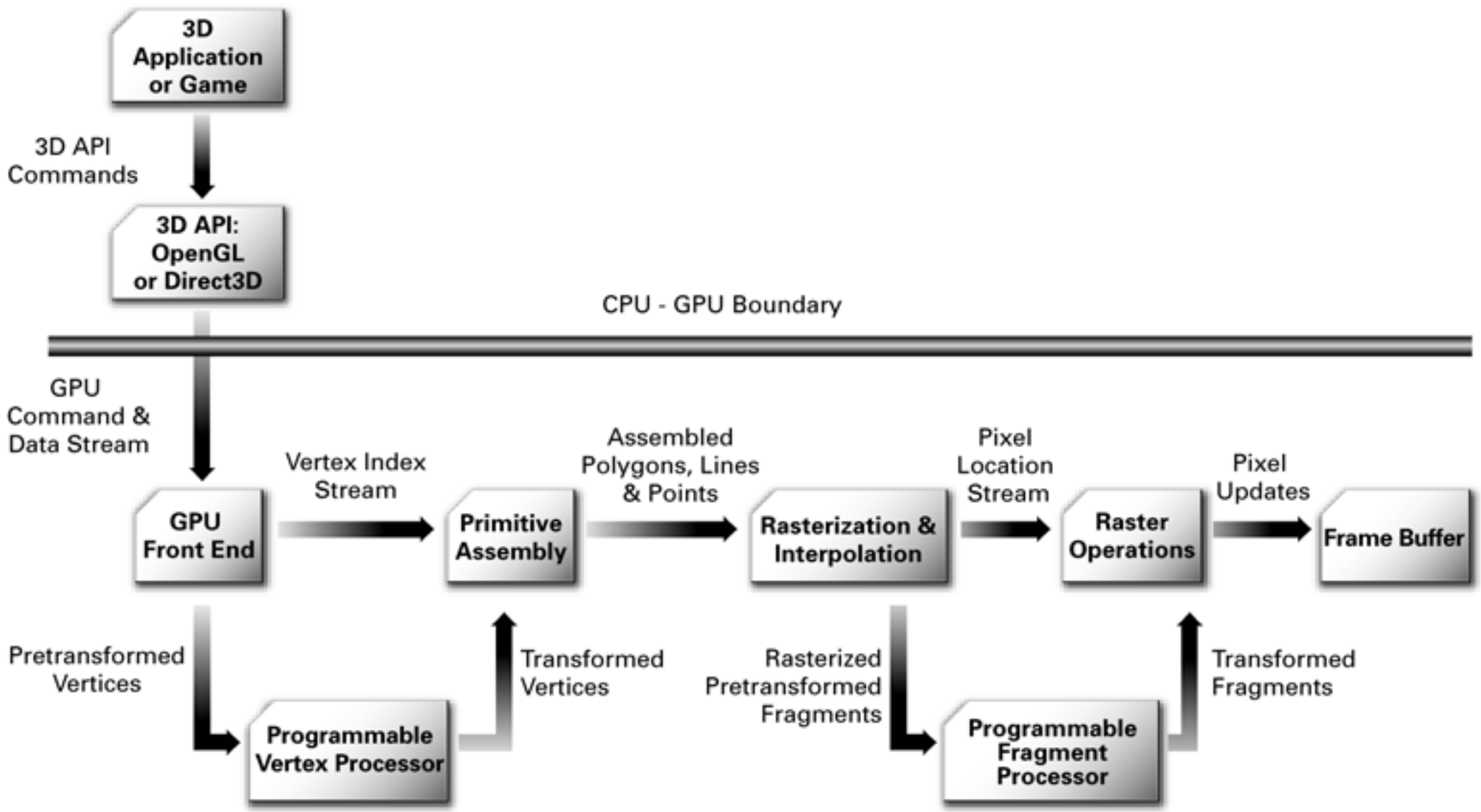
CS354R

DR SARAH ABRAHAM

GRAPHICS PIPELINE OVERVIEW

RENDERING

- Abstracted graphics pipeline:



FROM INTERACTIVE 3D GRAPHICS AND ANIMATION (HTTP://GOANNA.CS.RMIT.EDU.AU/~GL/TEACHING/INTERACTIVE3D/)

GRAPHICS PROCESSING UNIT

- The GPU is designed for high throughput
  - High bandwidth
  - Masked latency (multi-threading)
- Processes vertices and fragments (pixels)
- Now used as a general purpose, high throughput processor (GP-GPU)
  - But we'll focus on it's original task: shading!

