# 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

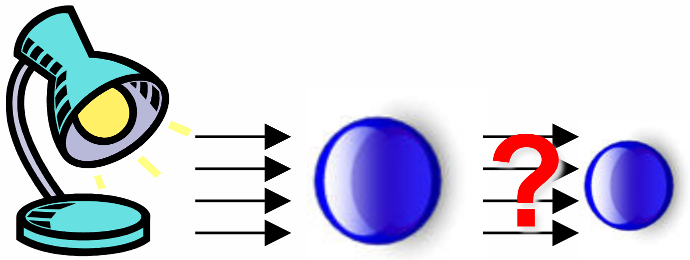
# Illumination models

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

Georgia Institute of Technology

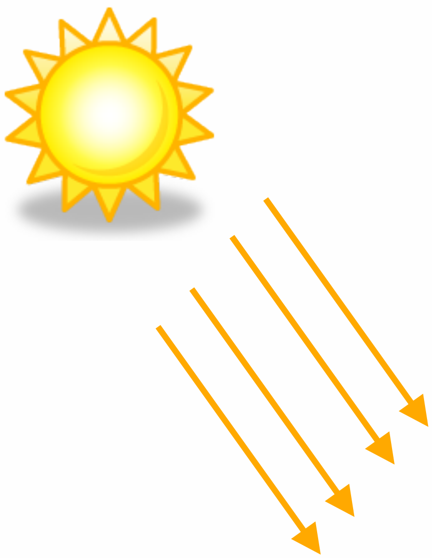
# Local vs. global illumination

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

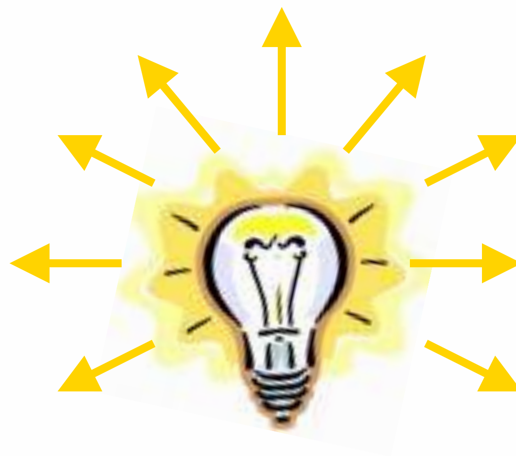



- Global illumination
  - Indirect illumination: Light bounces from one object to other objects
  - Adds more realism (non real-time rendering)
  - Computationally much more expensive
  - Ray tracing, radiosity

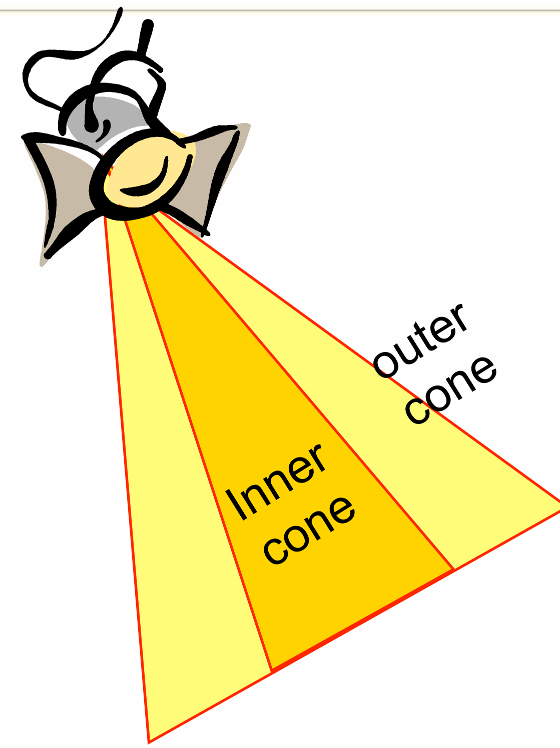
# Common light sources



Directional Light
(Infinitely far away)



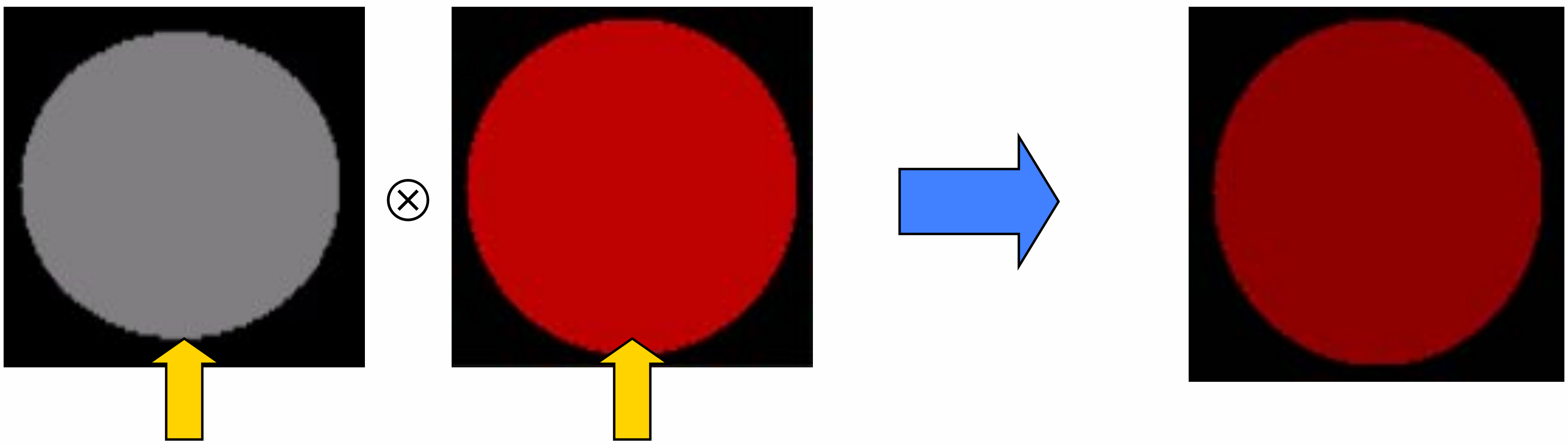
Point Light
(Emit in all directions)




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

Light Color

RGB multiplies separately

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

# Illumination: diffuse lighting

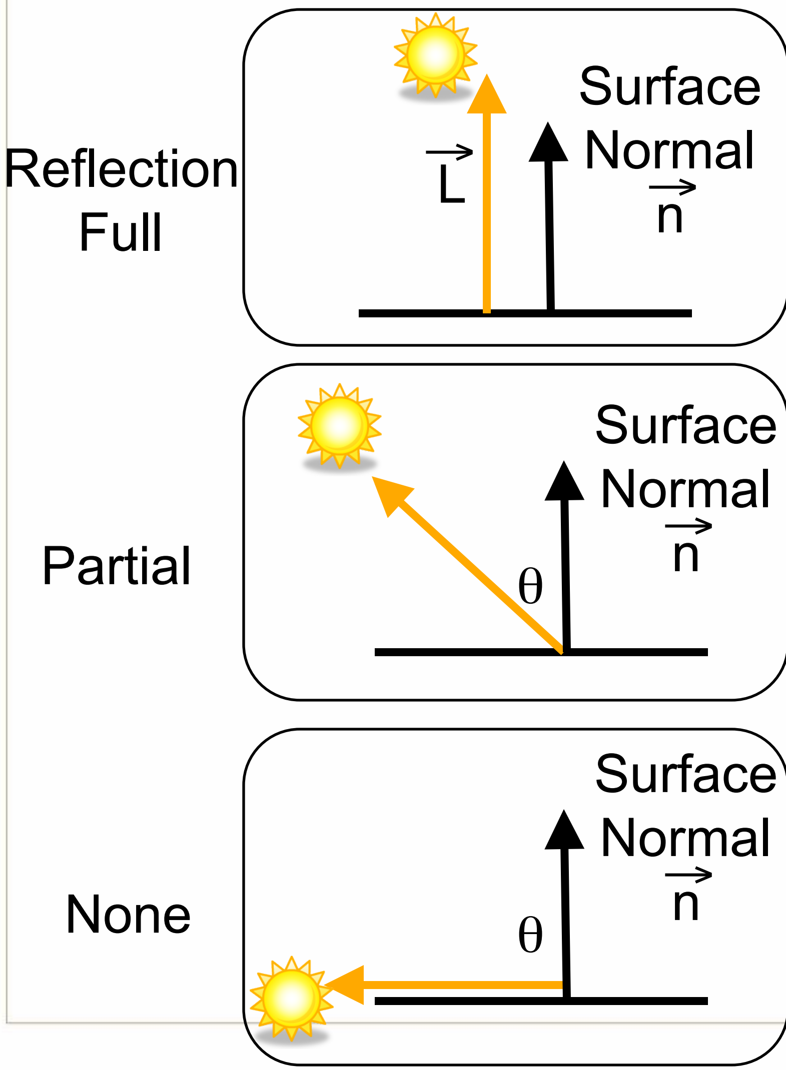
- Light sources are given
- Assume light bounces in all directions

