# COMP3320 Introduction to OpenGL

Alex Biddulph

The University of Newcastle, Australia

Based on the work provided at `www.learnopengl.com`

Semester 2, 2021

## Object Colour

- The colour that an object reflects
- This can be simulated as a simple multiplication

```
vec3 light_colour  = vec3(1.0f, 1.0f, 1.0f);
vec3 object_colour = vec3(1.0f, 0.5f, 0.31f);
// This is a component-wise multiplication
vec3 result        = light_colour * object_colour;
```

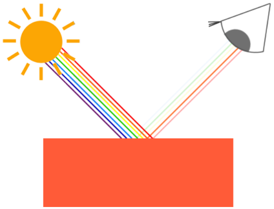
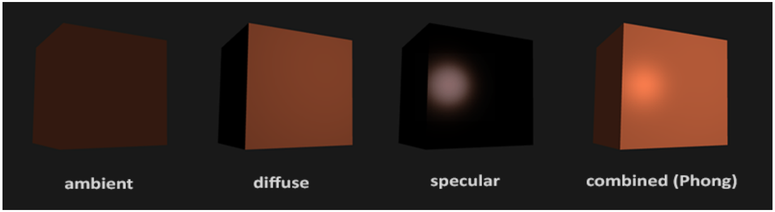


Figure: Image sourced from learnopengl.com/Lighting/Colors

## Basic Lighting



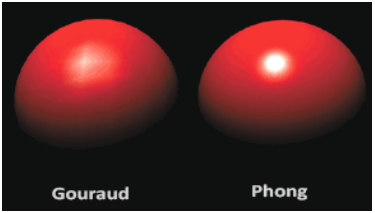
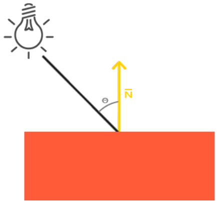


Figure: Images sourced from `learnopengl.com/Lighting/Basic-Lighting`

## Basic Lighting

- Ambient: Background/global lighting. Results in objects being dimly lit when all other lights are turned off.
- Diffuse: Brightness of reflected light is dictated by how closely the fragments normal vector aligns with the light direction.
- Specular: Light is reflected about the fragments normal vector. Light appears brightest when the viewing direction most closely aligns with the reflected direction.



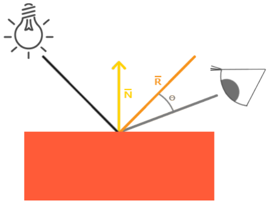


Figure: Images sourced from `learnopengl.com/Lighting/Basic-Lighting`

## Basic Lighting Equations

- Ambient:

```
vec3 ambient = ambient_strength * light_colour;
vec3 result  = ambient * object_colour;
```

- Diffuse:

```
vec3 norm           = normalize(frag_normal);
vec3 light_dir      = normalize(light_pos - frag_pos);
float diff_strength = max(dot(norm, light_dir), 0.0f);
vec3 diffuse        = diff_strength * light_colour;
vec3 result         = diffuse * object_colour;
```

## A Couple of Notes

### A Note on Content

The next slide contains a reference to homogeneous transformations, rotations, and matrices. If you don't know what these are or need a refresher be sure to check the Transformations lecture.

### A Note on Notation

The next slide uses a different notation for homogeneous transformations that is not used in the Transformations lecture. If you would like more information on this notational style see ☞NUbook

## Non-Uniform Object Scaling

- If objects are not scaled uniformly then normals can point in strange directions
- To account for this use the inverse transpose of the model-view rotation matrix
- If $H_m^v$ is a homogeneous transformation matrix that transforms vectors from model space to view space, then $R_m^v$ is the top-left 3x3 corner of the model-view matrix and forms the rotation component of the transformation

$$H_m^v = \begin{bmatrix} \boldsymbol{R}_m^v & \vec{\boldsymbol{r}}_{MV}^v \\ \vec{\boldsymbol{0}}_{1x3} & 1 \end{bmatrix}$$