SHADING

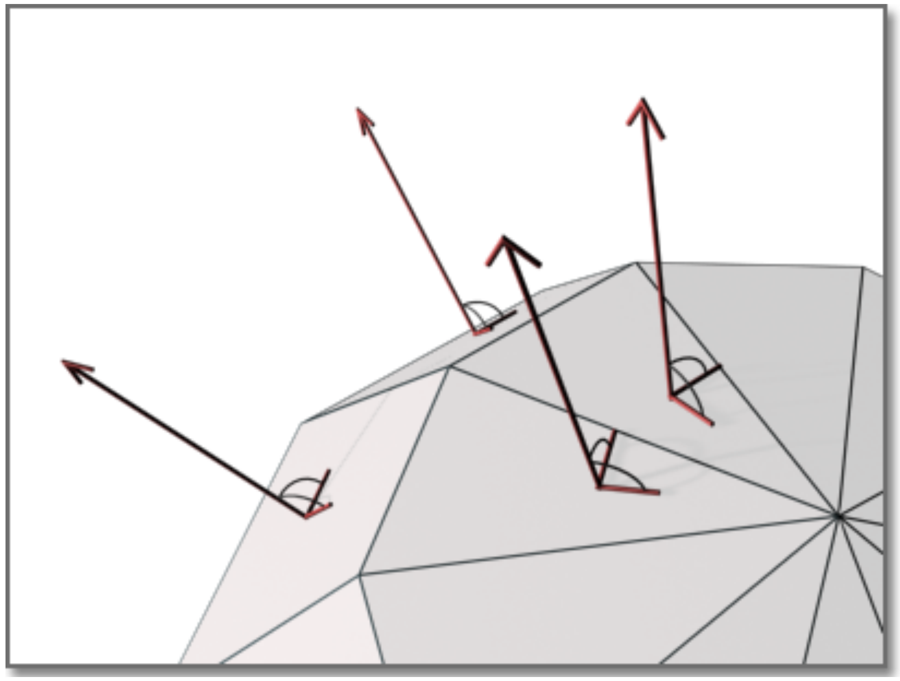
- Shading approximates the physical properties of light emission, reflectance, refraction, transmission etc
- What color is the object?
- What is the object's material?
- How does light interact with the object?
- What is the camera's position relative to the objects and lights?

Toy Story (1995)

Big Hero 6 (2014)

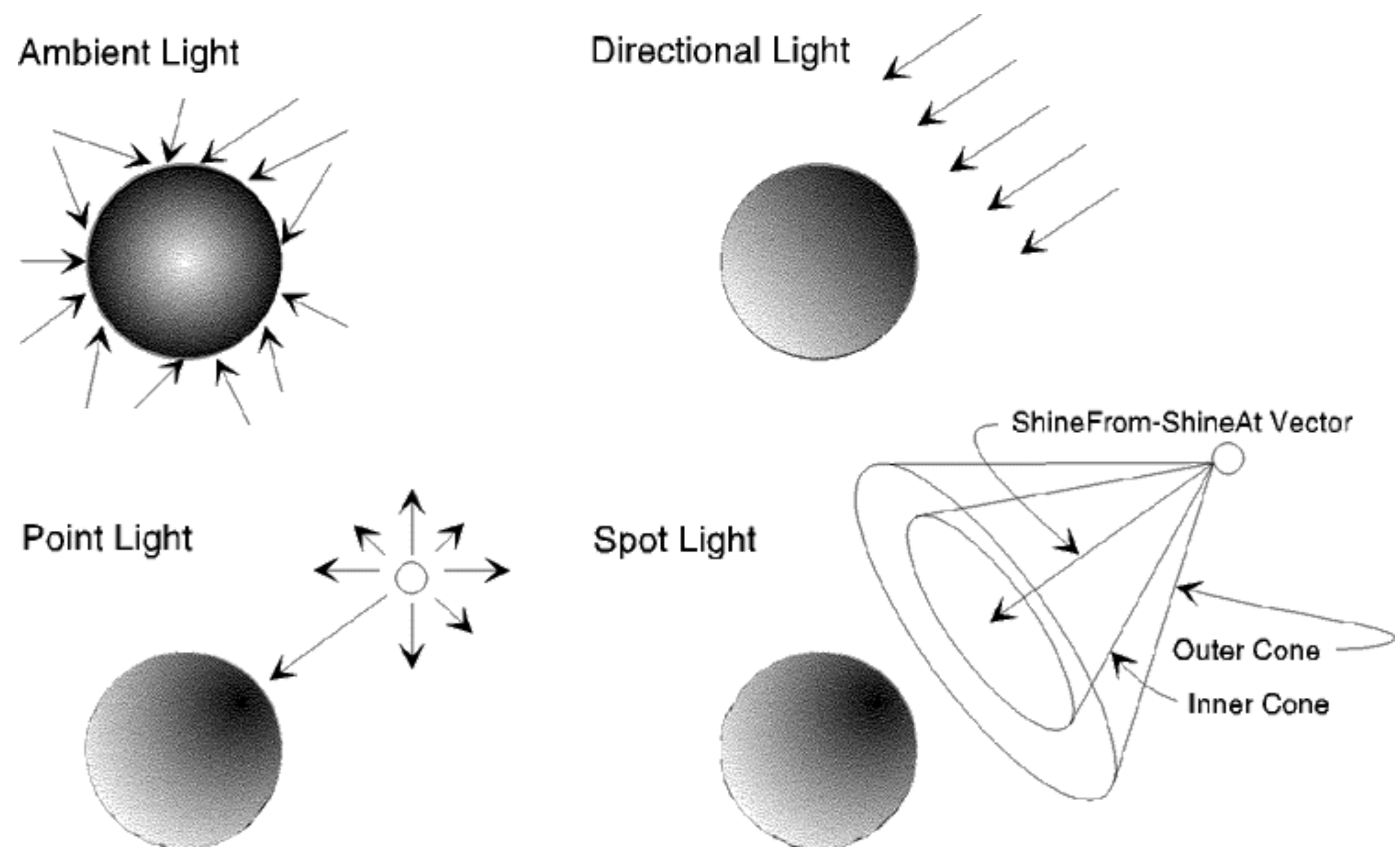
TRADITIONAL LIGHTING MODELS

- Lights assume a light direction and/or position for each light source
- The intensity of a surface depends on its **orientation** with respect to the light and the viewer
  - Surface orientation is the normal (N), which is perpendicular to the surface tangent plane



