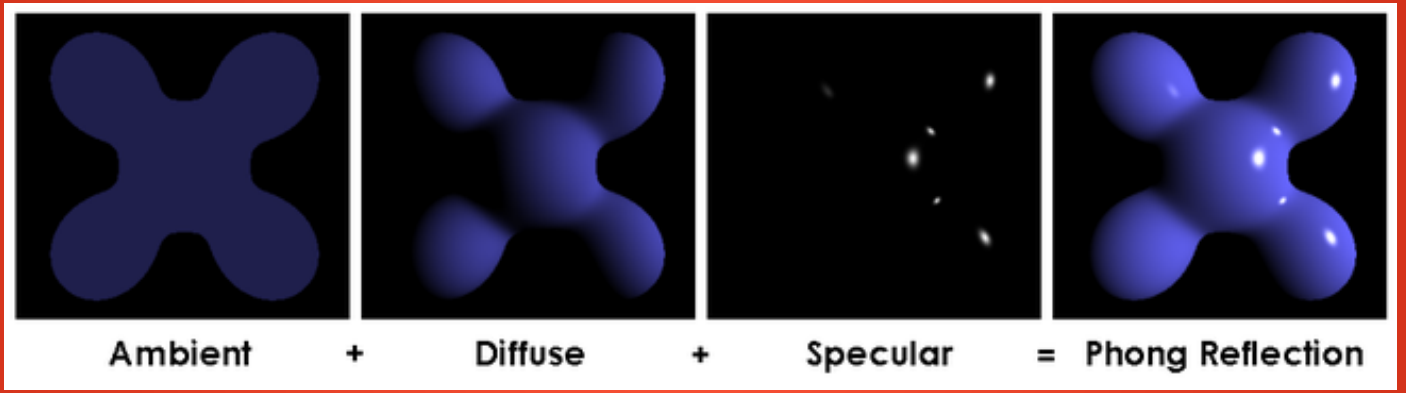
Phong Lighting and Directional Lights

Phong Lighting

- Consists of 3 parts:
  - Ambient Lighting: Light that is always present, even if a light source's direct path is blocked.
  - Diffuse Lighting: Light determined by direction of light source. Creates a faded effect further from the light.
  - Specular Lighting: Light reflected perfectly from the source to the viewer's eye. Effectively a reflection of the light source. More prominent on shiny object.
- Combined, they create the "Phong Lighting Model".

Phong Lighting



- Note: “Phong Reflection” is another name for Phong Lighting.

Ambient Lighting

- Simplest lighting concept.
- Simulates light bouncing off other objects.
- E.g. Create a shadow on the ground with your hand, using the sun… you can still see the colour in the shadow! It is still being lit.
- Global Illumination simulates this…
- Very advanced! Not necessary here.

