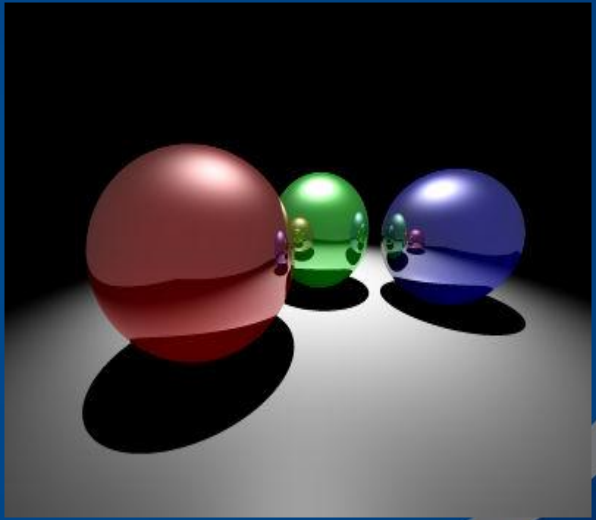
Lighting - Diffuse

Come and see the light!



Academy of Interactive Entertainment
www.aie.edu.au

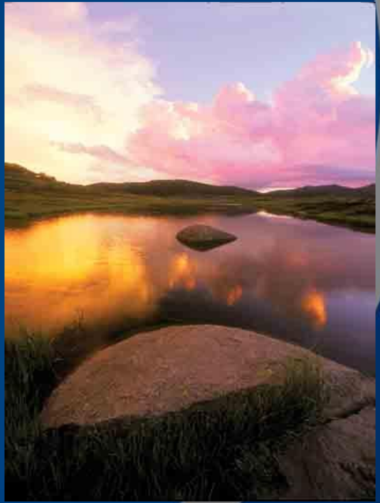
Canberra Institute of Technology
Workplace • Online • Campus | cit.edu.au

Lighting in Games

- Lighting plays a pivotal part in modern video games
- Lighting can be used to set the mood of a game, or direct the player down safe paths
- It increases the aesthetic of a scene and makes it more realistic
- Or it can be implemented in a way to make a scene unrealistic!

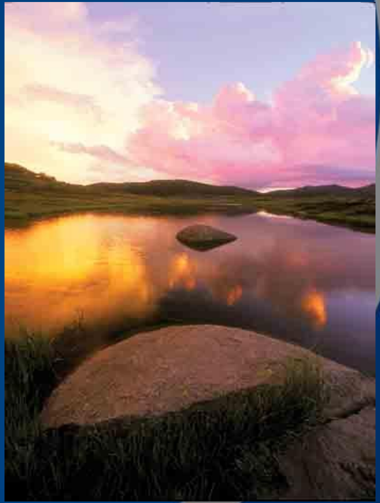
Lighting in Real Life

- In nature light is a stream of interacting photons
- A photon has a fixed wavelength and fixed frequency
  - Which determines light colour
- When light hits an object
  - Photons hit electrons in the object
  - Light beam is divided into
    - Photons absorbed - generating energy
    - Photons reflected – generating ray of light
    - Photons refracted - travel through object
- Light colours are simply the light reflecting from a surface