TRADITIONAL LIGHT SOURCES

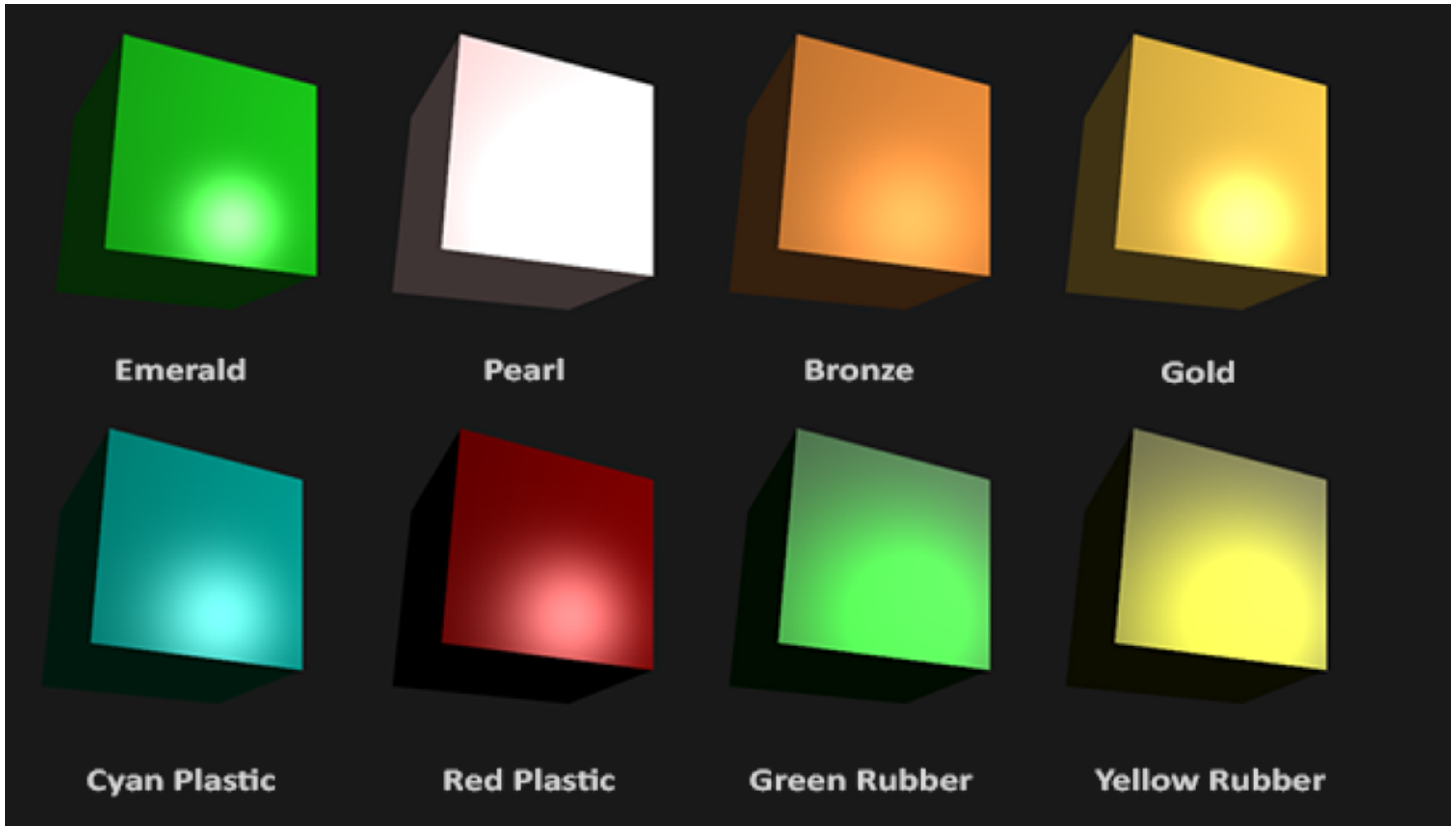
- Intensity and direction of light sources can change what surfaces are affected
- Potential light sources include:
  - Ambient
  - Point
  - Spot
  - Directional



