COMP3320 Introduction to OpenGL

Alex Biddulph

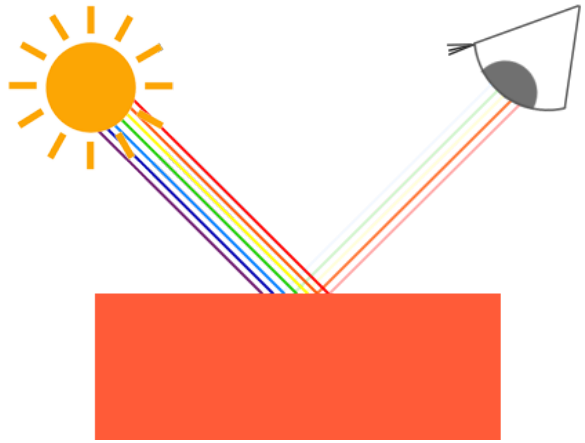
The University of Newcastle, Australia

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

Semester 2, 2019

Object Colour

- The colour that an object reflects

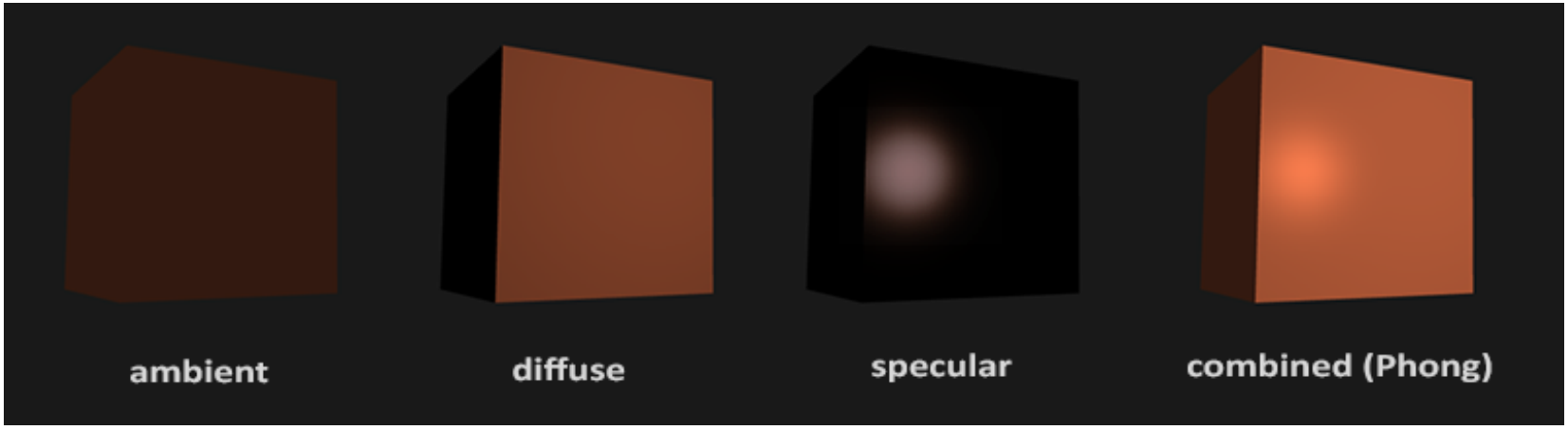


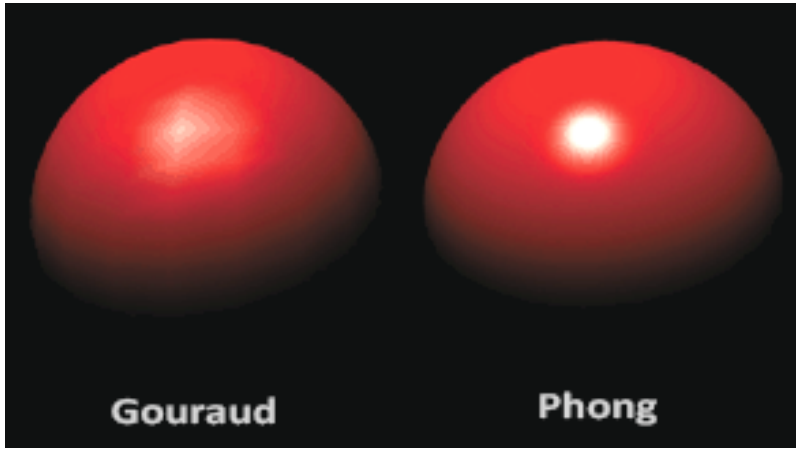
- This can be simulated as a simple multiplication

```
vec3 light_colour  = vec3(1.0f, 1.0f, 1.0f);
vec3 object_colour = vec3(1.0f, 0.5f, 0.31f);
vec3 result        = light_colour * object_colour;
```