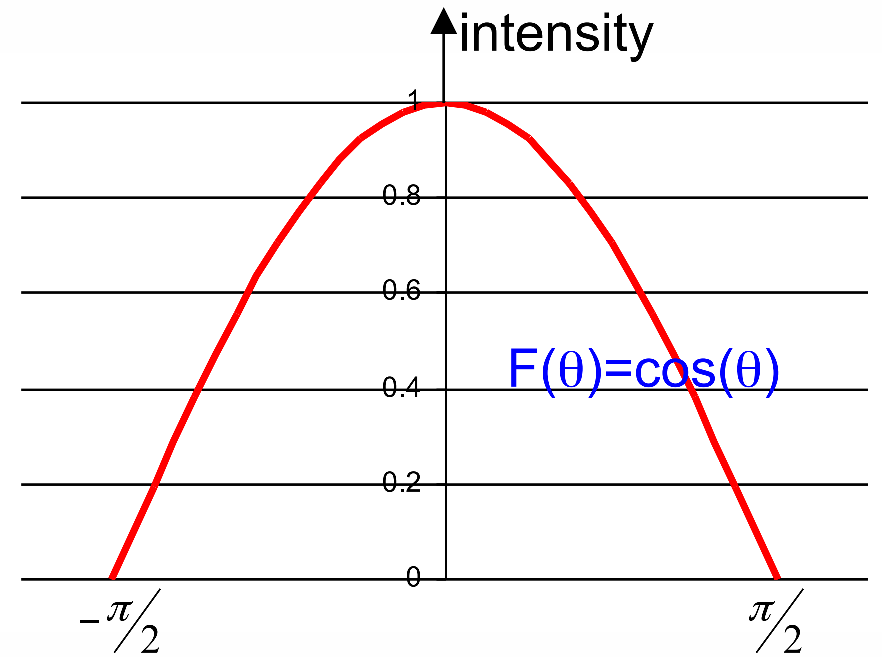
Diffuse surface

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

# Reflected light intensity calculation

- Reflectivity ∝ the entry angle
- Use **Lambert's cosine Law**



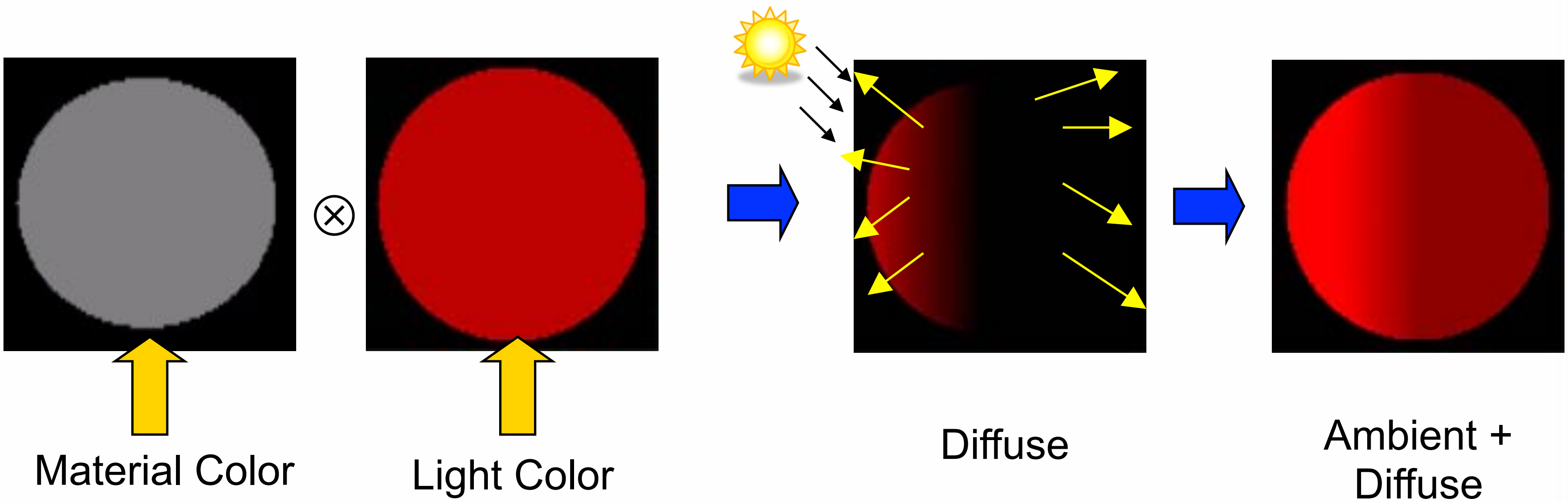




$$C_{diff} = \max(\vec{L} \bullet \vec{n}, 0) \cdot (M_{diff} \otimes L_{diff})$$

↑ Dot product

Georgia Institute of Technology

# Ambient + diffuse lighting

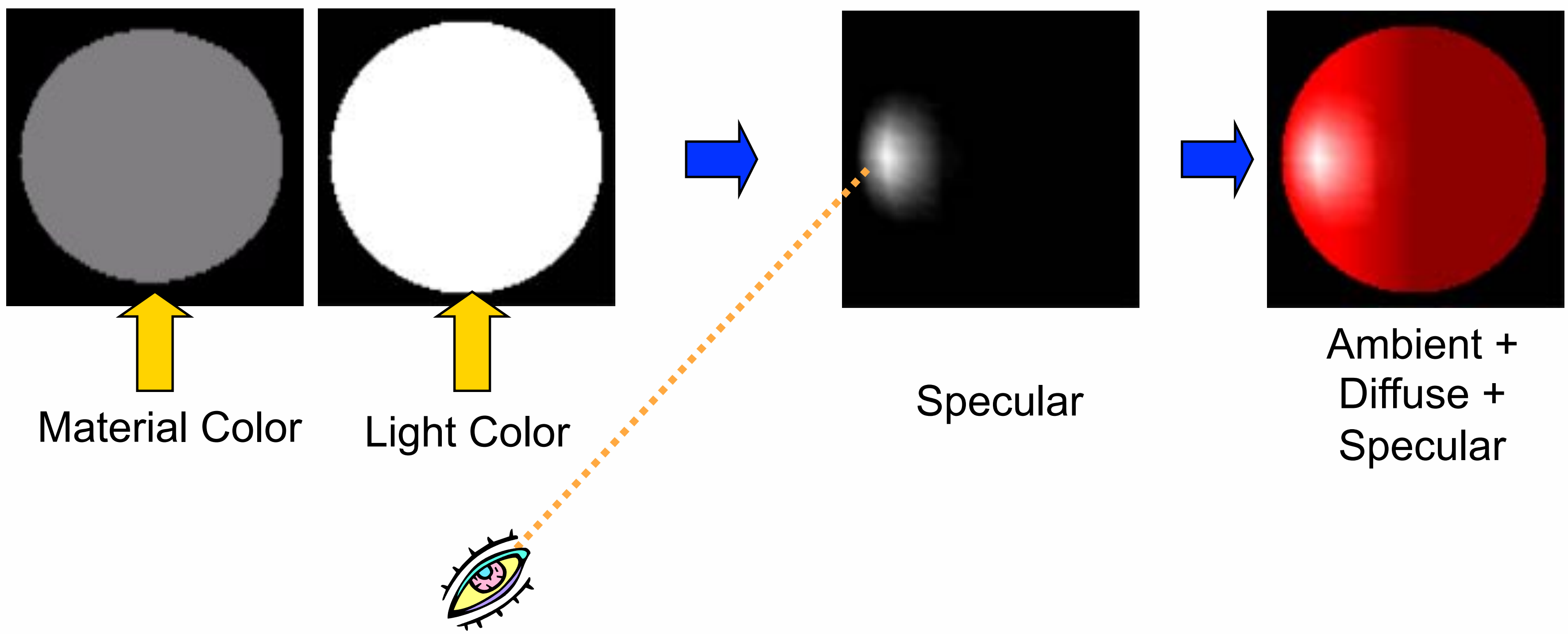



Ambient + Diffuse

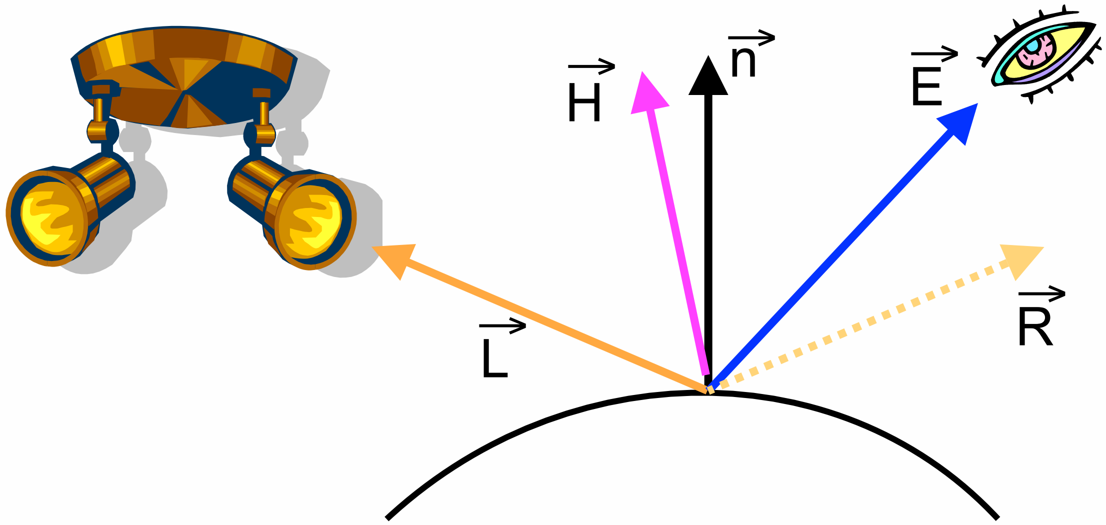
$$C_{diff} = M_{ambient} \otimes L_{ambient} + \max(\vec{L} \bullet \vec{n}, 0) \cdot (M_{diff} \otimes L_{diff})$$