Okino Computer Graphics

TRADITIONAL MATERIAL MODEL

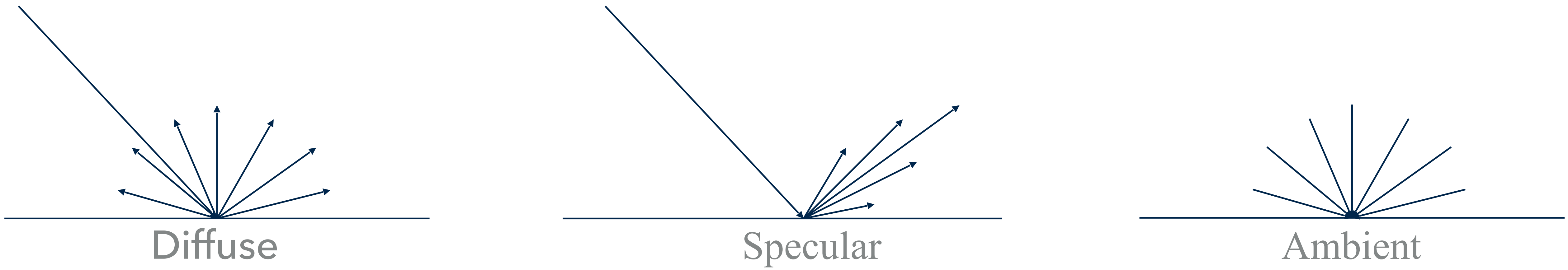
- Materials approximate basic physical interactions of light on their surface
  - Consider light direction, surface normals, and view direction



(https://learnopengl.com/Lighting/Materials)

STANDARD LIGHT INTERACTIONS

- Linearly combine several simple terms to model light and material interaction:
  - Diffuse component for the amount of incoming light reflected equally in all directions
  - Specular component for the amount of light reflected in a mirror-like fashion
  - Ambient term to approximate light arriving via other surfaces

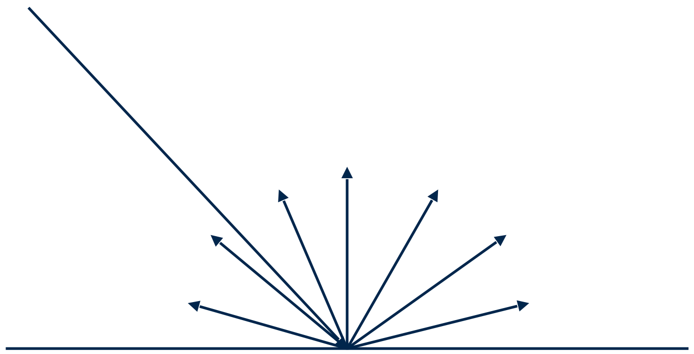


DIFFUSE ILLUMINATION

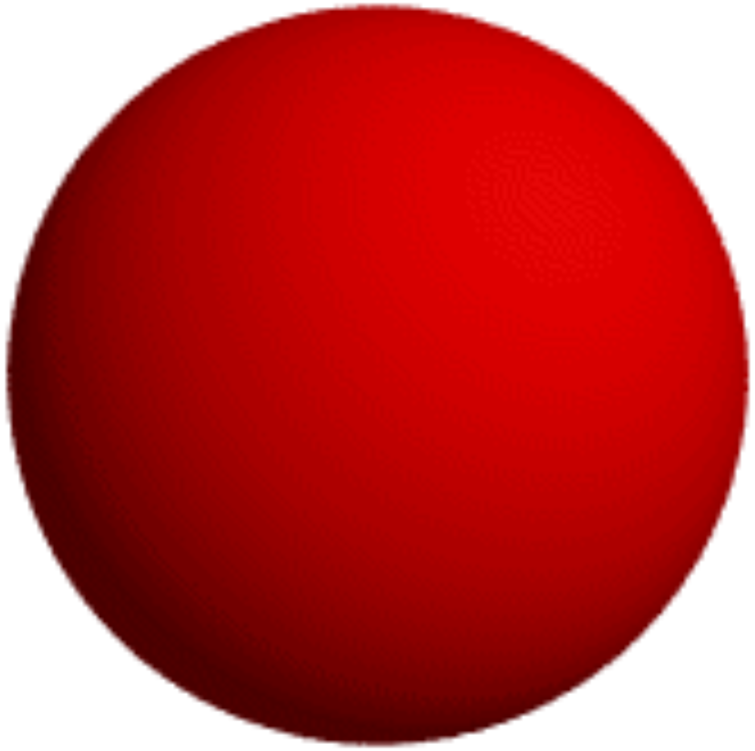
- Incoming light, I_i, from direction L, is reflected equally in all directions
  - No dependence on viewing direction
- Amount of light reflected depends on angle of surface with respect to light source
  - Determines how much light is collected by the surface to be reflected
  - Diffuse reflectance coefficient of the surface, k_d

