Intro to textures and lighting

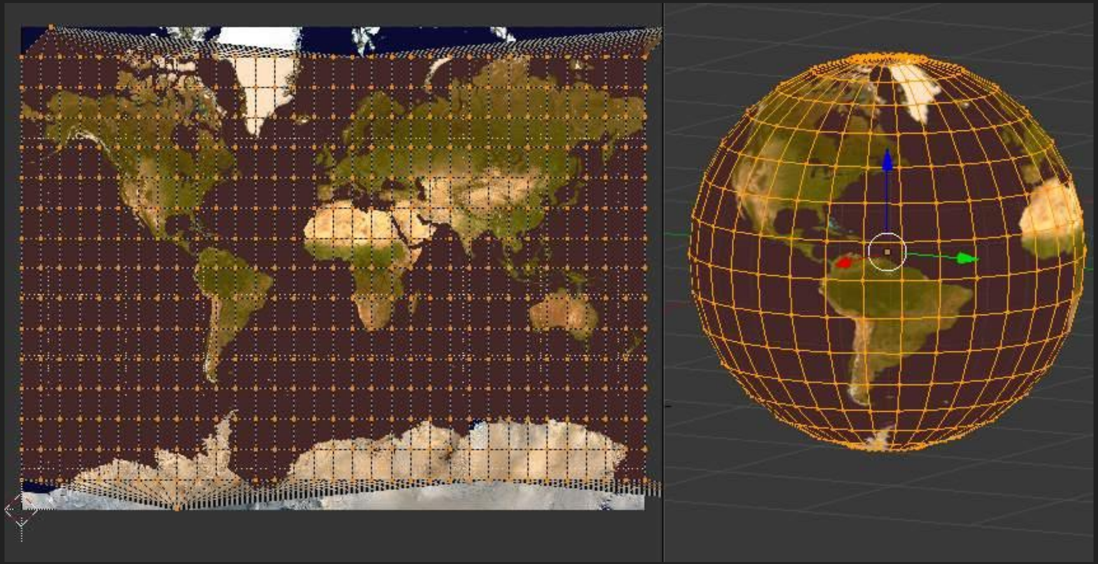
CM163 Lab 2

Today's Lab

- Texture mapping
- Basic lighting models
    - Flat shading
    - Gouraud shading
    - Phong Shading

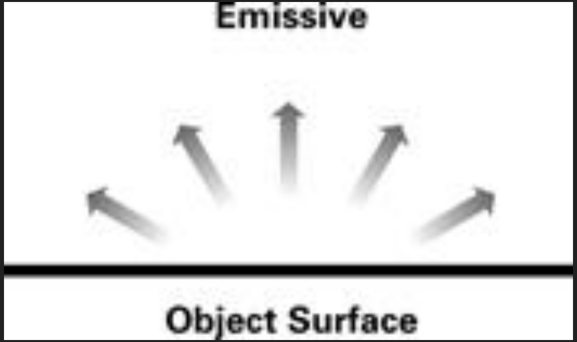
Texture Mapping

- Texture Coordinate (UV space)
- Goes from 0 to 1
- Every vertex of a mesh has a location in this UV space



Basic Lighting Model

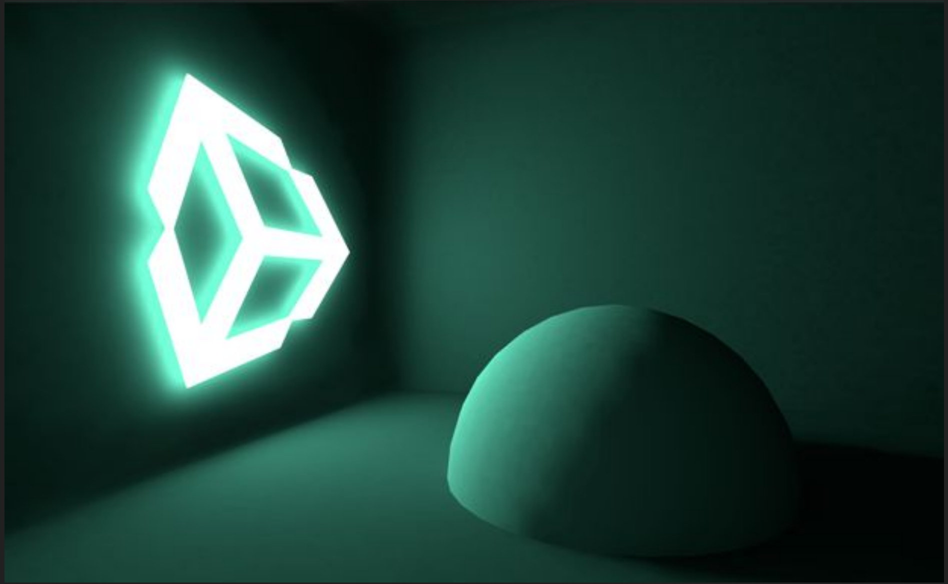
surfaceColor = emissive + ambient + diffuse + specular



