COMP220: Graphics & Simulation

6: Materials and Lighting

Learning outcomes

- **Explain** the Phong illumination model
- **Implement** Phong illumination in your own programs
- **Describe** how effects such as normal mapping can be used to render realistic materials

Vector products

Dot and cross product

$$a \cdot b = |a||b| \cos \theta$$

where θ is the **angle** between a and b

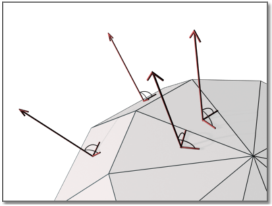
$$a \times b = (|a||b| \sin \theta) n$$

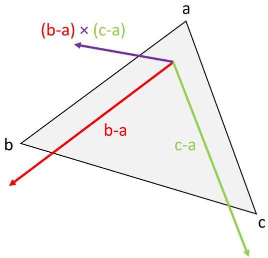
where n is a unit vector **perpendicular** to both a and b with direction given by the **right-hand rule**

Uses

- Both dot and cross product are **quick to calculate**
- Dot product can be used to find the **angle** between vectors
  - Actually the **cosine** of the angle
  - If $a \cdot b = 0$ (and a, b are non-zero) then $\cos\theta = 0$, i.e. $\theta = 90^\circ$ — a and b are **perpendicular**
  - If $a \cdot b = 1$ and a, b are unit vectors then $\cos\theta = 1$, i.e. $\theta = 0^\circ$ — a and b are **parallel**
- Cross product can be used to find a vector **perpendicular** to two others
- vector $\cdot$ vector $=$ number; vector $\times$ vector $=$ vector

Surface normals





- The **normal** to a surface is a **unit vector** that is **perpendicular** to the surface
- If we have two non-parallel vectors that are **tangent** to the surface, we can use the **cross product** to find the normal
- For a triangle with vertices a, b, c, two such vectors are $b - a$ and $c - a$
- So the normal is

$$\frac{n}{|n|} \quad \text{where} \quad n = (b - a) \times (c - a)$$

Passing normals to OpenGL

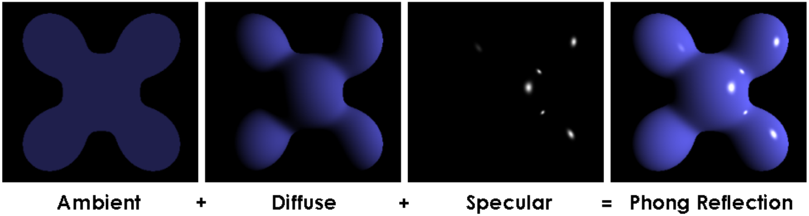
- We will pass normals as **vertex attributes**
- For now all vertices of a triangle have the same normal, but this will change later

The Phong illumination model

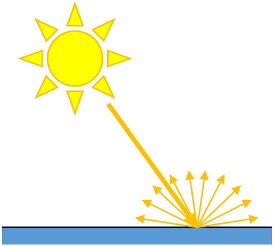
The Phong illumination model

Bui Tuong Phong, “Illumination for Computer Generated Pictures”. *Communications of the ACM*, 18(6):311–317, 1975.

The Phong model breaks lighting down into three parts: **ambient**, **diffuse**, and **specular**.

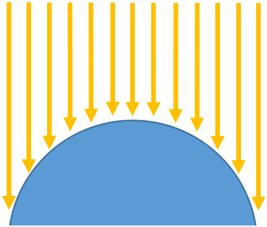


Diffuse lighting

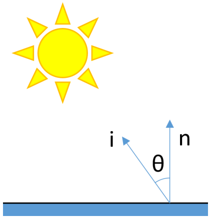


When light hits a “”rough surface, it is **scattered equally in all directions**

The amount of light hitting the surface depends on the **angle** between the surface and the light source



Diffuse lighting formula

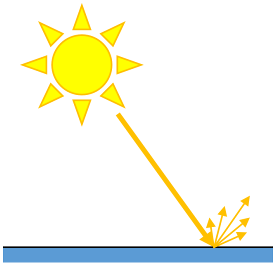


- Light intensity is proportional to the **cosine** of the angle between the **light direction** and the **surface normal**
- Let n be the normal, and i be a unit vector pointing towards the light source
- Light intensity is proportional to $\cos\theta = n \cdot i$
- If the surface is **pointing away** from the light source, we get $\theta > \frac{\pi}{2}$ so $\cos\theta < 0$ — in this case we **clamp** the answer to 0

Light direction and intensity

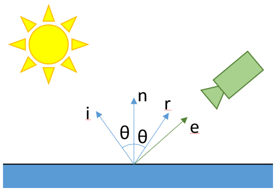
- For a **distant** light source (e.g. the sun), direction and intensity are **constant**
- For a **point** light source (e.g. a lightbulb):
  - Direction is calculated by subtracting the light position from the fragment position
  - Intensity obeys an **inverse square law**: if the distance between the fragment and the light source is d, then the light intensity is $\frac{1}{d^2}$

Specular lighting



When light hits a "smooth" surface, it is **reflected** across a narrow range of angles

Specular lighting formula



- Let r be the reflection angle (can be calculated in GLSL by `reflect(-i, n)`)
- e is a unit vector pointing from the surface towards the camera
- Specular light intensity is proportional to

$$\text{clamp}(e \cdot r)^s$$

where s is a "shininess" parameter, and clamp(x) clamps its argument between 0 and 1

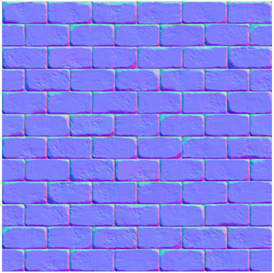
Ambient lighting

- Currently, surfaces pointing away from the light are completely **black** (light intensity = 0)
- In the real world, light scattered from one surface illuminates others
- In the Phong model, we cheat and add a little **ambient** intensity to the lighting
- Another option would be to add more light sources...

Normals revisited

- Currently, all points on a triangle have **the same** normal
- This gives the triangles a “flat” look, which may be what we want
- Using different normals for the vertices can give a “curved” look

Normal mapping



- A **normal map** is a texture which is used to slightly alter the normal across a surface
  - Each pixel in the normal map represents a 3D vector, with *xyz* mapped to RGB
- Can be used to add detail to flat, low-poly surfaces
- Can use textures to change other lighting parameters across a surface, e.g. **specular mapping**

Sprint review