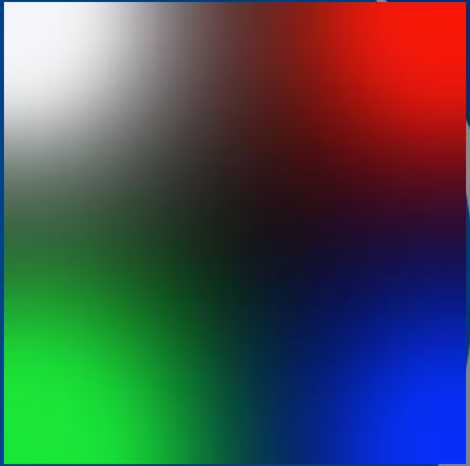
Lighting in Real Life

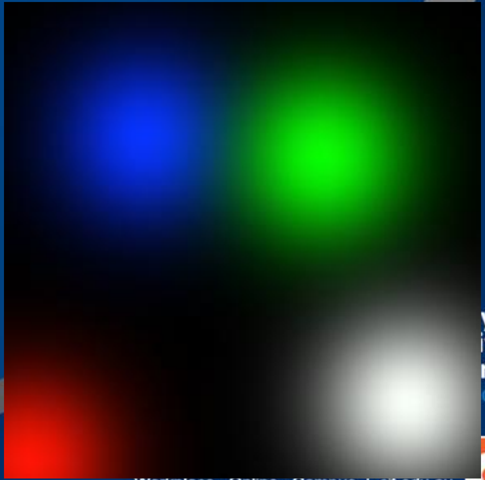
- A shadow is the occlusion of light
- Impossible to emulate real lighting on a computer in real-time
  - Have to simulate as best as possible
  - Shadows are implemented separate to lighting
- We can also use offline rendering algorithms that are slowly gaining real-time implementations:
  - Ray tracing
  - Global Illumination / Radiosity



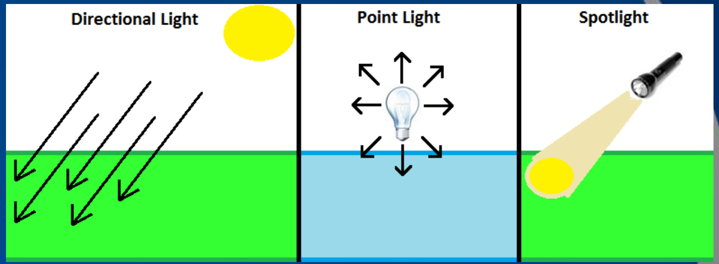
Lighting in Games

- In games we simulate lighting by calculating colours on visible pixels depending on if the light is shining on it
  - The colour can either be calculated per-vertex and interpolated by the Rasteriser across the pixels, or we can calculate it at the per-vertex level
- Per-Vertex Lighting
  - Light calculated in the Vertex Shader
  - Vertices must be illuminated by the light source
    - Light shining on the primitive but not covering any vertices won’t be calculated!
- Per-Pixel Lighting
  - Light calculated in the Fragment Shader
  - Each pixel calculates its own lighting rather than receiving interpolated light colour from the Rasteriser
  - Higher resolution lighting





Types of Lights



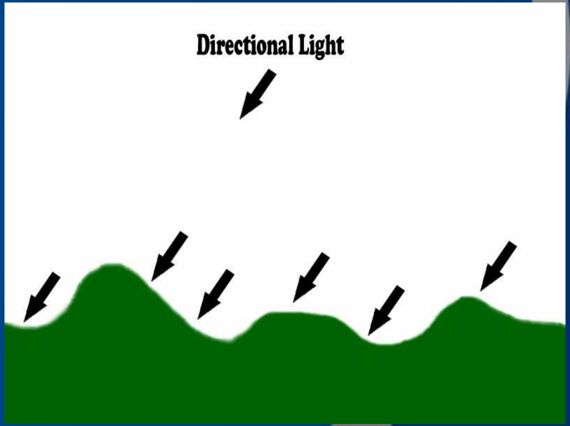
- There are a few different types of lights used in games:
- Ambient:
  - Represents the ambient reflected light in a scene when there is no light directly reflecting off a surface
- Directional:
  - Light travels globally in a set direction, with no single originating position
- Point:
  - Light emits from a single position outwards in all directions
  - Usually has a limited range or falloff
- Spot:
  - Light emits from a single position in a limited cone direction

Ambient Light

- Ambient Light is a way of simulating all of the reflected light in a scene that does not directly shine onto surfaces
  – An example could be the ambient sunlight in a house even though the Sun isn’t directly shining inside the house
  – Usually the entire scene uses the same ambient light

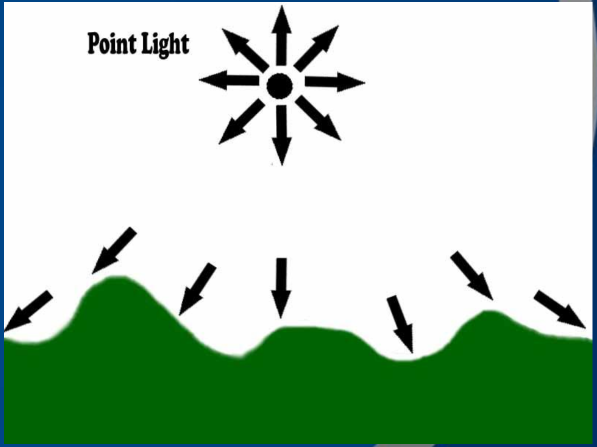
Directional Lights

- A directional light is typically a light that falls globally onto every surface in a set direction
- Most games treat the Sun as a directional light and it is usually the only directional light
- Any surface facing towards the direction the light is coming from can have light reflected from it