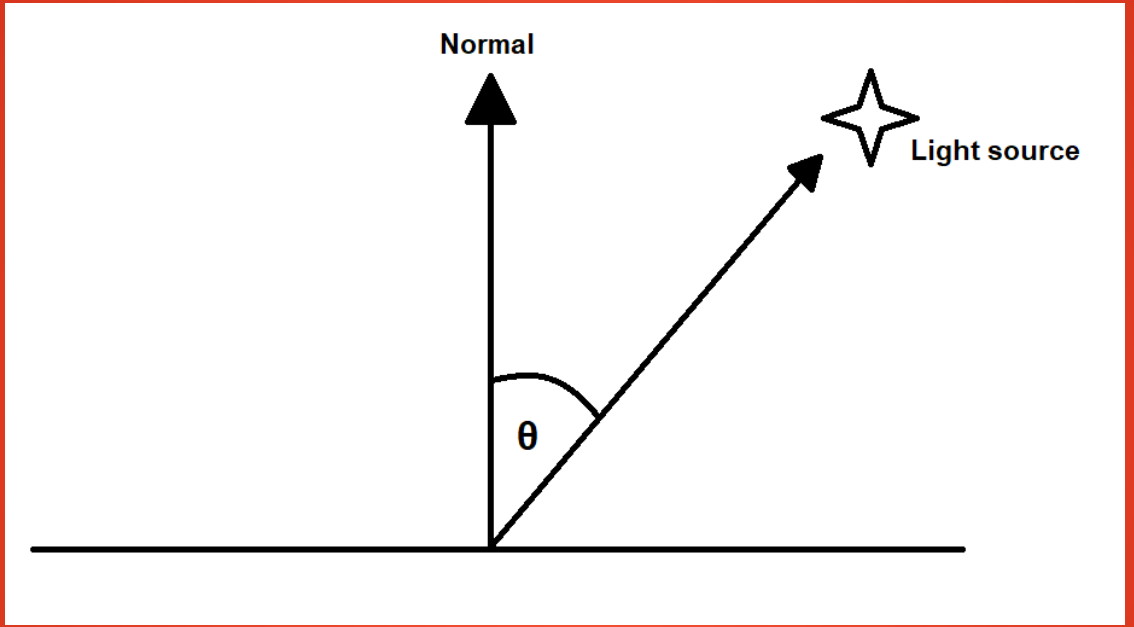
Ambient Lighting

- Create an ambient lighting factor.
- ambient = lightColour * ambientStrength;
- This factor is how much of a fragment’s colour this light’s ambient shows.
- fragColour = objectColour * ambient;
- If ambient is 1 (full power) then the fragment is always fully lit.
- If ambient is 0 (no power) then the fragment is always black.
- I ambient is 0.5 (half power) then fragment is half its normal colour.

Diffuse Lighting

- More complex!
- Simulates the drop-off of lighting as angle of lighting becomes more shallow.
- Side facing directly at a light is brightly lit.
- Side facing at an angle is more dim.
- Can use the angle between the vector connecting light source to fragment and the vector perpendicular to the face (surface “normal”).

Diffuse Lighting

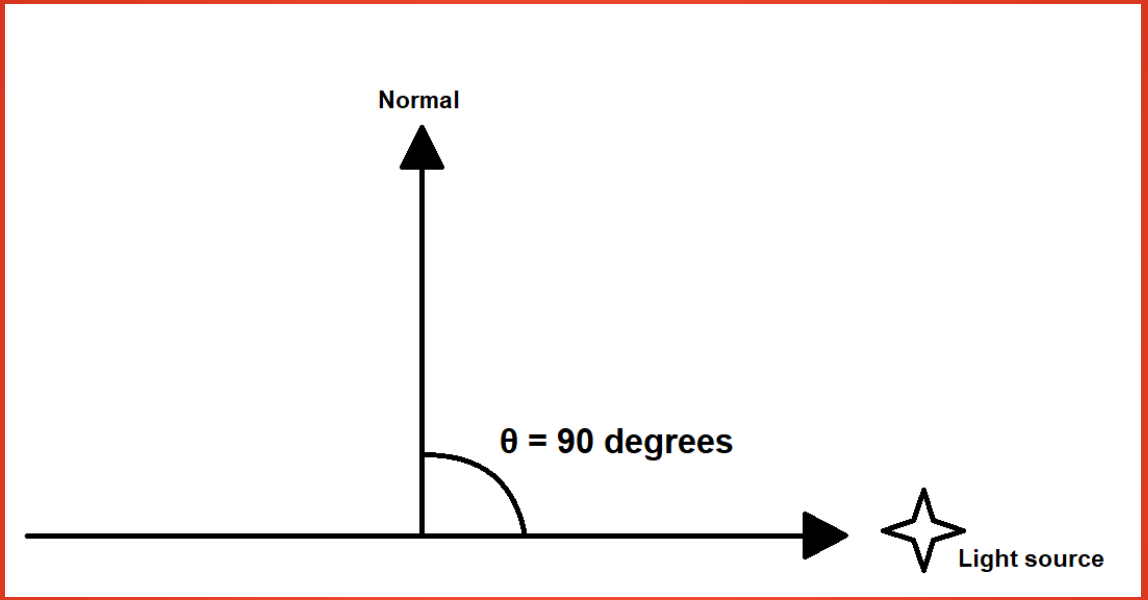


- Use θ to determine Diffuse Factor.
- Smaller θ: More light.
- Larger θ: More dim.