We don't want to illuminate back side, so use:

$$k_d I_i max(L \cdot N, 0)$$



DIFFUSE EXAMPLE

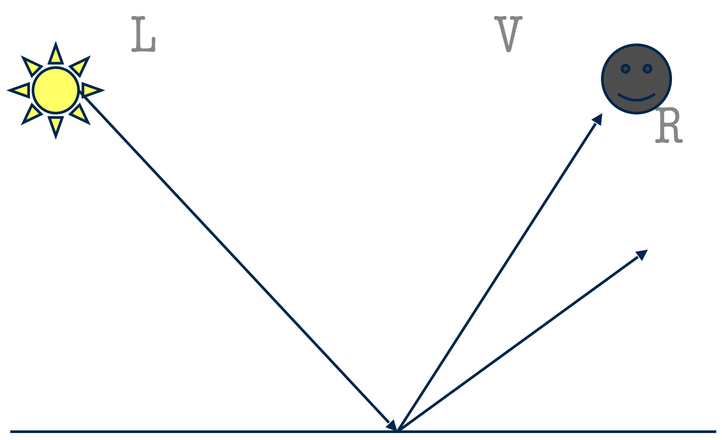


Diffuse Lighting

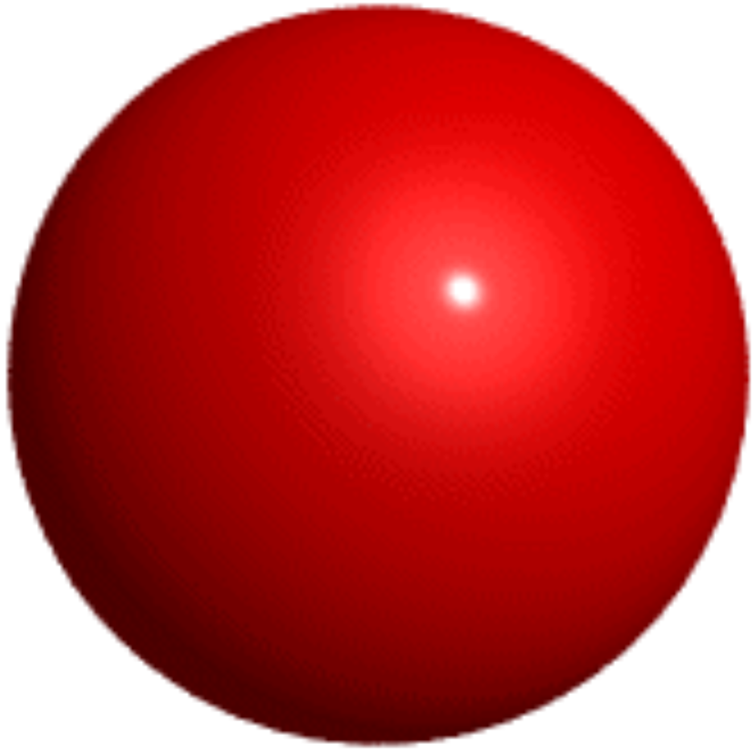
SPECULAR REFLECTION (PHONG MODEL)

- Incoming light is reflected primarily in the "mirror" direction R
  - Perceived intensity depends on the relationship between the viewing direction V and the mirror direction R
  - Bright spot is called a specular highlight
- Intensity controlled by:
  - The specular reflectance coefficient k_s
  - The parameter n controls the apparent size of the specular highlight (higher n, smaller highlight)