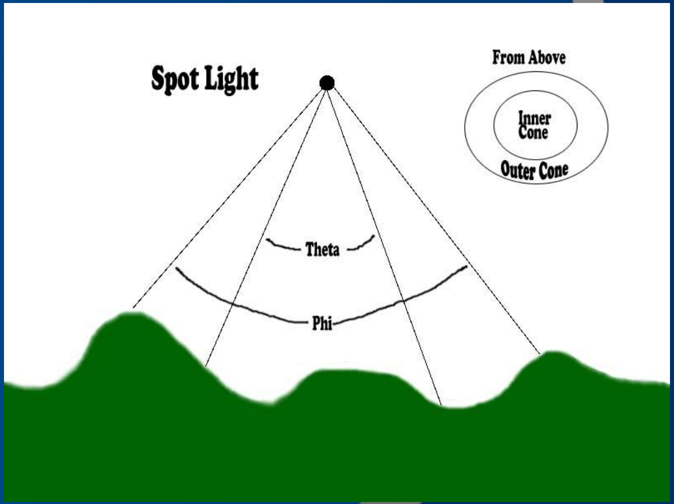
Point Lights

- Point lights are common in games, representing things such as fire, candles, explosions, light bulbs
- The direction that the light is traveling is determined by a vector between the surface position and the light position
- Point lights typically have a maximum distance, or falloff, controlled by an attenuation factor that reduces the light intensity based off distance to the surface



Spot Lights

- Spot lights can be used to represent street lamps, flashlights, car headlights
- Similar to a point light in that they have a position and a falloff, but also use two angles to determine the spot light's cone of influence
  - Theta and Phi
- Theta represents the inner angle of the light where there is no angle falloff
- Phi represents the outer angle of the spot light's influence
  - The spot light's light falls off between Theta and Phi

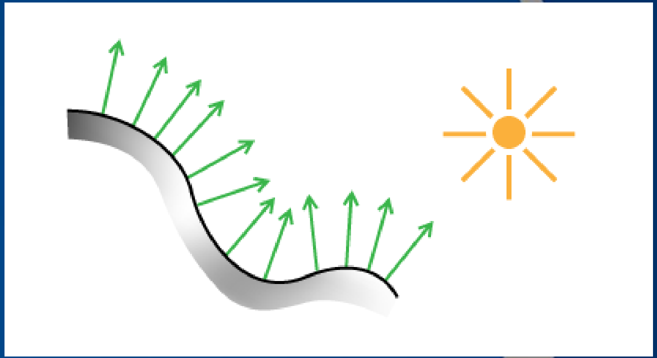


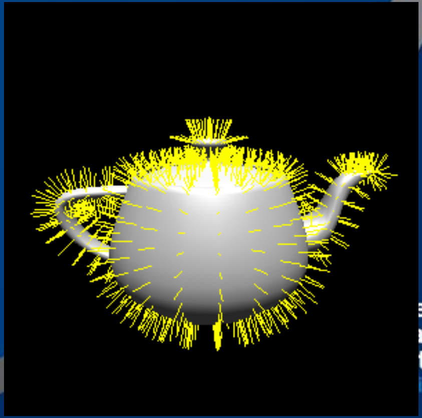
Surface Normals

- To correctly calculate lighting for a surface we need to know the direction the surface is facing so that we can calculate if light is shining on it
  - This direction is called the surface **normal**
  - All of our vertices in our vertex buffer need to contain additional information about the normal at each point