- To account for non-uniform object scaling we can multiply our vertex normals by

$$\boldsymbol{N}_m^v = \left( (\boldsymbol{R}_m^v)' \right)^T$$

- This is the inverse transpose of $\boldsymbol{R}_m^v$ and is known as ☞the normal matrix

## Basic Lighting Equations

- Specular:

```
vec3  norm        = normalize(frag_normal);
vec3  light_dir   = normalize(frag_pos - light_pos);
vec3  view_dir    = normalize(view_pos - frag_pos);
vec3  reflect_dir = reflect(light_dir, norm);
float strength    = pow(max(dot(view_dir, reflect_dir), 0.0f), 32.0f);
vec3  specular    = 0.5f * strength * light_colour;
vec3  result      = specular * object_colour;
```

Shininess

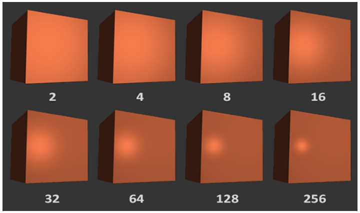
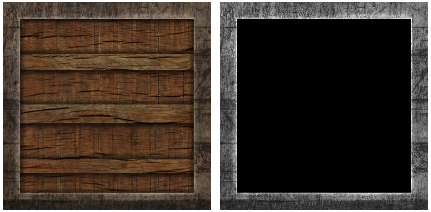


Figure: The effect of `strength` (the shininess factor) on the specular highlight. Images sourced from `learnopengl.com/Lighting/Basic-Lighting`

## Lighting Maps

- Use textures to provide object colour per fragment.
- Use textures to specify which areas of an object give specular reflections



- Use a single uniform to provide material properties per object

```
struct Material {
    sampler2D diffuse;
    sampler2D specular;
    float shininess;
};
in vec2 texture_coordinates;
uniform Material material;
```

- Use the equations as before, but sample the appropriate texture to get `object_colour`.

## Types of Light

- Directional: Light source is very far away. When the light reaches the object light rays are basically parallel to each other.
- Point Light: A nearby light that illuminates equally in all directions.
- Spot Light: A nearby light that illuminates in a single direction.

## Light Attenuation

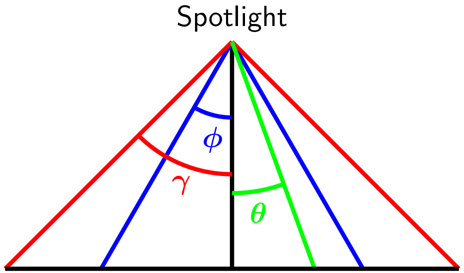
- Light intensity drops off over distance
- A common formula for attentuation is

$$F_{att} = \frac{1}{K_c + K_l d + K_q d^2}$$

- $K_c$ represents a constant amount of attentuation regardless of listance
- $K_d$ represents a attentuation that is linearly proportional to distance
- $K_q$ represents a attentuation that is quadratically proportional to distance
- Use trial and error to find some good values or use some ☞ pre-calculated values

## Spotlight Smoothing

- Represent spotlight as a cone
- If light stops at edge of cone there will be a hard line between light/no light
- Instead represent spotlight with two cones and fade spotlight intensity between them
- $\theta$ is the angle between the spotlight direction and the fragment position
- $\phi$ and $\gamma$ are the angles between the spotlight direction and the inner and outer cones



$$I = \frac{\cos(\theta) - \cos(\gamma)}{\cos(\phi) - \cos(\gamma)}$$

## Types of Light

- Directional:

```
struct DirectionalLight {
    vec3 direction;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
uniform DirectionalLight sun;
```

- Point Light:

```
struct PointLight {
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;

    float Kc;
    float Kl;
    float Kq;
};
```

## Types of Light

- Spot Light:

```
struct SpotLight {
    vec3 position;
    vec3 direction;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;

    float Kc;
    float Kl;
    float Kq;

    float phi;
    float gamma;
};
uniform SpotLight torch;
```