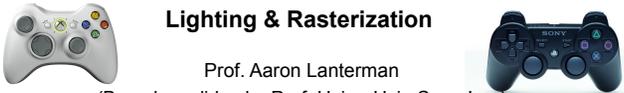


ECE4893A/CS4803MPG:
MULTICORE AND GPU
PROGRAMMING
FOR VIDEO GAMES



Lighting & Rasterization

Prof. Aaron Lanterman
 (Based on slides by Prof. Hsien-Hsin Sean Lee)
 School of Electrical and Computer Engineering
 Georgia Institute of Technology



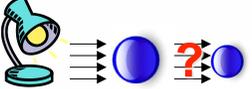
Illumination models

- It won't look 3-D without lighting
- Part of geometry processing
 - Can also be part of rasterization
- Illumination types
 - Ambient
 - Diffuse
 - Specular
 - Emissive



Local vs. global illumination

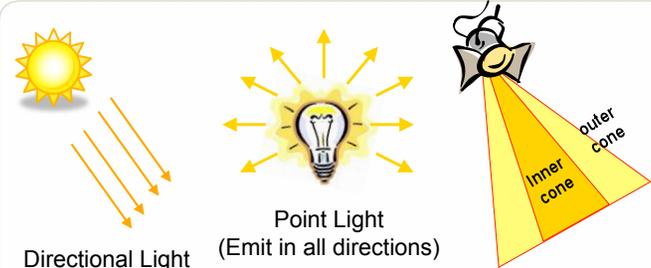
- Local illumination
 - Direct illumination: Light shines on all objects without blocking or reflection
 - Used in most games



- Global illumination
 - Indirect illumination: Light bounces from one object to other objects
 - Adds more realism (non real-time rendering)
 - Computationally much more expensive
 - Ray tracing, radiosity



Common light sources



Directional Light (Infinitely far away)

Point Light (Emit in all directions)

Spot Light (Emit within a cone)



Illumination: ambient lighting

- Not created by any light source
- A constant lighting from all directions
- Contributed by scattered light in a surrounding

Material Color \otimes Light Color \rightarrow Ambient Color

RGB multiplies separately

$$C_{ambient} = M_{ambiet} \otimes L_{ambient}$$

Georgia Institute of Tech

Illumination: diffuse lighting

- Light sources are given
- Assume light bounces in all directions

Diffuse surface

Reflected light will reach the eyes no matter where the camera is!

Georgia Institute of Tech

Reflected light intensity calculation

- Reflectivity \propto the entry angle
- Use Lambert's cosine Law

Reflection Full

Partial

None

intensity

$F(\theta) = \cos(\theta)$

$$C_{diff} = \max(-\vec{L} \cdot \vec{n}, 0) \cdot (M_{diff} \otimes L_{diff})$$

Dot product

Georgia Institute of Tech

Ambient + diffuse lighting

Material Color \otimes Light Color \rightarrow Ambient Color

Diffuse \rightarrow Ambient + Diffuse

Ambient + Diffuse

$$C_{diff} = M_{ambient} \otimes L_{ambient} + \max(-\vec{L} \cdot \vec{n}, 0) \cdot (M_{diff} \otimes L_{diff})$$

Georgia Institute of Tech

Illumination: specular lighting

- Create shining surface (surface perfectly reflects)
- Viewpoint dependent

Material Color Light Color Specular Ambient + Diffuse + Specular

Georgia Institute of Technology

Jim Blinn's specular model

$$\vec{H} = \frac{\vec{E} + \vec{L}}{|\vec{E} + \vec{L}|}$$

Half-way vector

$$C_{spec} = (\max(\vec{n} \cdot \vec{H}, 0))^S \cdot M_{spec} \otimes L_{spec}$$

- A (usually) more computationally efficient approximation of the Phong specular model that uses the reflective vector R
- "S" controls the bright region around surface