# Illumination: specular lighting

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



# Jim Blinn's specular model




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

Half-way vector

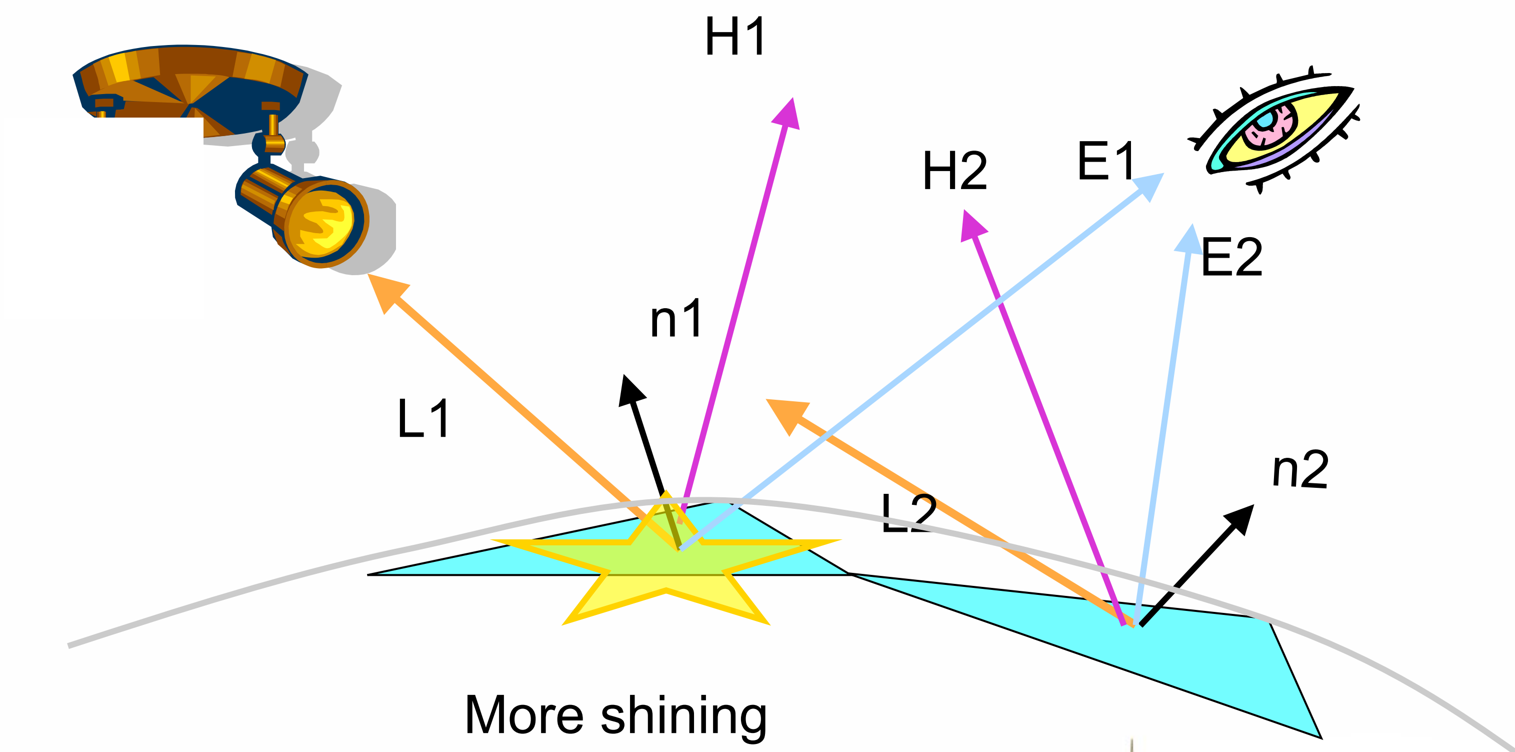
$$C_{spec} = (\max(\vec{n} \bullet \vec{H}, 0))^{S} \cdot M_{spec} \otimes L_{spec}$$

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

# Specular brightness effect

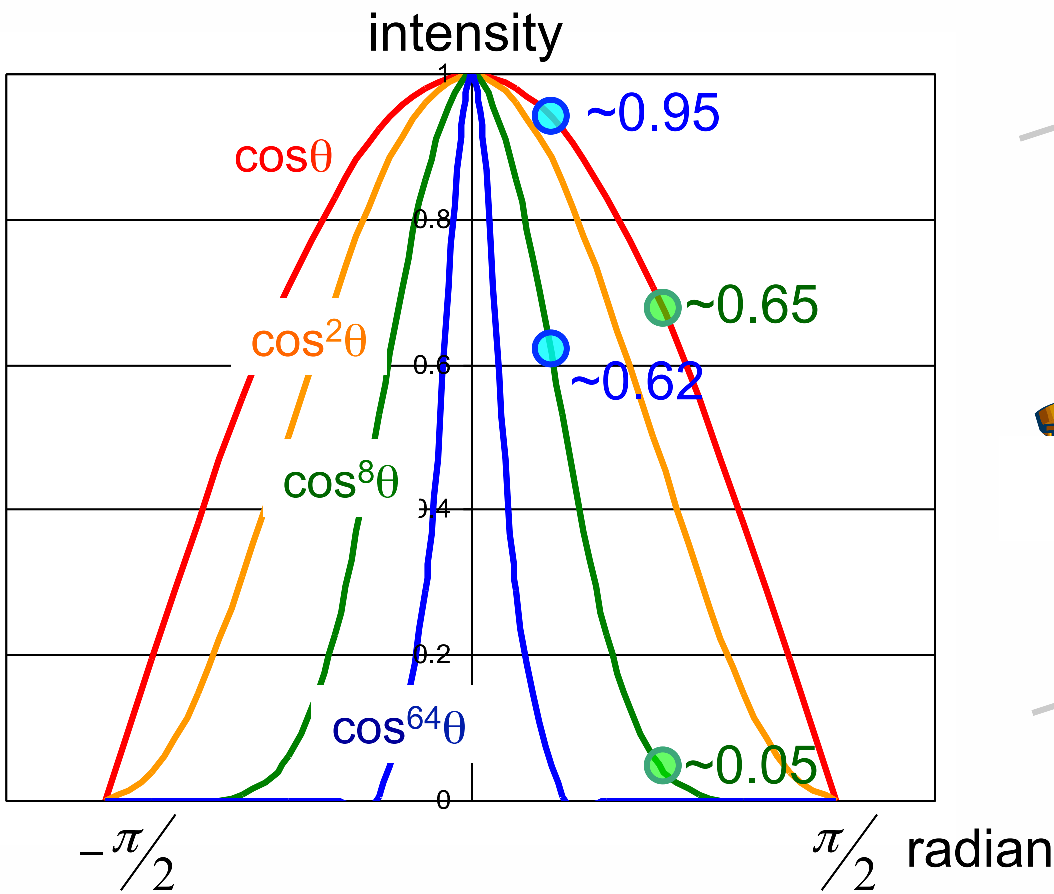
$$C_{spec} = (\max(\vec{n} \bullet \vec{H}, 0))^S \cdot M_{spec} \otimes L_{spec}$$