$$emissive = K_e$$

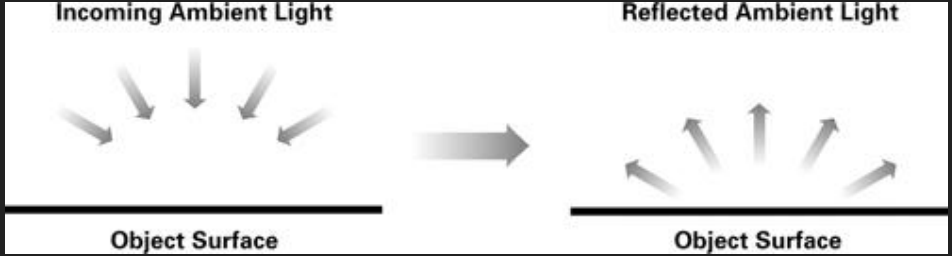
where:

- K_e is the material's emissive color.



Basic Lighting Model

surfaceColor = emissive + ambient + diffuse + specular



$$ambient = K_a \times globalAmbient$$

where:

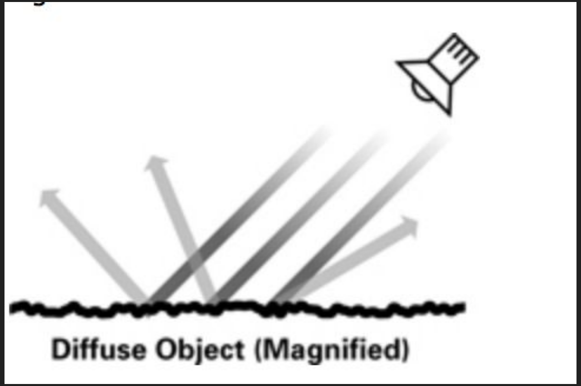
- K_a is the material's ambient reflectance and
- *globalAmbient* is the color of the incoming ambient light.

Ambient - Natural Light inside a scene. For example sun

UNITY_LIGHTMODEL_AMBIENT

Basic Lighting Model

surfaceColor = emissive + ambient + diffuse + specular



$$diffuse = K_d \times lightColor \times \max(N \cdot L, 0)$$

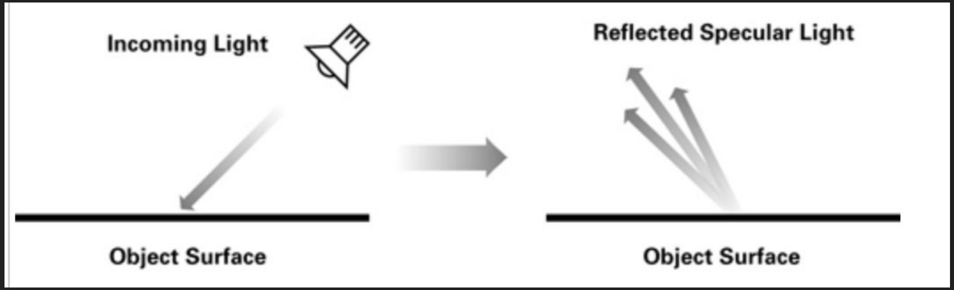
where:

- K_d is the material's diffuse color,
- *lightColor* is the color of the incoming diffuse light,
- N is the normalized surface normal,
- L is the normalized vector toward the light source, and
- P is the point being shaded.

Diffuse - Reflects light equally in all directions. Only depend on the normal and light vector

Basic Lighting Model

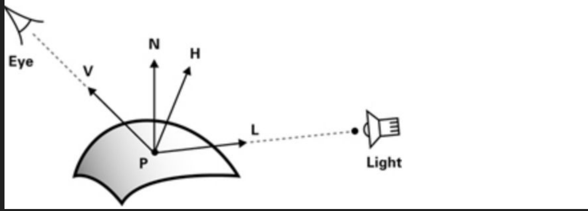
surfaceColor = emissive + ambient + diffuse + specular



$$specular = K_S \times lightColor \times facing \times (\max(N \cdot H, 0))^{shininess}$$

where:

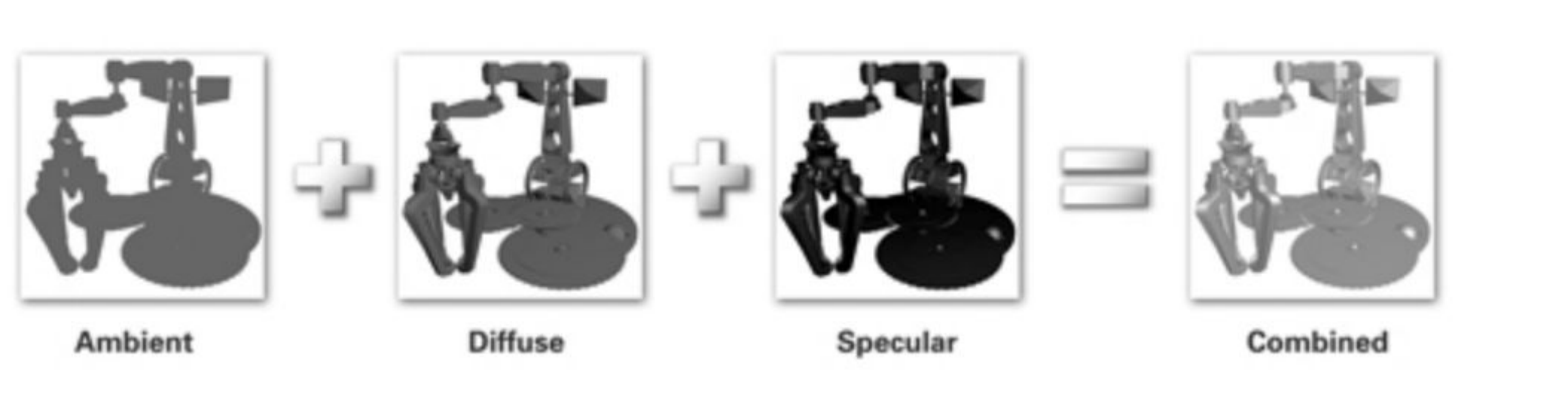
- K_S is the material's specular color,
- *lightColor* is the color of the incoming specular light,
- N is the normalized surface normal,
- V is the normalized vector toward the viewpoint,
- L is the normalized vector toward the light source,
- H is the normalized vector that is halfway between V and L,
- P is the point being shaded, and
- *facing* is 1 if $N \cdot L$ is greater than 0, and 0 otherwise.



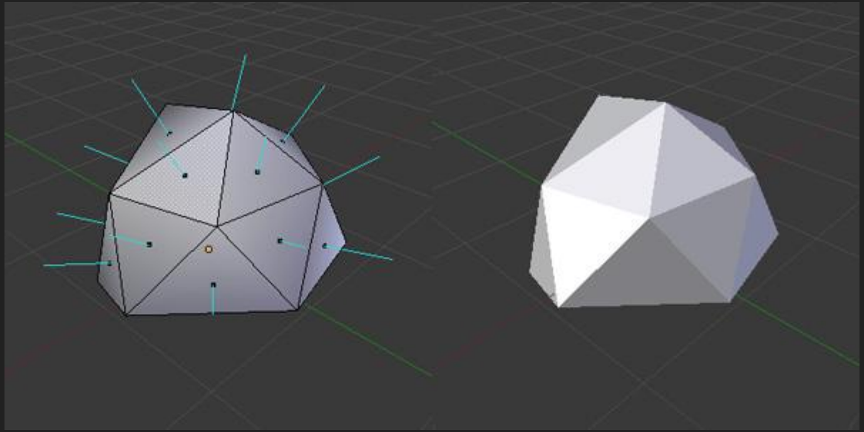
Specular - Reflects light around the mirror direction. Depends on the viewer. H vector is used for better specular approximation