- This direction is then used to determine if a light is facing the surface

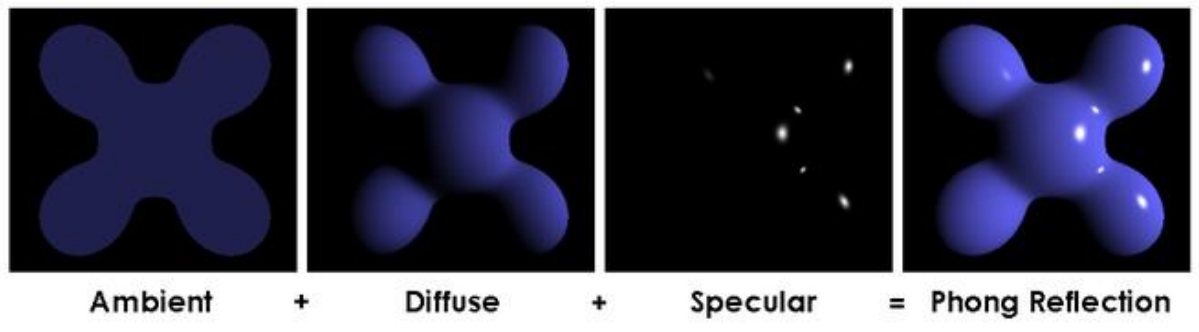
- A normal is typically a unit vector made up of XYZ and specifies a direction, not a position





Lighting Models

- Researchers have found many different ways to calculate lighting based on light properties and surfaces
  - Some simplify the equation for speed, not accuracy
  - Others are far more accurate but take a lot of processing
- By far the most common lighting model in use in computer graphics is Phong Lighting due to its efficiency and ability to closely simulate real lighting
  - Phong lighting takes 3 light properties, Ambient, Diffuse and Specular, and combines them with matching material properties to create a final colour

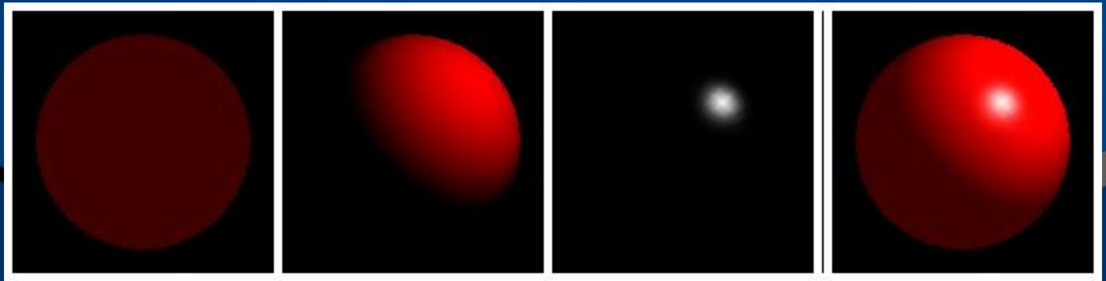


Ambient, Diffuse and Specular Properties

- There are 3 common colour properties of lighting
  - Ambient : already explained
  - Diffuse : colour of the reflected light reflected in such a way that the light is reflected at many angles
    - As if the light is shining off a rough surface
  - Specular : colour of the reflected light reflected as a single ray off the objects surface

- Typically both lights and surface materials have these properties
  - A material may have a blue diffuse colour but when light shines at a certain angle it could have a red specular colour
  - Similarly light may be yellow, for the sun, but could have an odd green specular colour

- Ambient material can be thought of the colour of an object when no light shines on it



Phong Lighting

- The mathematical model for Phong Lighting is:

$$I_p = k_a i_a + \sum_{m \in lights} (k_d (L_m \cdot N) i_d + k_s (R_m \cdot V)^a i_s)$$

  - That looks complex! But in practice it isn't…

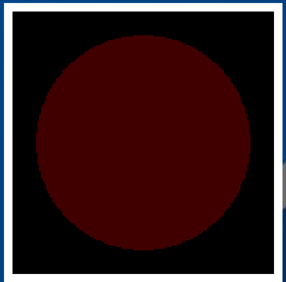
  - ***k*** refers to the surface's material property colours (***a***mbient, ***d***iffuse, ***s***pecular)
  - ***i*** refers to the light properties (***a***mbient, ***d***iffuse, ***s***pecular)
  - ***N*** is the surface normal vector
  - ***Lm*** is the light direction
  - ***Rm*** is a reflected light direction that we will discuss next session
  - ***V*** is a view direction that we will also discuss next session
  - ***a*** is a specular power that will again be discussed next session

Phong Ambient and Diffuse Equation

- First we will discuss how to implement the Ambient and Diffuse portions of the equation:

$$I_p = \underbrace{k_a i_a}_{\text{Ambient}} + \sum_{m \in lights} (\underbrace{k_d(L_m \cdot N)i_d}_{\text{Diffuse}} + \underbrace{k_s(R_m \cdot V)^a i_s}_{\text{Specular}})$$