$$k_s I_i max(R \cdot V, 0)^n$$



SPECULAR EXAMPLE



Plus Specular Highlight

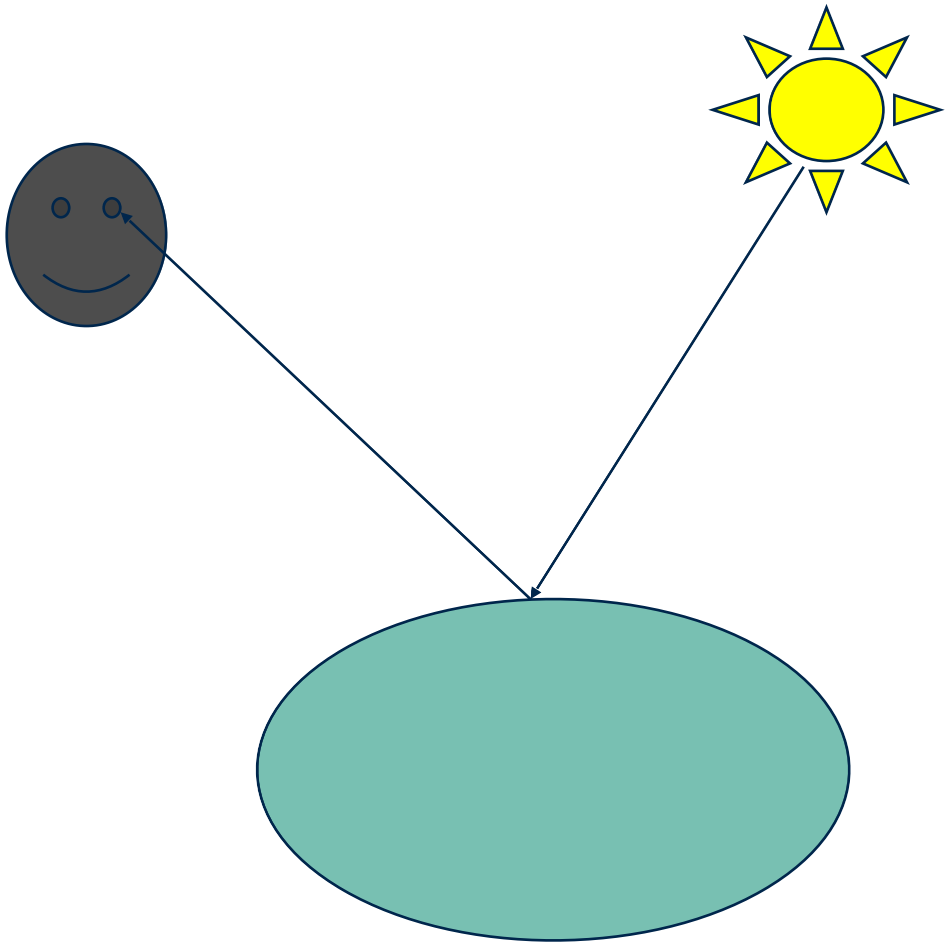
PUTTING IT TOGETHER

- Global ambient intensity, I_a:
  - Gross approximation to light bouncing around of all other surfaces
  - Modulated by ambient reflectance k_a
- Emitted term I_e – comes from object rather than reflected light
- Final lighting equation is the sum all the terms
  - If there are multiple lights, sum contributions from each light
  - Several variations, and approximations …

$$I = I_e + k_a I_a + \sum_{\text{lights } i} I_i \left(k_d (\mathbf{L}_i \bullet \mathbf{N}) + k_s (\mathbf{R}_i \bullet \mathbf{N})^n \right)$$

THE PHONG LIGHTING MODEL

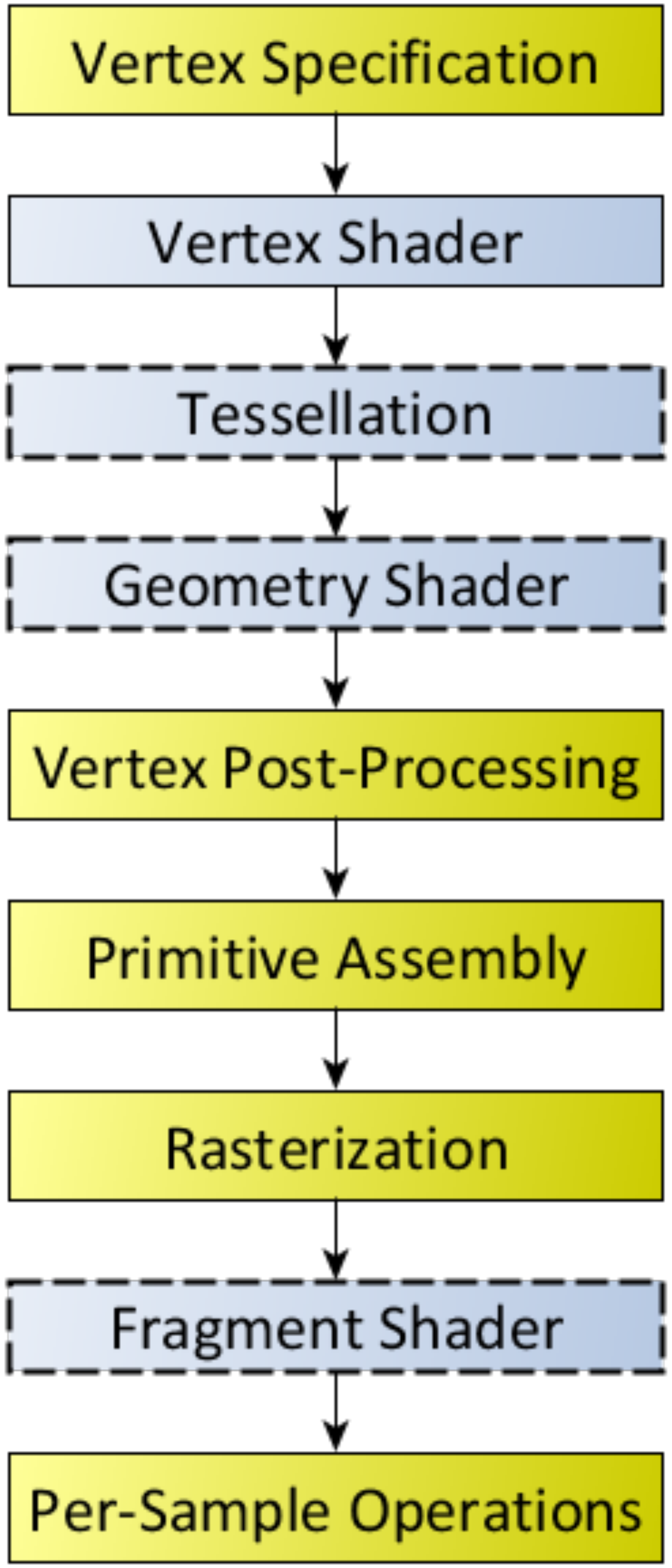
- A local shading model
  - Don't consider light interactions with other objects in the scene
  - Fast and simple to compute
- What they capture:
  - Direct illumination from light sources
  - Diffuse and Specular components
- What they don't capture:
  - Shadows
  - Mirrors
  - Refraction
  - Most of the pretty stuff



USING THE GRAPHICS PIPELINE

- GPU must process mesh vertices, material attributes, and all scene lighting sources
- Shader code (small, highly parallel programs that run on the GPU) process scene objects during rendering
  - OpenGL ES (mobile and web graphics library) supports vertex and fragment shaders
  - OpenGL library additionally supports geometry and tessellation shaders

RENDERING PIPELINE



HOW TO DISCRETIZE SHADING MODEL?