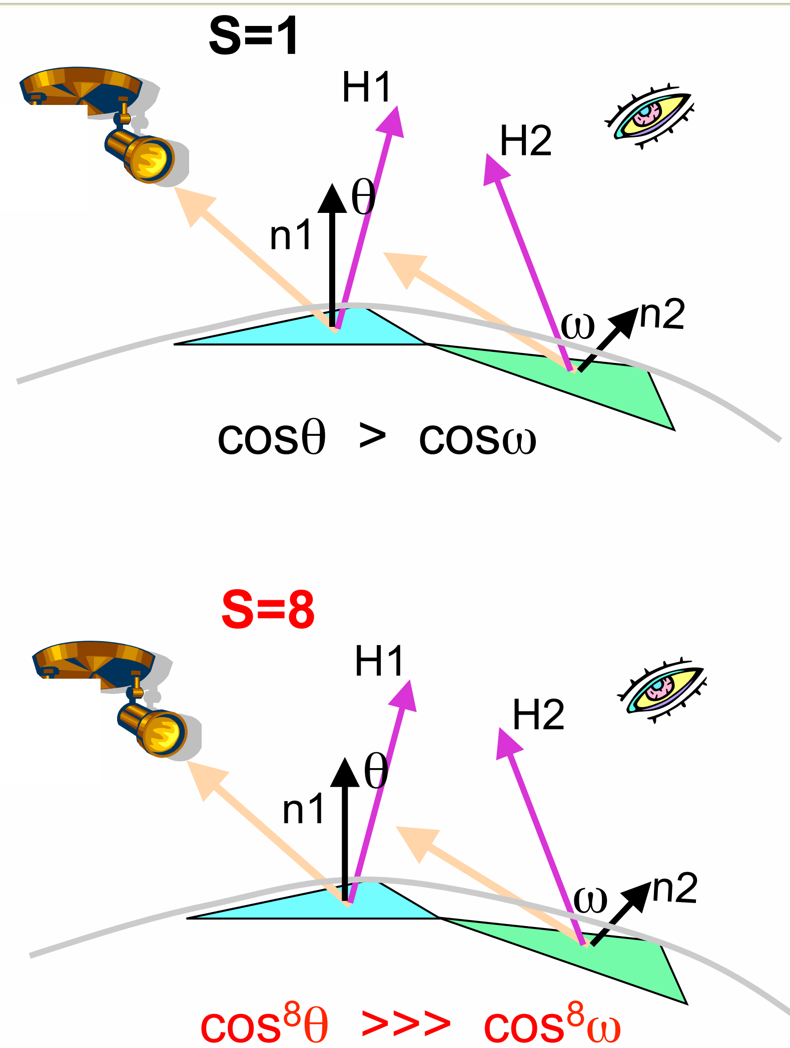
$$\text{where} \quad \vec{n} \bullet \vec{H} = |n||H|\cos\theta$$