Diffuse Lighting

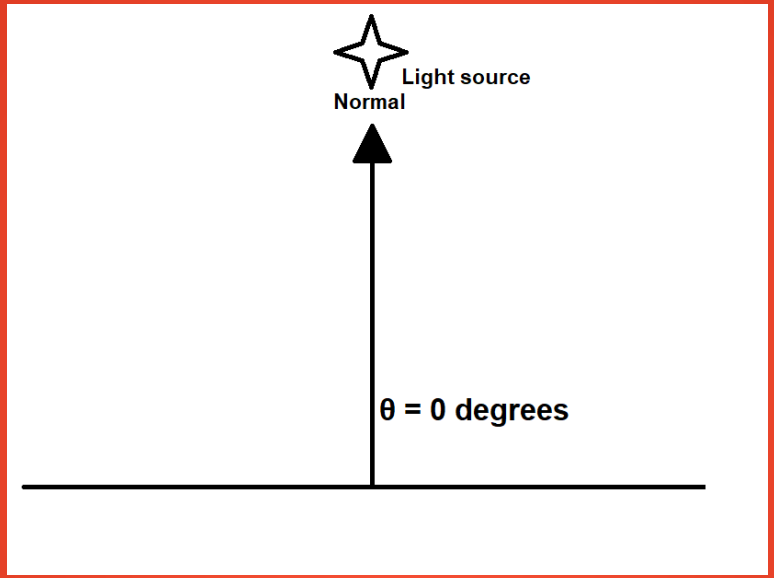
- Recall from Vectors: Dot Product.
- $v_1 \cdot v_2 = |v_1| \times |v_2| \times \cos(\theta)$
- If both vectors are normalized (converted to unit vectors…
- Then: $|v_1| = |v_2| = 1$
- So: $v_1 \cdot v_2 = \cos(\theta)$
- Since cos(0 degrees) = 1, and cos(90 degrees) = 0
- We can use the output of $v_1 \cdot v_2$ to determine a diffuse factor!

Diffuse Lighting



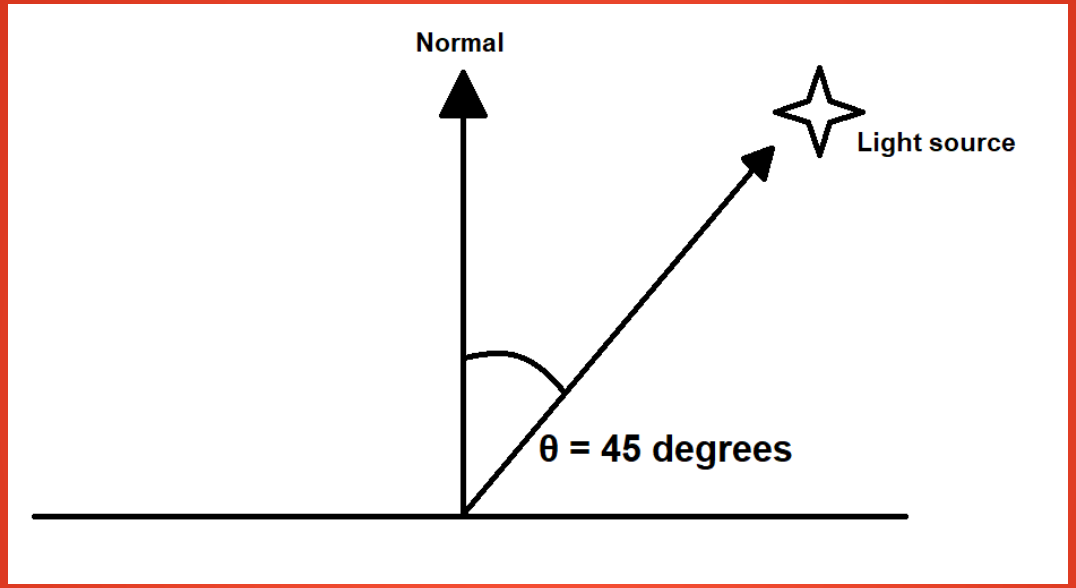
- Normalize normal and light vectors…
- $v_1 \cdot v_2 = \cos(\theta) = \cos(90 \text{ degrees}) = 0$
- Diffuse factor of 0 (0% diffuse lighting)!

Diffuse Lighting



- Normalize normal and light vectors…
- $v_1 \cdot v_2 = \cos(\theta) = \cos(0 \text{ degrees}) = 1$
- Diffuse factor of 1 (100% diffuse lighting)!

Diffuse Lighting



- Normalize normal and light vectors…
- $v_1 \cdot v_2 = \cos(\theta) = \cos(45 \text{ degrees}) = 0.71$
- Diffuse factor of 0.71 (71% diffuse lighting)!