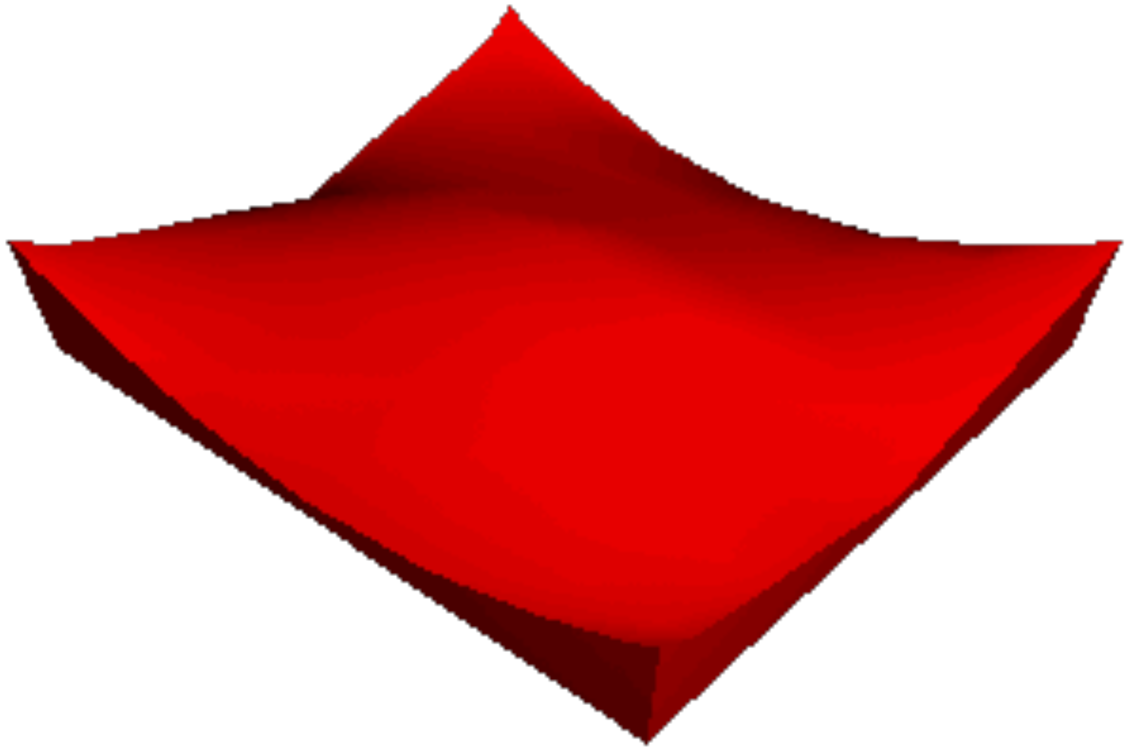
- We know material and lighting values at mesh vertices
  - How to light mesh surface from this?

FLAT SHADING

- Compute shading at a representative point and apply to whole polygon
  - Use one of the polygon vertices
- Advantages:
  - Fast - one shading value per polygon
- Disadvantages:
  - Inaccurate
  - Discontinuities at polygon boundaries

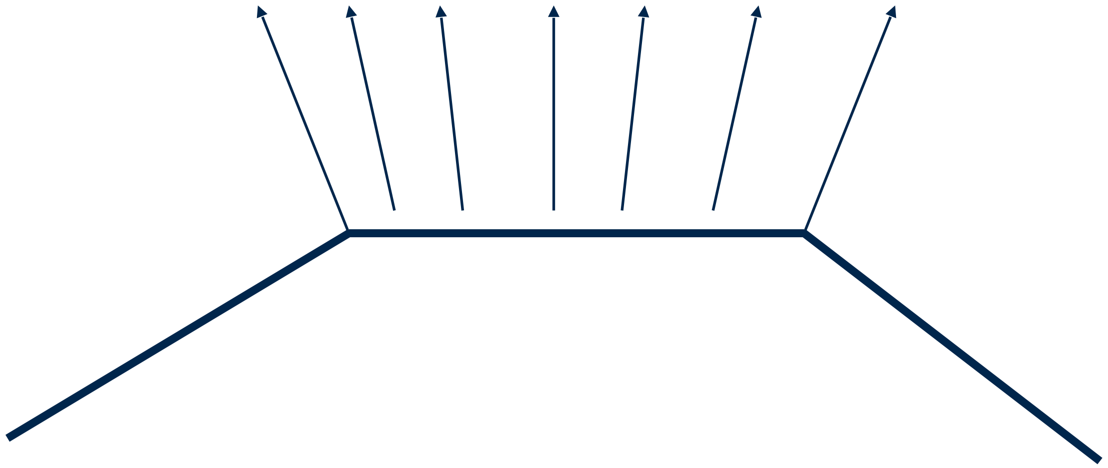
GOURAND SHADING

- Shade each vertex with its own location and normal
- Linearly interpolate across the face
- Advantages:
  - Fast - incremental calculations when rasterizing
  - Smoother - one normal per shared vertex creates continuity
- Disadvantages:
  - Specular highlights get lost