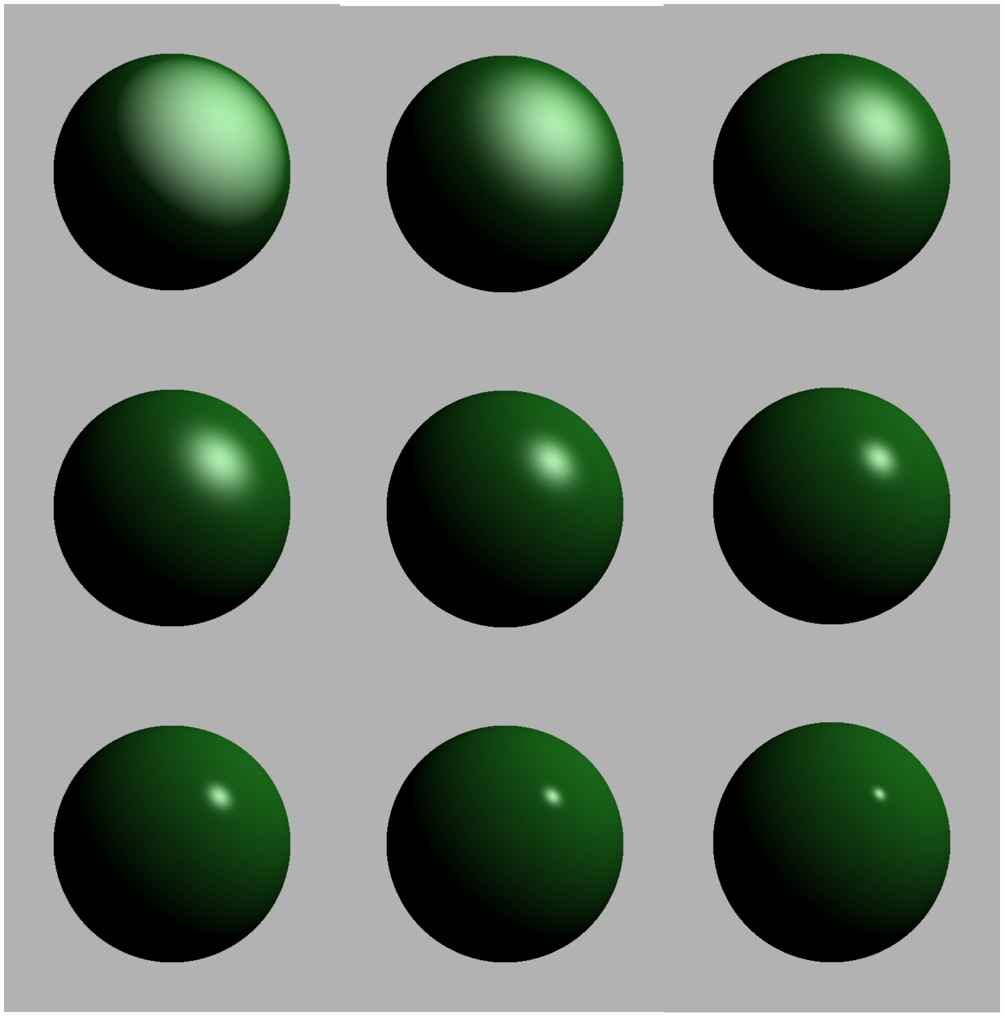
# Role of brightness parameter *S*

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



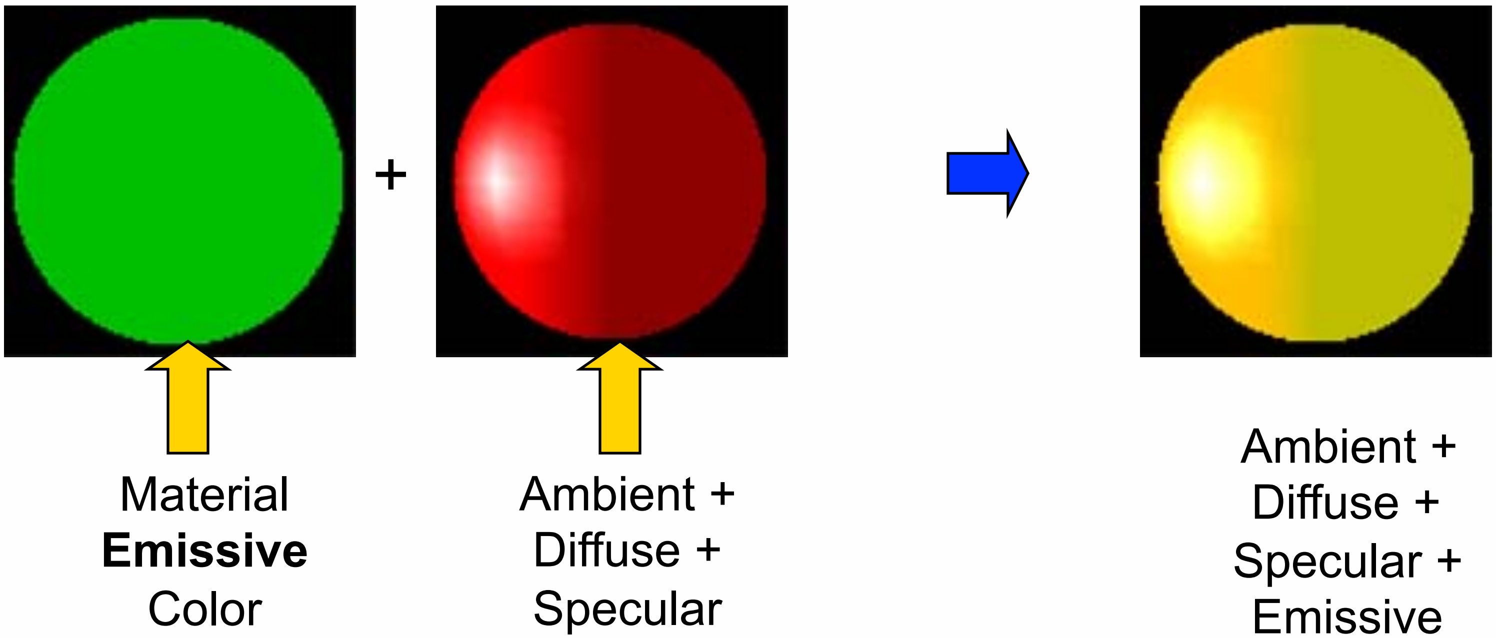
# Specular lighting effect



- A larger S shows more concentration of the reflection

Georgia Institute of Technology

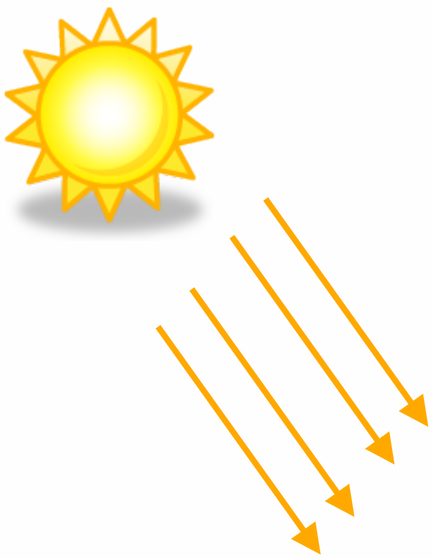
# Illumination: emissive lighting



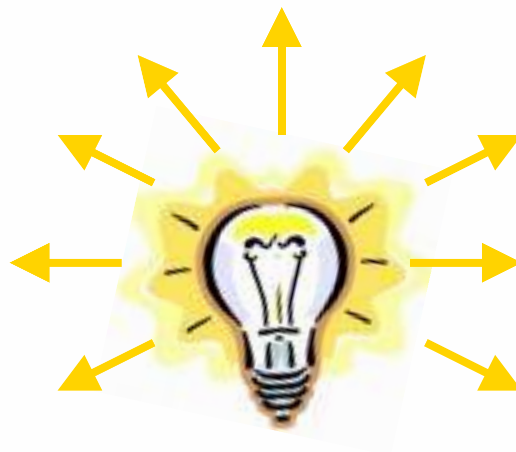

$$C_{all} = C_e + M_a \otimes L_a + \max(\vec{L} \bullet \vec{n}, 0) \cdot (M_d \otimes L_d) + (\max(\vec{n} \bullet \vec{H}, 0))^n \cdot M_s \otimes L_s$$

- Color is emitted by the **material** only

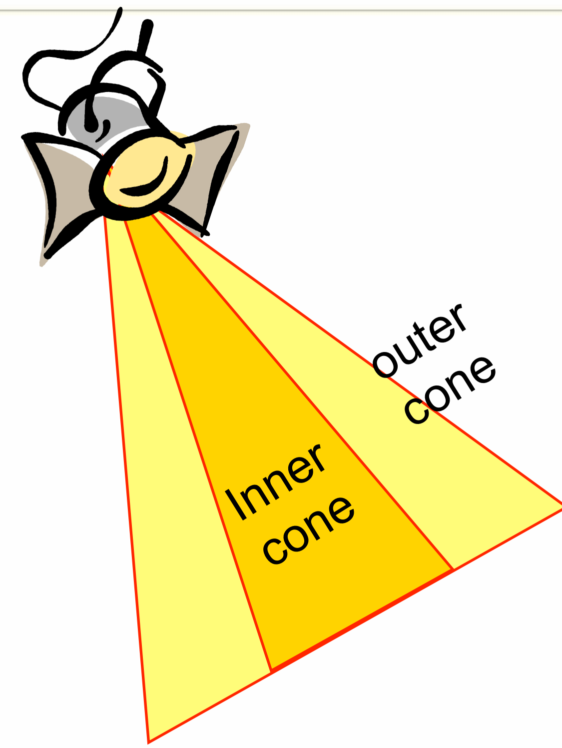
# Common light sources (revisited)



Directional Light
(Infinitely far away)



Point Light
(Emit in all directions)




Spot Light
(Emit within a cone)

# Light source properties

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

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

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

Georgia Institute of Technology

# Spotlight effect

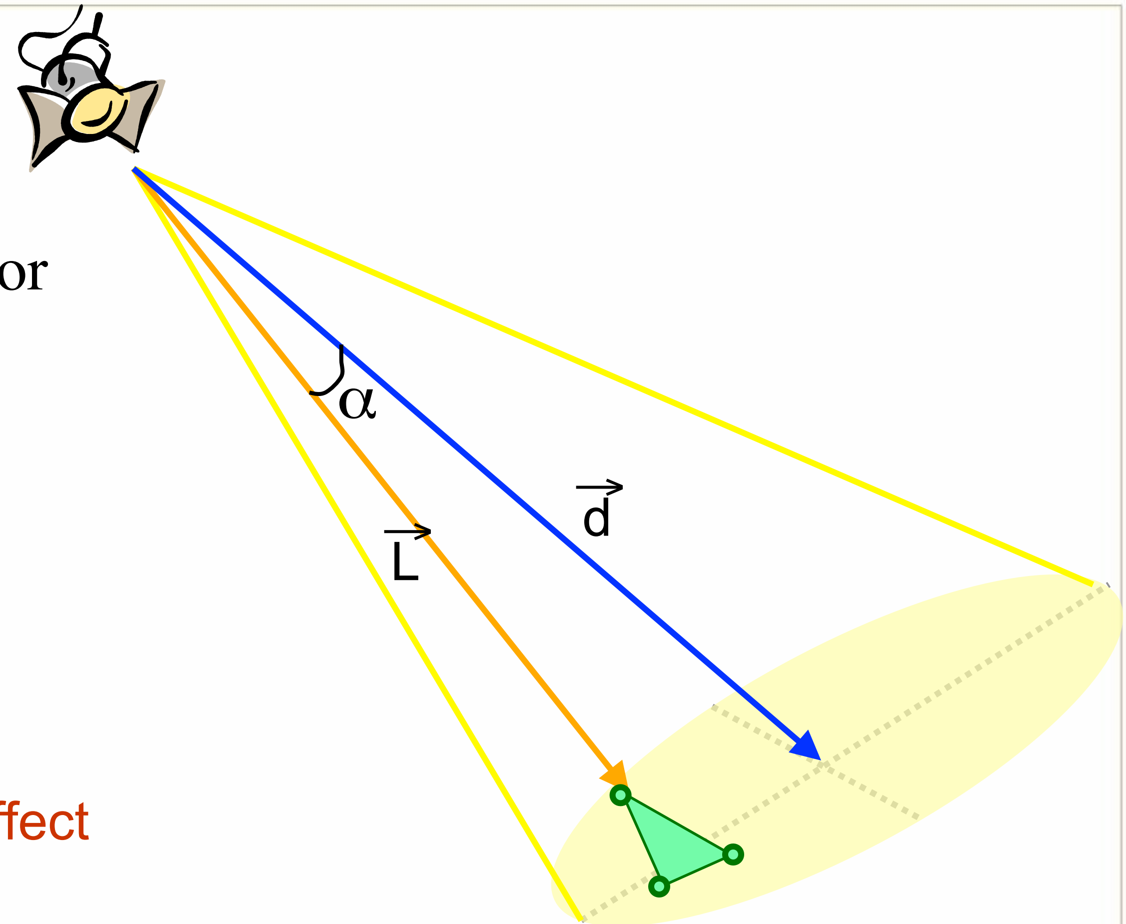
$$spot = (\max(\cos\alpha, 0))^{f}$$

$$spot = (\max(\vec{L} \bullet \vec{d}, 0))^{f}$$

where f is the *falloff* factor

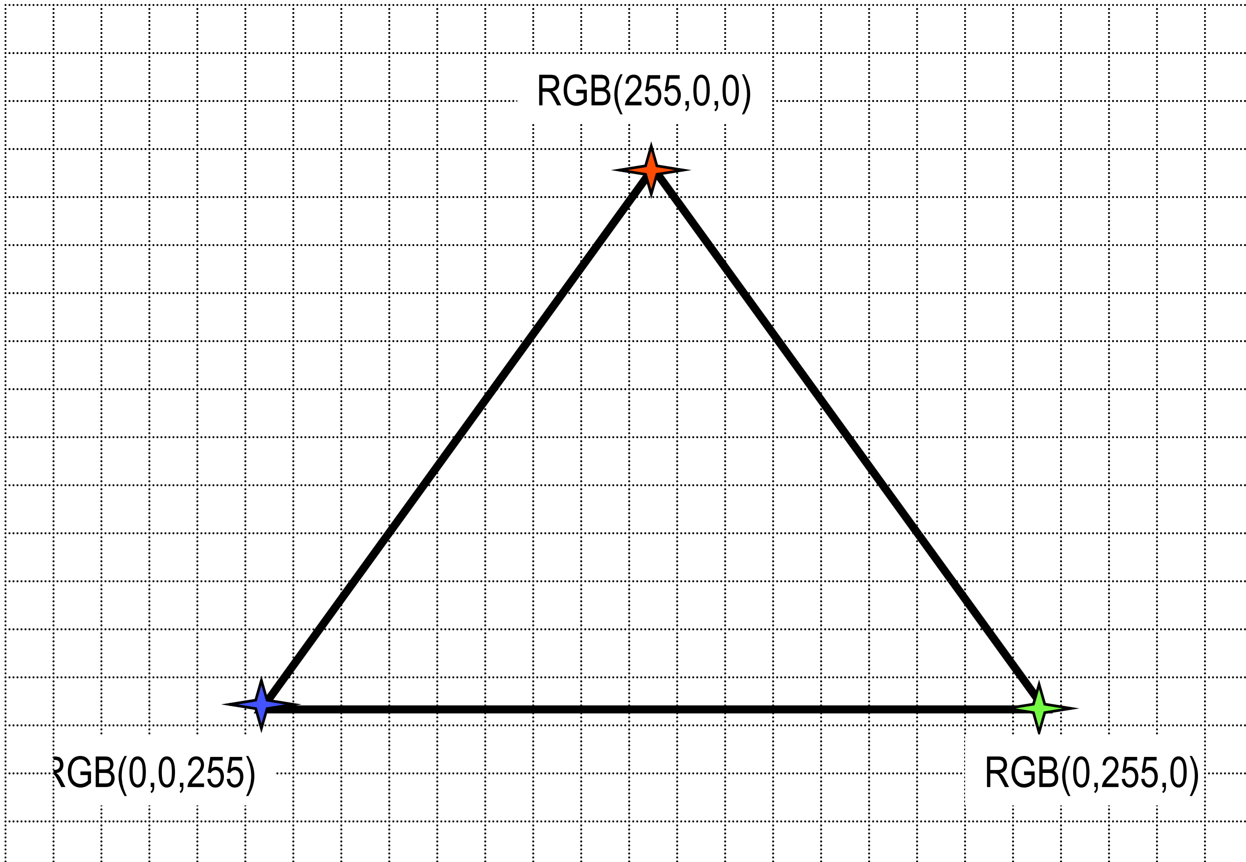
$$C_{whatever} = spot \cdot C_{whatever}$$