Georgia Institute of Technology

Specular brightness effect

$$C_{spec} = (\max(\vec{n} \cdot \vec{H}, 0))^S \cdot M_{spec} \otimes L_{spec}$$

where $\vec{n} \cdot \vec{H} = |\vec{n}| |\vec{H}| \cos \theta$

More shining

Georgia Institute of Technology

Role of brightness parameter S

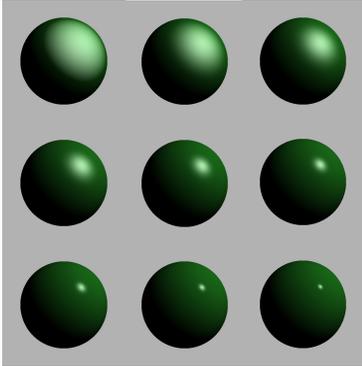
$$C_{spec} = (\max(\vec{n} \cdot \vec{H}, 0))^S \cdot M_{spec} \otimes L_{spec}$$

intensity

$-\pi/2$ $\pi/2$ radian

Georgia Institute of Technology

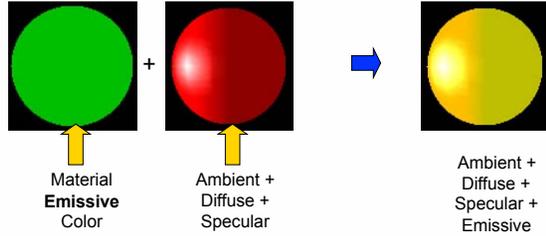
Specular lighting effect



- A larger S shows more concentration of the reflection

Georgia Institute of Technology

Illumination: emissive lighting



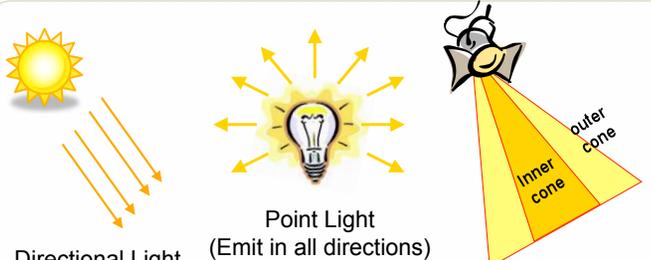
Material Emissive Color + Ambient + Diffuse + Specular → Ambient + Diffuse + Specular + Emissive

$$C_{all} = C_e + M_a \otimes L_a + \max(-\vec{L} \cdot \vec{n}, 0) \cdot (M_d \otimes L_d) + (\max(\vec{n} \cdot \vec{H}, 0))^n \cdot M_s \otimes L_s$$

- Color is emitted by the *material* only

Georgia Institute of Technology

Common light sources (revisited)



Directional Light (Infinitely far away)

Point Light (Emit in all directions)

Spot Light (Emit within a cone)

Georgia Institute of Technology

Light source properties

- Position
- Range
 - Specifying the visibility
- Attenuation
 - The farther the light source, the dimmer the color

$$Atten = a_0 + a_1 \cdot d + a_2 \cdot d^2$$

$$C_{all} = C_e + M_a \otimes L_a + \frac{\max(-\vec{L} \cdot \vec{n}, 0) \cdot (M_d \otimes L_d) + (\max(\vec{n} \cdot \vec{H}, 0))^n \cdot M_s \otimes L_s}{Atten}$$

Georgia Institute of Technology

Spotlight effect

$spot = (\max(\cos\alpha, 0))^f$
 $spot = (\max(\vec{L} \cdot \vec{d}, 0))^f$
 where f is the *falloff* factor

$C_{whatever} = spot \cdot C_{whatever}$

Falloff effect

- Similar in form to specular lighting (but different!)
- Falloff factor determines the fading effect of a spotlight
- " f " exponentially decreases the $\cos(\alpha)$ value