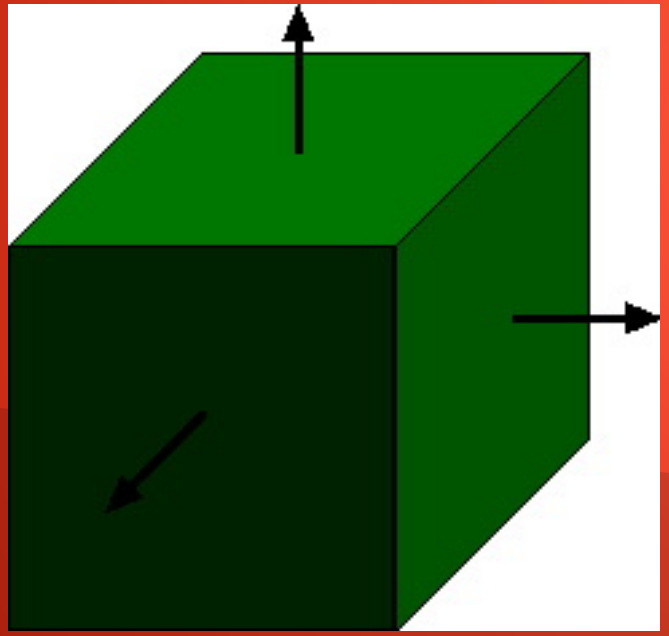
Diffuse Lighting

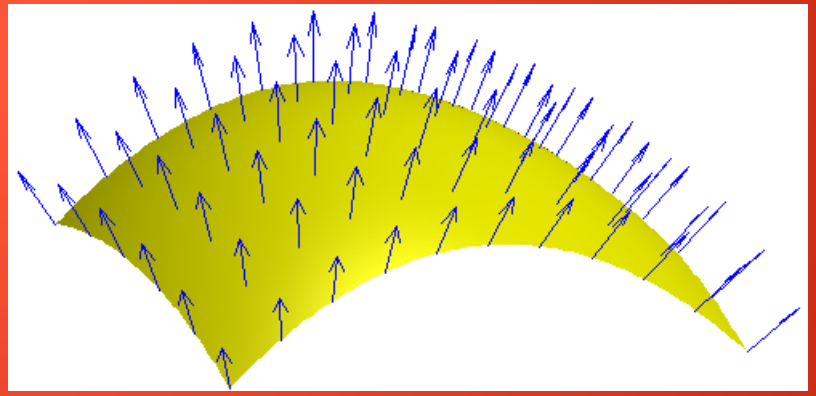
- If factor is negative (less than 0) then light is behind surface, so default to 0.
- Apply diffuse factor with ambient:

  fragColour = objectColour * (ambient + diffuse);

Diffuse Lighting - Normals

- Normals are vectors perpendicular to their point on a surface.





Diffuse Lighting - Normals

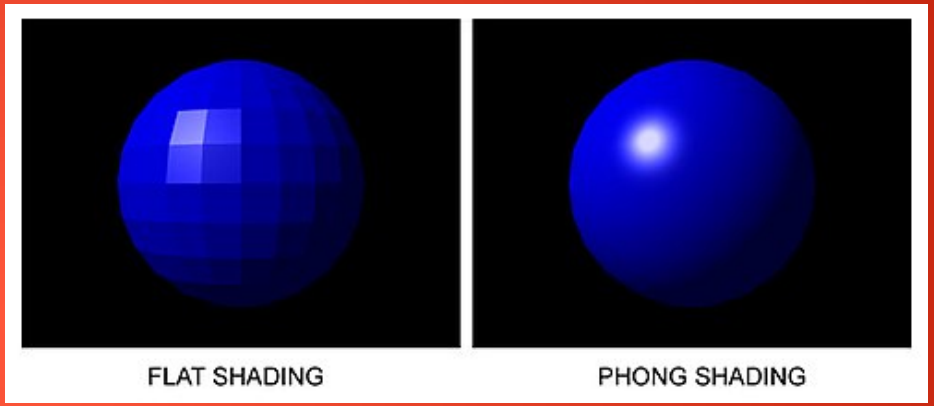
- Can define them for each face…
- Each vertex will have multiple normals, one for each face it is part of.
- Good for “Flat Shading”, not good for realistic smooth shading.
- Also doesn’t work too well for indexed draws: We only define each vertex once per face.