Falloff effect



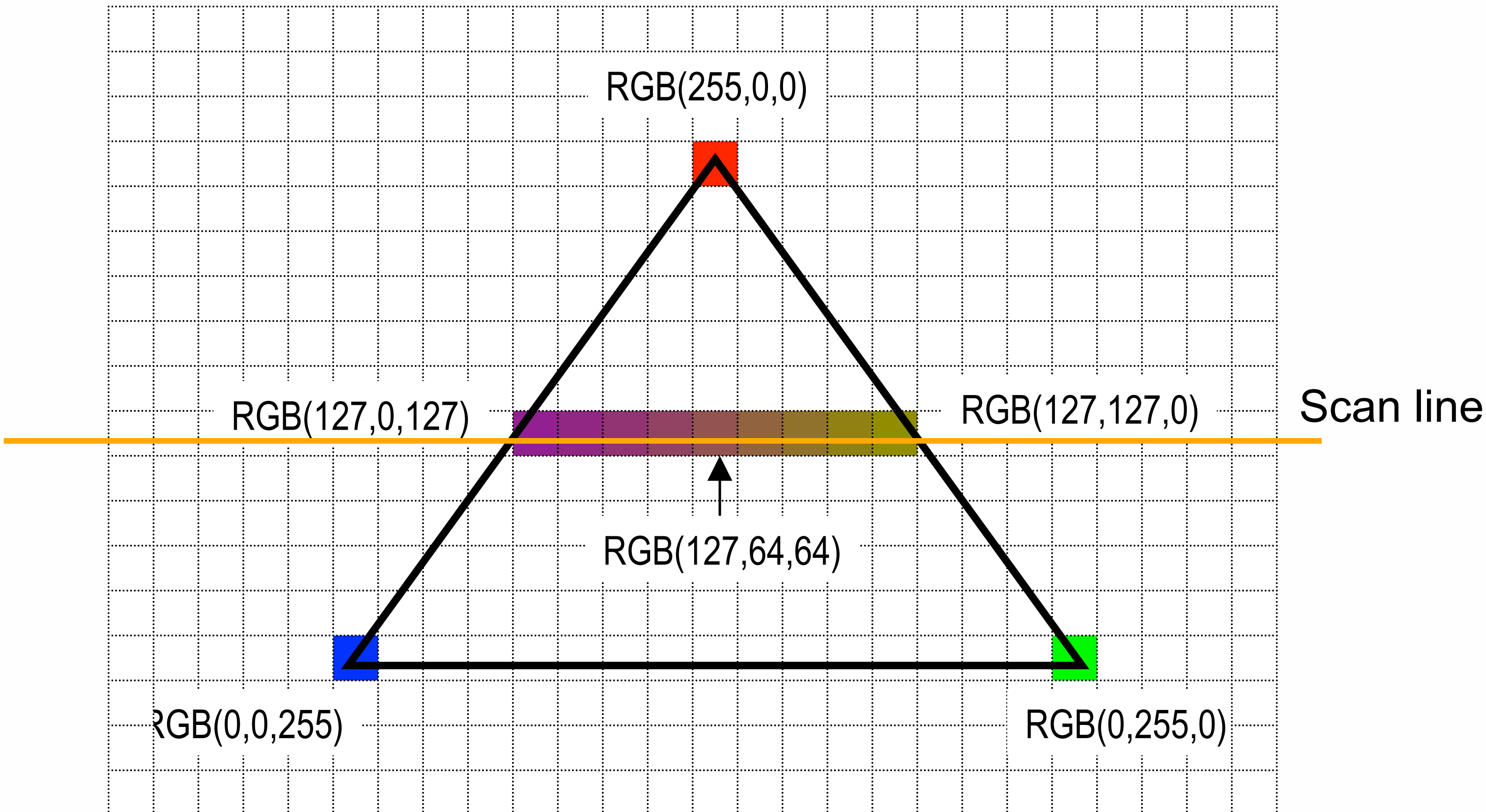

- Similar in form to specular lighting (but different!)
- Falloff factor determines the fading effect of a spotlight
- "*f*" exponentially decreases the cos(α) value