Georgia Institute of Technology

Rasterization: shading a triangle

- Converting geometry to a raster image (i.e., pixels)
- Paint each pixel's color (by calculating light intensity) on your display
- Gouraud shading: intensity interpolation of vertices

Georgia Institute of Technology

Gouraud shading

- Scan conversion algorithm

Georgia Institute of Technology

Comparison of shading methods

Source: Michal Necasek

Flat shading Gouraud shading Phong shading

- Gouraud shading supported by (even old) 3-D graphics hardware
- Phong shading
 - Requires generating per-pixel normals to compute light intensity for each pixel, not efficient for games
 - Can be done on modern GPUs using Cg or HLSL

Georgia Institute of Technology

Double buffering

- Display refreshes at 60 ~ 75 Hz
- Rendering could be “faster” than the refresh period
- Too fast leads to
 - Frames not shown
- Too slow leads to
 - New and old frame mixed
 - Flickering
- Solution:
 - Double or multiple buffering

Front Buffer

surface1



Back Buffer

surface2



↔ swap ↔

surface1



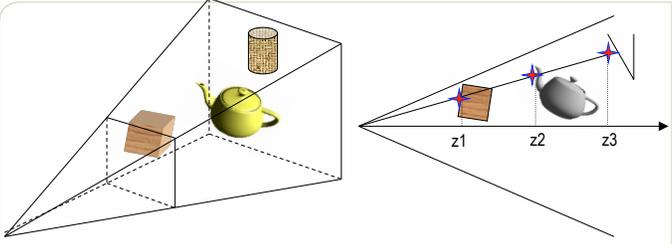
surface2

surface2



Georgia Institute of Technology

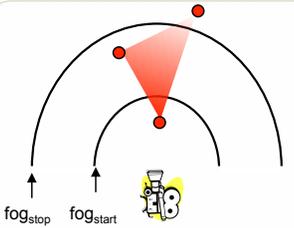
The Z-buffer



- Also called *depth buffer*
- Draw the pixel which is nearest to the viewer
- Number of the entries corresponding to the screen resolution (e.g. 1024x768 should have a 768k-entry Z-buffer)
- Granularity matters
 - 8-bit never used
 - 16-bit z value could generate artifacts

Georgia Institute of Technology

Fog effects



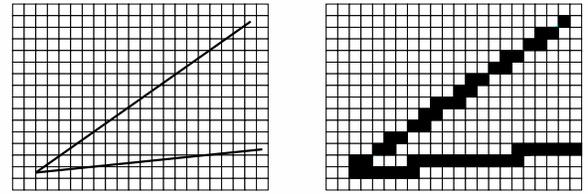
$$color = (1 - f) \cdot Color_{vertex} + f \cdot Color_{fog}$$

$$f = MAX\left(\frac{dist(eye, vertex) - fog_{start}}{fog_{stop} - fog_{start}}, 0\right)$$

- Calculate distance
- Calculate intensity of vertex color based on distance
 - Color blending
 - Linear density, exponential density
- Blending color schemes
 - Per-vertex (then interpolate pixels), less expensive
 - Per-fragment basis (NVIDIA hardware), better quality
- Provide depth cue
 - Simulate weather condition
 - Avoid *popping* effect
- Color blending

Georgia Institute of Technology

Aliasing



- Jagged line (or staircase)
- Can be improved by increasing resolution (i.e. more pixels)

Georgia Institute of Technology

Anti-aliasing by multisampling (Example: Supersampling)

3x3 Virtual Pixels
(Bartlett window)

(255, 159, 159)

Actual Screen Pixels

Example

- GPU samples multiple locations for a pixel
- Several different methods
 - e.g., grid (as shown), random, GeForce's quincunx
- Downside
 - Blurry image
 - Increased memory (e.g., z-buffer) storage for subpixel information

Georgia Institute of Technology

Anti-aliasing example

Ideal

No MSAA

With MSAA

Georgia Institute of Technology

Visualizing anti-aliasing example

No MSAA

With MSAA

Georgia Institute of Technology