster
- Half angle $H = L+V$ (renormalize)
- Intensity $(N \bullet H)^{s} MC$



Per Vertex Lighting

- Calculate lighting at vertex
- Linearly interpolate across polygon
  - This is often called Gouraud shading
  - Real Gouraud shading averages normals at vertexes and then interpolates
- Effort proportional to number of vertexes
- May miss important effects for large polygons

Per pixel lighting

- Calculate lighting at pixel
- Calculate ambient and emission by vertex
  - Set L,P,V,H for use in frag shader
- Calculate diffuse and specular by pixel
- Effort proportional to number of pixels

Shader Textures

- Pointer to texture
  - sampler2D *name*;
  - Points to current texture from glBindTexture()
- Extract pixel at vec2 texture coordinate *pos*
  - texture2D(*name*,*pos*);
- Different sampler/function for 1D,2D,LOD,...
- Returns vec4 (RGBA)

Assignment 3: Performance

- Explore the performance of different ways to do things in shaders
  - int vs. float
  - built-in functions vs. expressions
  - functions vs. inline
  - OS, hardware, etc dependencies
- Use lighting, textures, procedural textures, etc. to measure performance
  - Use fps(int) signal from Ex04
- Make sure VSYNC is disabled

Shadows in Computer Graphics

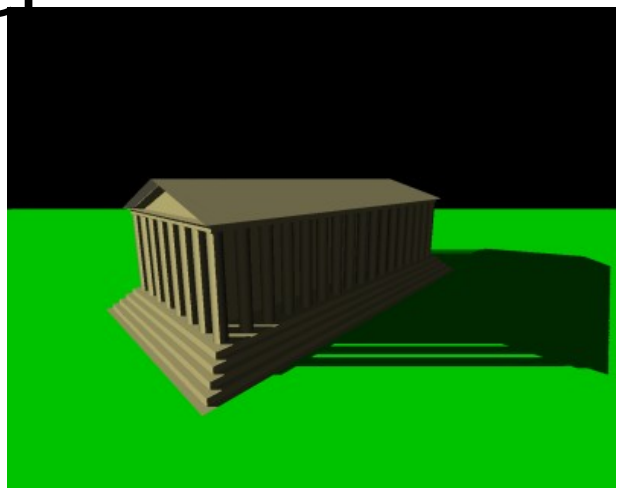
- Shadows are important to realism
  - Depth cues
  - Relative positions of objects
- Doesn't “just happen” when lighting is turned on
  - Nor is there a glEnable(GL_SHADOWS)
- Scene must be rendered 2-4 times
- Shader implementation can be efficient
  - Draw once every time the light or scene changes
  - Draw once for every eye position

Shadow Methods

- Planar Shadows (CSCI 5229 ex38)
  - Shadows on the floor only
- Shadow Volumes (CSCI 5229 ex39)
  - True shadow, very hard
  - Requires four passes (two are fast)
- Shadow Maps (CSCI 5229 ex40)
  - True shadows, depth in textures
  - Fast implementation via shader

Shadow Mapping

- Project with light as viewpoint
- Depth buffer from light
- Light/shadow determined just like visibility
  - Objects in light foremost in depth buffer
  - Objects in shadow depth obscured
- Requires second depth buffer
  - Store depth to texture
  - Compare R to texture
- In OpenGL extensions
- Used in *Toy Story* etc.



Shadow Map Shader

- Draw shadow map
  - Bind framebuffer to depth texture
  - Draw scene with eye at light to generate depths
  - Update if light or scene changes
- Draw scene
  - Generate texture coordinates with light PoV
  - Compare depth (R) with depth texture
    - R=depth means lit – light as normal
    - R>depth means shadowed – ambient light only
- Fast, Simple, Realistic