# 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

# Gouraud shading

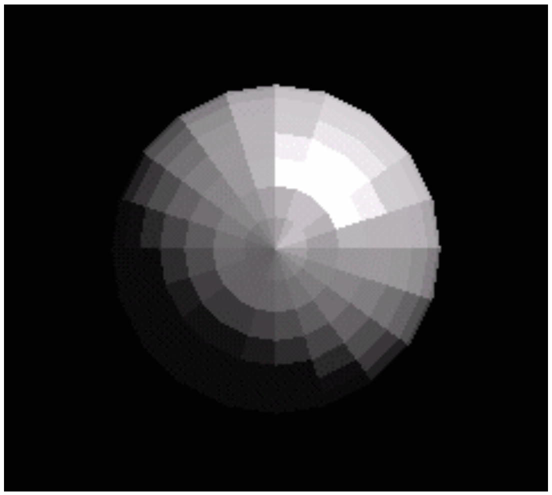



- Scan conversion algorithm

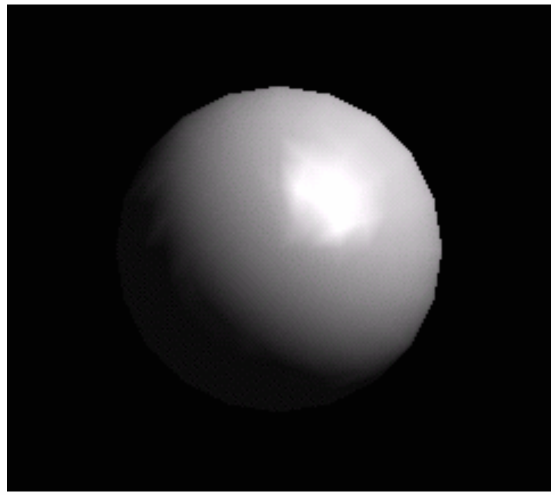
Georgia Institute of Technology

# Comparison of shading methods

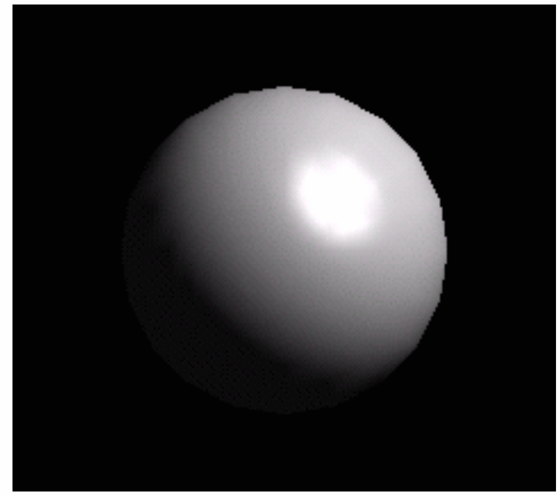
Source: Michal Necasek



Flat shading



Gouraud shading

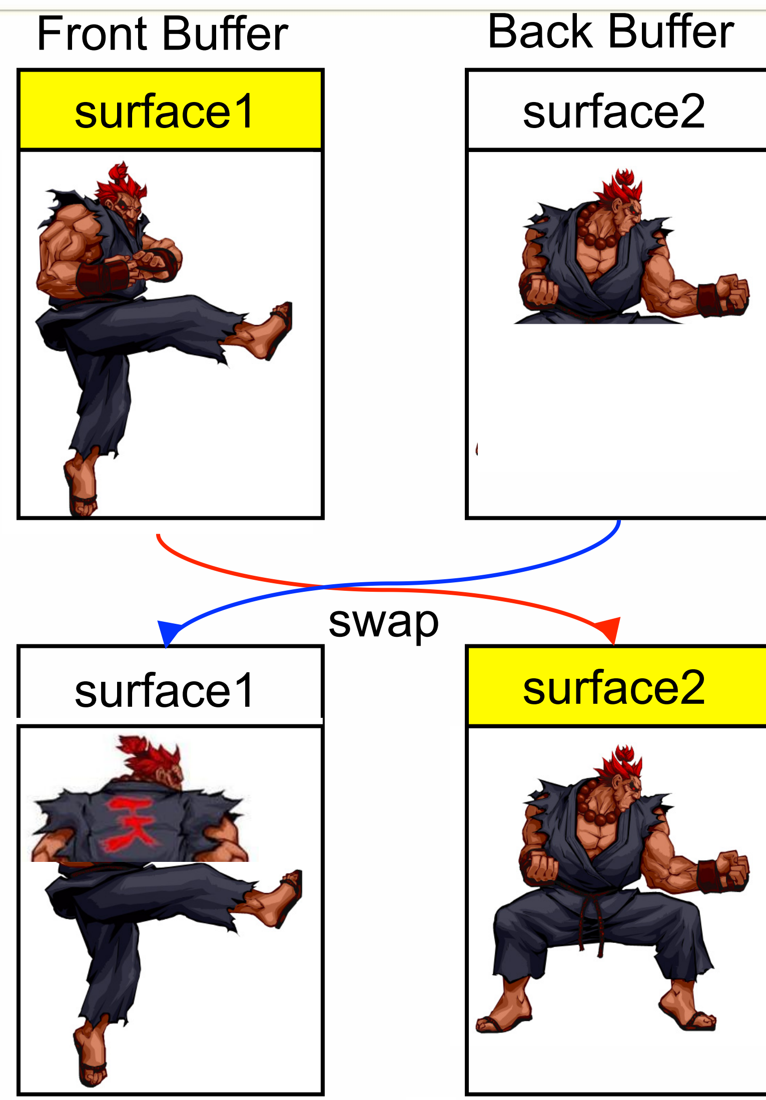


Phong shading

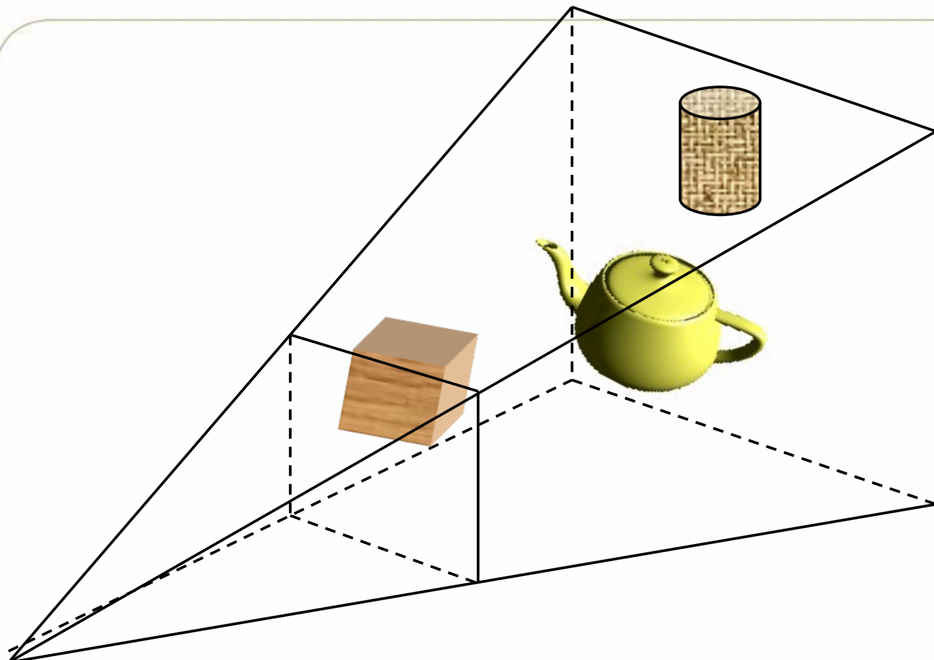
- Gouraud shading supported by (even old) 3-D graphics hardware
- Phong shading
  - Requires generating per-pixel normals to compute light intensity for each pixel, not efficient for games
  - Can be done on modern GPUs using Cg or HLSL