Diffuse Lighting - Normals

- Alternative: Phong Shading (not Phong Lighting!).
- Each vertex has an average of the normals of all the surfaces it is part of.
- Interpolate between these averages in shader to create smooth effect.
- Good for complex models!
- Not so good for simple models with sharp edges (unless you use some clever modelling techniques).

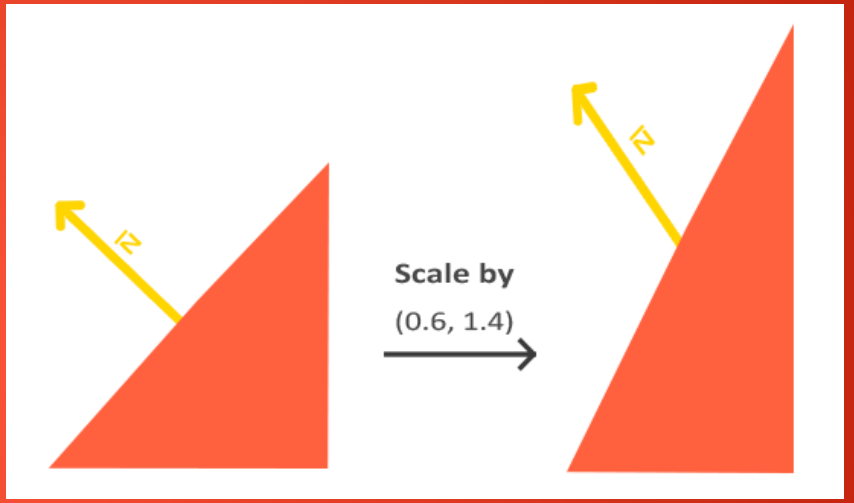
Diffuse Lighting - Normals

- Phong Shaded sphere is defined the same way as Flat Shaded.
- Smoothness is illusion created by interpolating and effectively “faking” surface normals to look curved.



Diffuse Lighting - Normals

- Problem with non-uniform scales.
- Wrongly skewing normals.
- Can be countered by creating a “normal matrix” from model matrix.

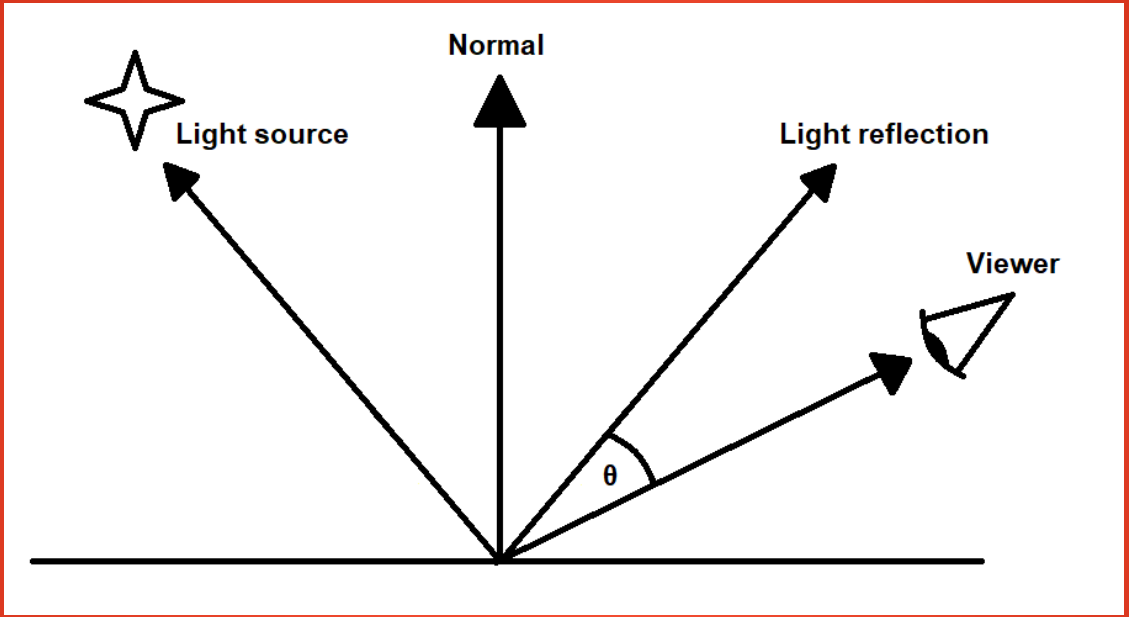


- Transform normals with: mat3(transpose(inverse(model)))
- Full explanation: http://www.lighthouse3d.com/tutorials/glsl-tutorial/the-normal-matrix/