Image sourced from `learnopengl.com/Lighting/Colors`

Basic Lighting

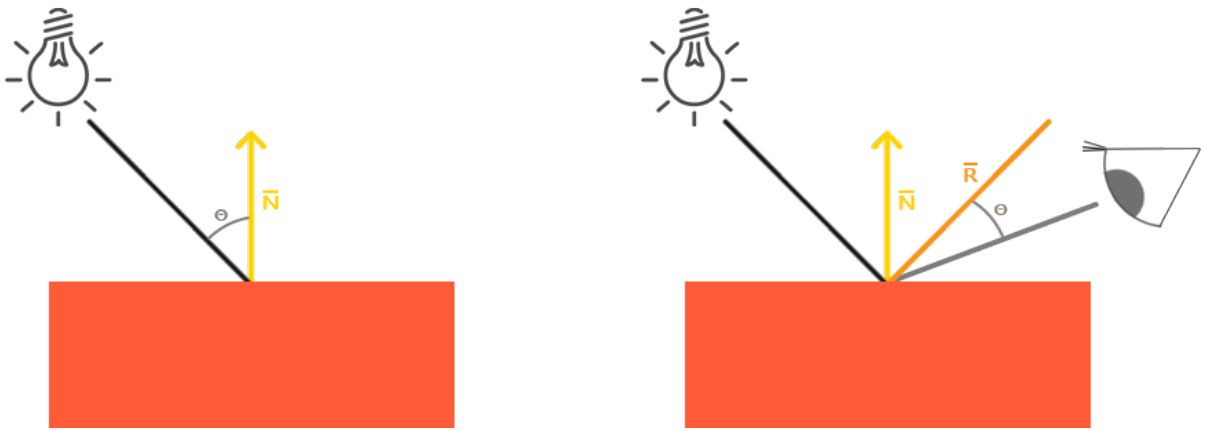




Images sourced from `learnopengl.com/Lighting/Basic-Lighting`

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.



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

Basic Lighting Equations

- Ambient:

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

- Diffuse:

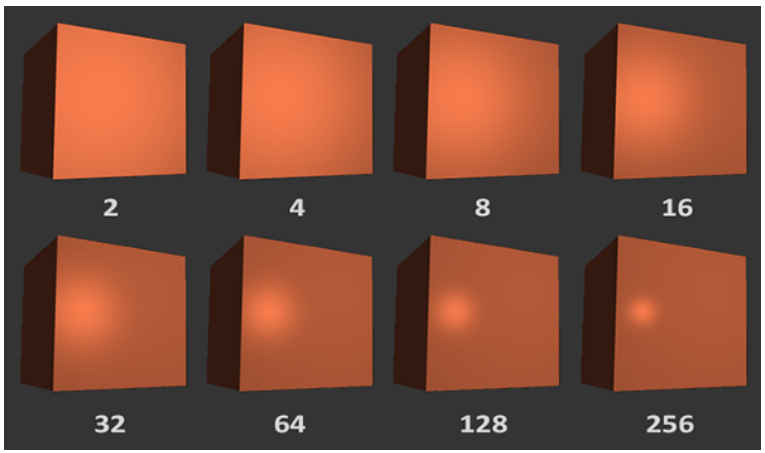
```
vec3 norm = normalize(frag_normal);
vec3 light_dir = normalize(light_pos - frag_pos);
float diff_strength = max(dot(norm, light_dir), 0.0f);
vec3 diffuse = diff_strength * light_colour;
vec3 result = diffuse * object_colour;
```

- Non-Uniform Scaling: If objects are not scaled uniformly then normals can point in strange directions. To account for this use $\vec{n}_f = \left(M^{'}\right)^{T} \vec{n}_a$

Basic Lighting Equations

- Specular:

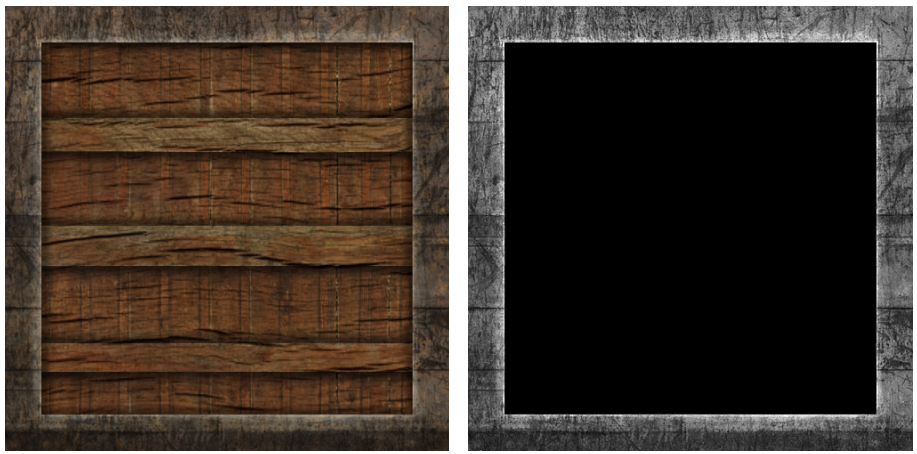
```
vec3 norm = normalize(frag_normal);
vec3 light_dir = normalize(frag_pos - light_pos);
vec3 view_dir = normalize(view_pos - frag_pos);
vec3 reflect_dir = reflect(light_dir, norm);
float spec_strength = max(dot(view_dir, reflect_dir), 0.0f);
spec_strength = pow(spec_strength, 32.0f);
vec3 specular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



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`.

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

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.
- Attentuation: Intensity of a light drops off over distance. A common formula for attentuation is

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

- Smoothing: Spotlight intensity fades smoothly outwards

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

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;
};
uniform PointLight ceiling_light;
```

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;
```