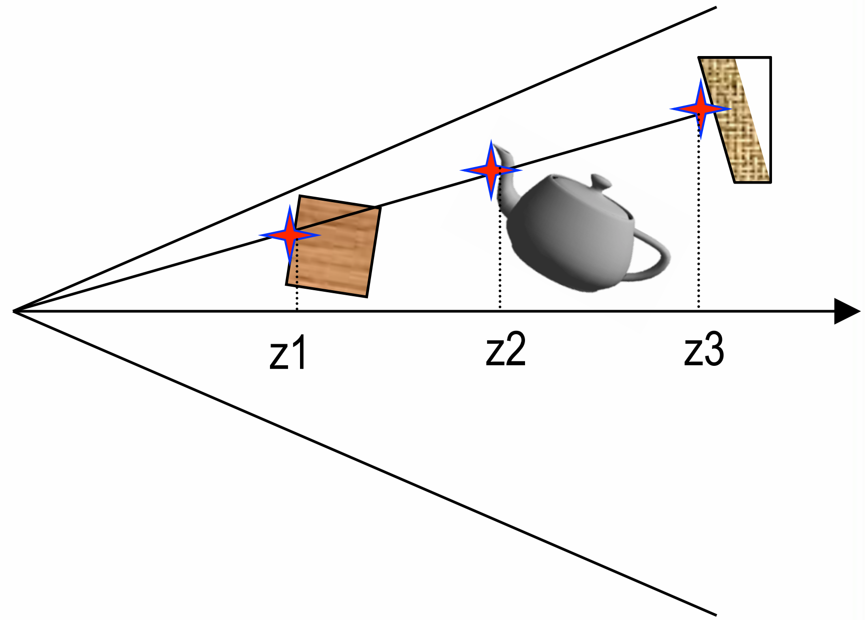
# Double buffering

- 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**



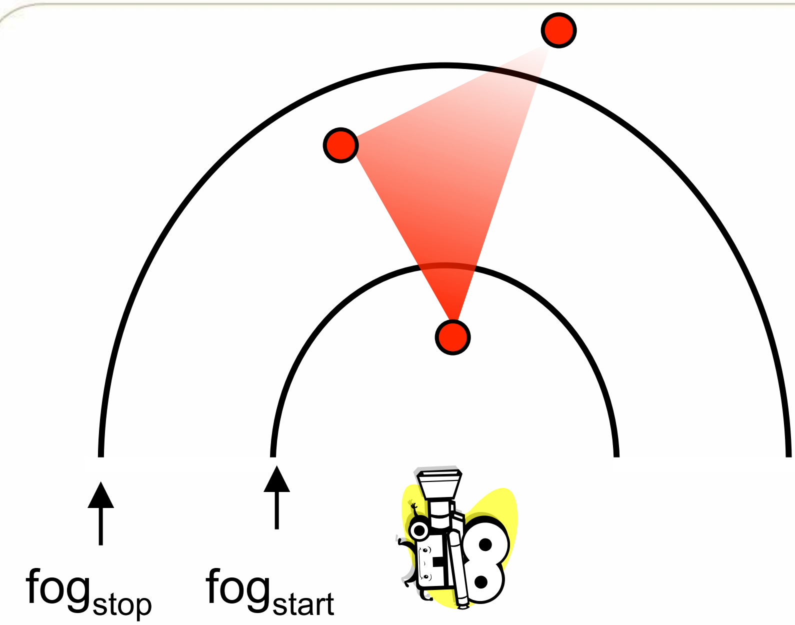
# The Z-buffer






- 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

# Fog effects




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

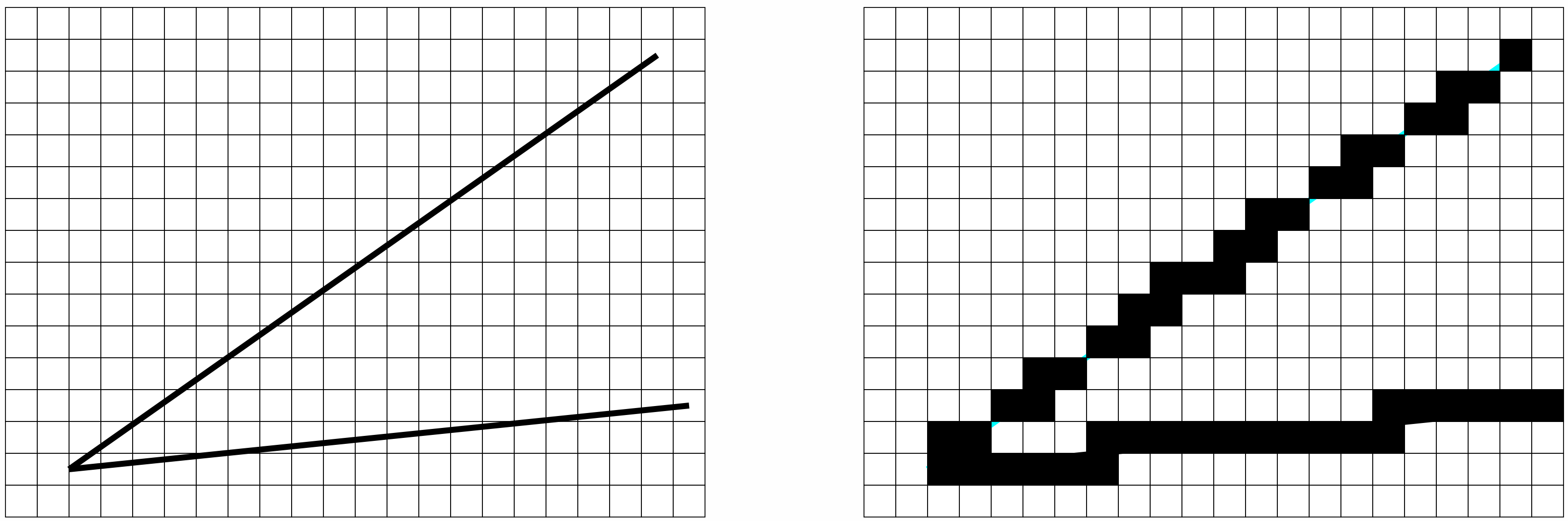
$$f = MAX\left(\frac{dist(eye, vertex) - fog_{start}}{fog_{stop} - fog_{start}}, 0\right)$$

- Provide depth cue
  - Simulate weather condition
  - Avoid *popping* effect
- Color blending

- 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

Georgia Institute of Technology

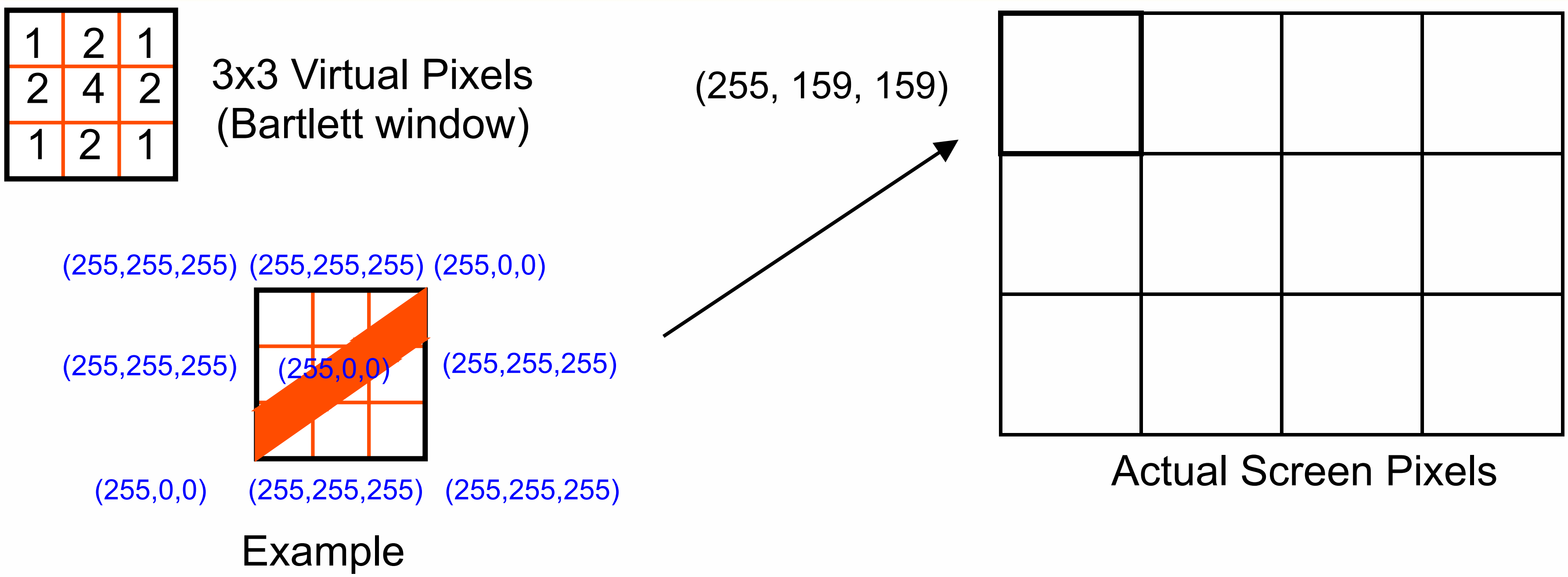
# Aliasing



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

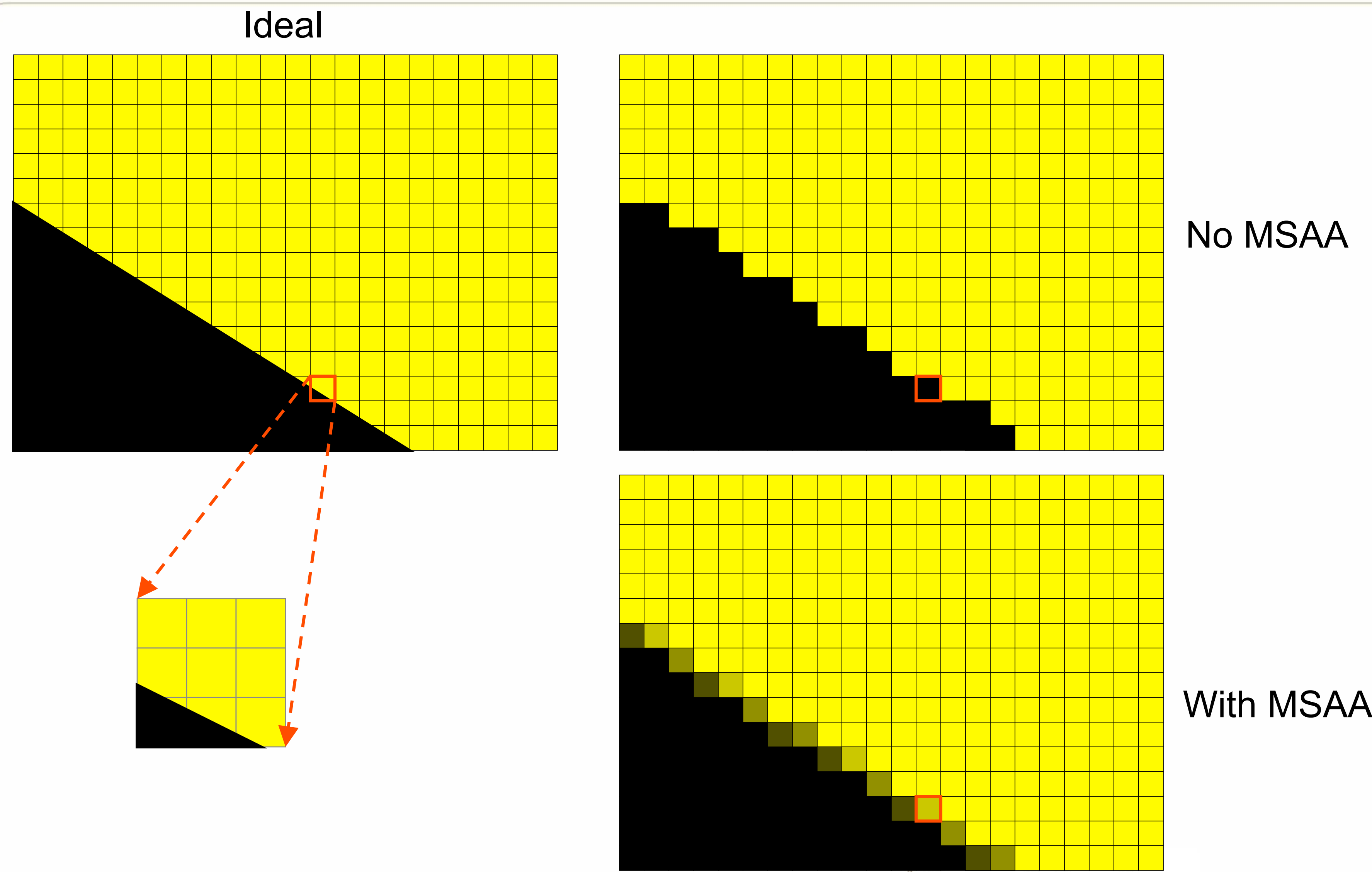
Georgia Institute of Technology

# Anti-aliasing by multisampling (Example: Supersampling)




- 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

# Anti-aliasing example



# Visualizing anti-aliasing example

No MSAA With MSAA