Specular Lighting

- Specular relies on the position of the viewer.
- It is the direct reflection of the light source hitting the viewer’s eye.
- Moving around will affect the apparent position of the specular reflection on the surface.
- Therefore, we need four things:
  - Light vector
  - Normal vector
  - Reflection vector (Light vector reflected around Normal)
  - View vector (vector from viewer to fragment)

Specular Lighting



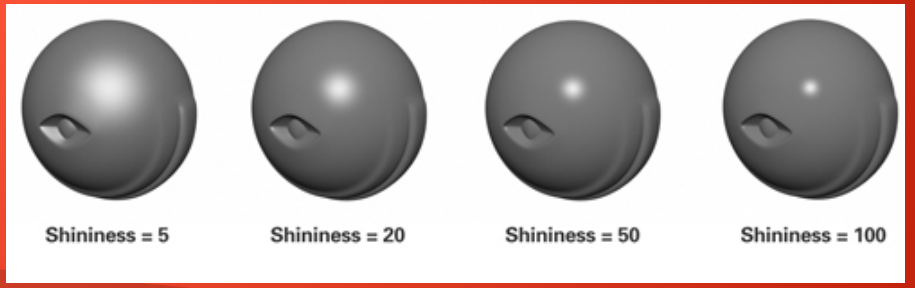
- Need the angle between viewer and reflection.
- Smaller θ: More light.
- Larger θ: More dim.

Specular Lighting

- View vector is just the difference between the Fragment Position and the Viewer (Camera) position.
- Reflection vector can be obtained with a built-in GLSL function: reflect(incident, normal)
  - Incident: Vector to reflect.
  - Normal: Normal vector to reflect around.
- Just as with diffuse, use dot product between normalized forms of view and refection vector, to get specular factor.

Specular Lighting - Shininess

- One last step to alter Specular Factor: Shininess.
- Shininess creates a more accurate reflection.
- Higher shine: Smaller more compact specular.
- Lower shine: Larger, faded specular.
- Simply put previously calculated specular factor to the power of shininess value.
- $\text{specularFactor} = (\text{view} \cdot \text{reflection})^{\text{shininess}}$



Specular Lighting

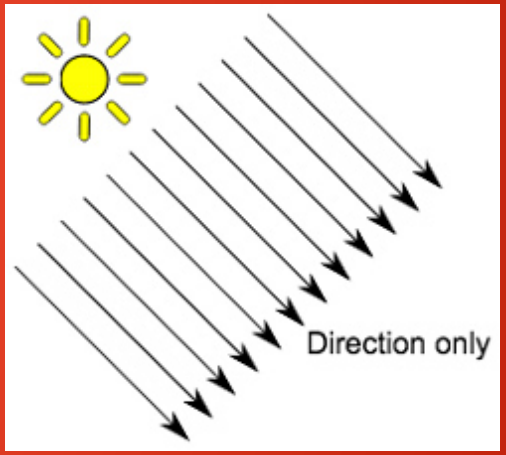
- Then apply Specular Factor as with ambient and diffuse…
- fragColour = objectColour * (ambient + diffuse + specular);
- ...but where is this light coming from?

Types of Light

- Directional Light: A light without a position or source. All light is coming as parallel rays from an seemingly infinite distance. Best example is the Sun.
- Point Light: A light with a position that shines in all directions. Best example is a lightbulb.
- Spot Light: Similar to a Point Light, but cut down to emit in a certain range, with a certain angle. Best example is a flashlight.
- Area Light: More advanced light type that emits light from an area. Think of a large light up panel on a wall or ceiling.

Directional Light

- Simplest form of light!
- Only requires basic information (colour, ambient, diffuse, specular…) and a direction.
- Treat all light calculations using the same direction for the light vector.
- Don’t need to calculate a light vector!



Summary

- Phong Lighting Model combines Ambient, Diffuse and Specular lighting.
- Diffuse and Specular require normals.
- Use dot product and normals to determine diffuse lighting.
- Use dot product and light reflected around normals to determine specular lighting.
- Phong Shading interpolates averaged normals to create smooth surfaces.
- Four main types of light: Directional, Point, Spot and Area.
- Directional Light is easiest, since it only requires a direction and allows us to skip calculating a light vector!

See you next video!