PHONG INTERPOLATION

- Interpolate **normals** across faces
- Shade each pixel
- Advantages:
  - High quality, narrow specular highlights
- Disadvantages:
  - Still an approximation for most surfaces
- Different from Phong Lighting

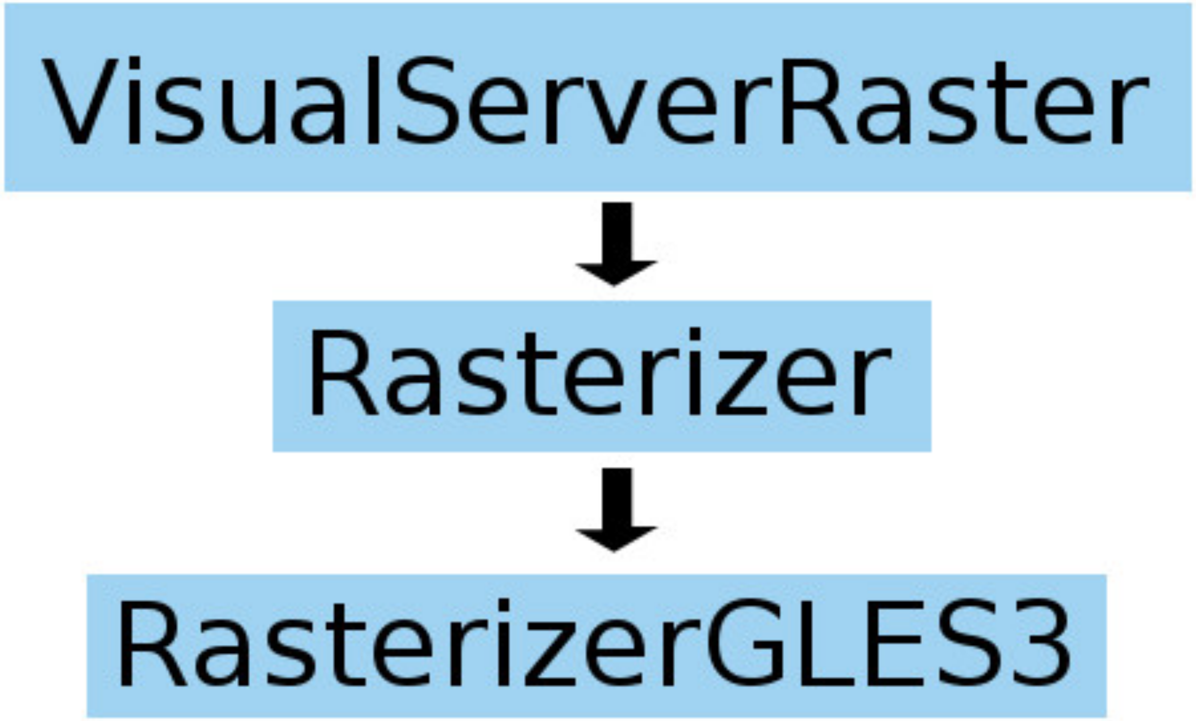


Gouraud

Phong

GODOT GRAPHICS PIPELINE

- VisualServerRaster handles spatial indexing of objects and builds a render list
- Rasterizer provides interface to graphics library



GODOT GRAPHICS PIPELINE

- Rendering happens in RenderingServer
  - Designed to abstract rendering process via API wall
  - Objects treated as RIDs (Resource IDs)
  - Allows modifications to renderer without affecting existing games
  - Separate rendering thread
- Connects to the Vulkan graphics API for low level, efficient rendering on modern hardware
- Connects to GLES3 (Graphics Library Embedded System 3) for web and mobile rendering

FORWARD RENDERING

- Godot uses **forward**, rather than deferred, rendering
- Forward rendering means scene geometry is passed down graphics pipeline and has shaders applied in sequence to it
- Deferred rendering composes scene geometry into a **texture** then applies lights and shading at end of pipeline
- Deferred versus forward:
  - Deferred used in most modern, commercial engines (we will come back to it later in the semester)
  - Forward works has more limitations on dynamic lights, but is conceptually simpler and works with multi-sample anti-aliasing (MSAA)

MORE MODERN GRAPHICS FEATURES IN GODOT 4

- Signed Distance Field Global Illumination
- Voxel Global Illumination
- Improved Shadow Maps
- Automatic Occlusion Culling and Mesh LODing
- Screen Space Indirect Lighting

BASIC GODOT RENDERING SEQUENCE

1. Depth-buffer pre-pass
2. Light setup
3. Opaque sort and render pass
4. Sky rendering
5. Screen space ambient occlusion (SSAO)
6. Subsurface scattering
7. Screenspace reflection
8. Transparent sort and render pass
9. Depth of field (DOF) blur
10. Exposure and bloom
11. Compositing

GODOT RENDERING SEQUENCE

- Can add or remove steps in sequence based on rendering configuration
- Effects hard-coded for efficiency
    - Designed to run well on low-end hardware