- The Ambient portion is simple
  - We simply multiply the surface's ambient colour against the environment's ambient light colour
  - This can be further simplified by simply using the surface's diffuse colour



Example white ambient light with red ball

Phong Diffuse Equations

- Diffuse is slightly more complex

$$I_p = k_a i_a + \sum_{m \in lights} (\underbrace{k_d (L_m \cdot N) i_d}_{\text{Diffuse}} + k_s (R_m \cdot V)^a i_s)$$

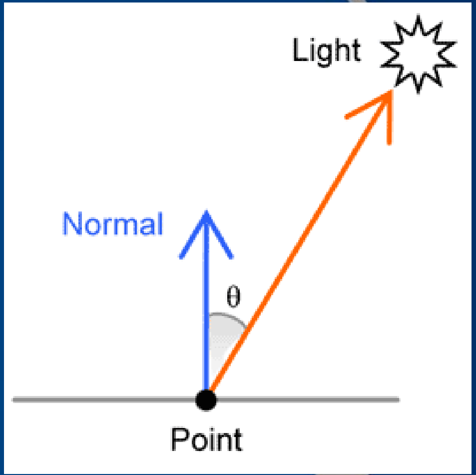
- We calculate it for every light and break the equation into 2 parts
  - Calculate the diffuse colour
  - Calculate Lambert's Cosine Law to determine how much light is reflecting off the surface
- Calculating the diffuse colour is as easy as the ambient
  - Multiply the surface's diffuse colour with the light's diffuse colour

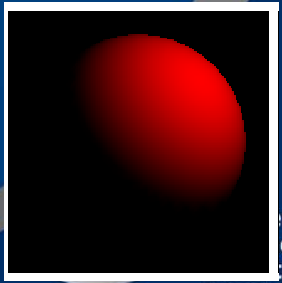
Lambert’s Cosine Law

- Lambert’s Cosine Law isn’t as scary as it sounds!

- To calculate the cosine (commonly called the Diffuse Term or Lambertian Term) you just:
  - Perform a dot product between the surface’s normal vector and a vector in the direction the light is coming from
  - Both vectors must be unit length
  - The value is then clamped between 0 and 1

- The final diffuse part of the equation is just this diffuse term multiplied against the combined diffuse colour
  - This effect shades the surface, lighting the part facing the light



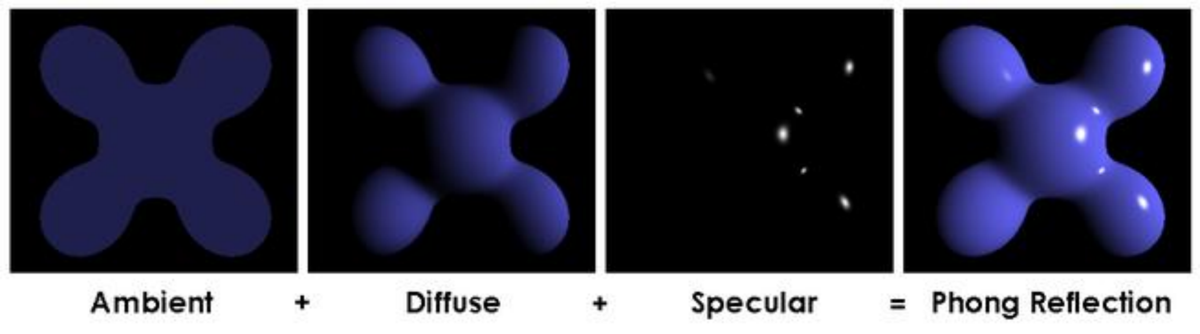


Combining the Ambient, Diffuse and Specular

- We will be covering the Specular property next session!

$$I_p = \underbrace{k_a i_a}_{\text{Ambient}} + \sum_{m \in lights} (\underbrace{k_d (L_m \cdot N) i_d}_{\text{Diffuse}} + \underbrace{k_s (R_m \cdot V)^a i_s}_{\text{Specular}})$$

- The final pixel colour for Phong Lighting is simply a combination of the Ambient colour and each of the Diffuse/Specular colours calculated for each light reflecting off a surface



Summary

- There are many models that implement lighting
  - Phong Lighting being the most common
- There are four common light types
  - Ambient, Directional, Point and Spot
- There are three common light and material properties
  - Ambient, Diffuse and Specular

http://en.wikipedia.org/wiki/Phong_reflection_model