Basic Lighting Model

surfaceColor = emissive + ambient + diffuse + specular

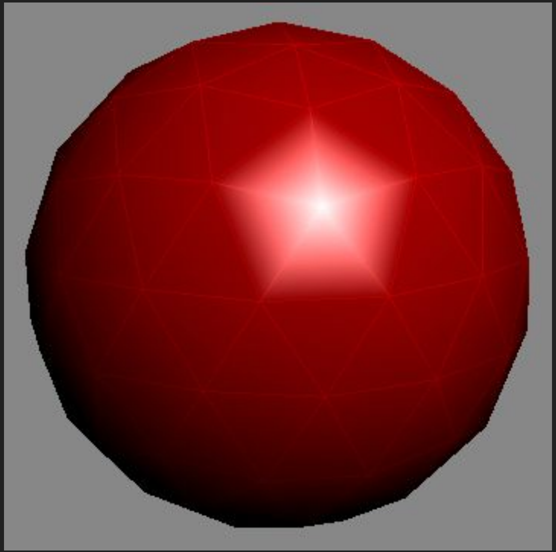


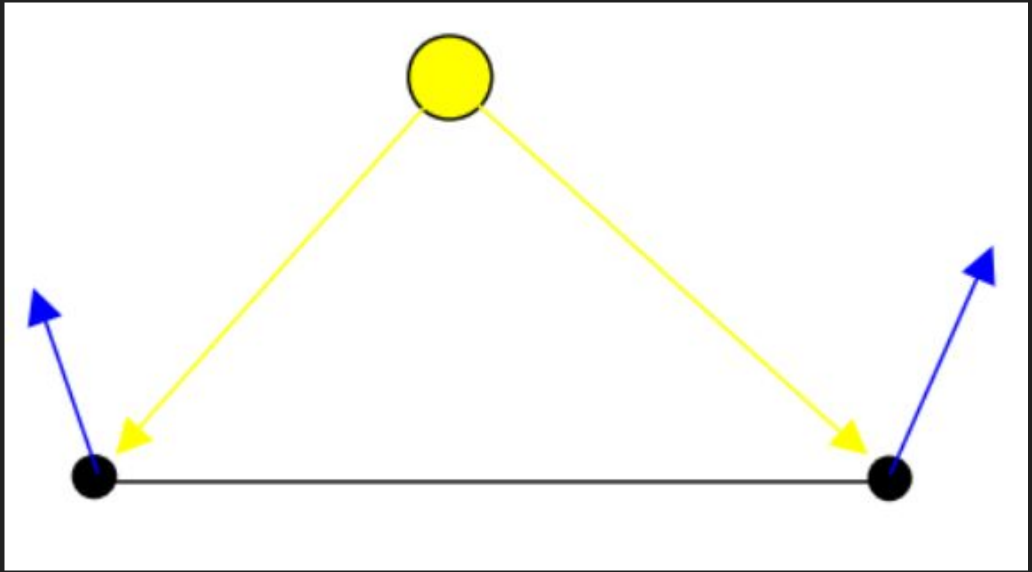
Flat Shading



Shading one color per triangle in a mesh

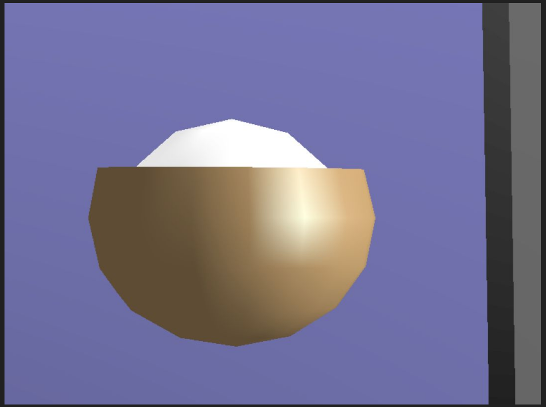
Gouraud Shading



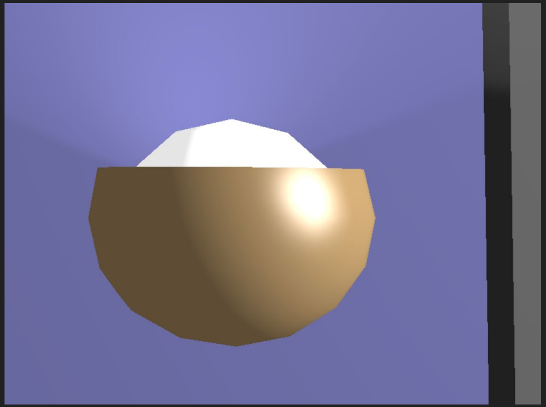


Light Calculation is done in the vertex shader for each vertex and interpolated via rasterizer. Cheaper to compute but poor specular highlights

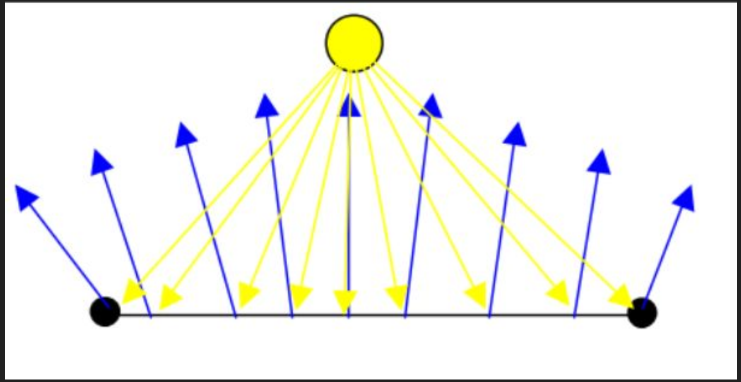
Phong Shading



Gouraud



Phong



Light Calculation is done in the fragment shader for each pixel using interpolated normals

Ambient Component

```
float 3 Ka = float3(1, 1, 1);
float3 globalAmbient = float3(0.2, 0.2, 0.2);

float3 ambientComponent = Ka * globalAmbient;
```

$$ambient = K_a \times globalAmbient$$

where:

- K_a is the material's ambient reflectance and
- *globalAmbient* is the color of the incoming ambient light.

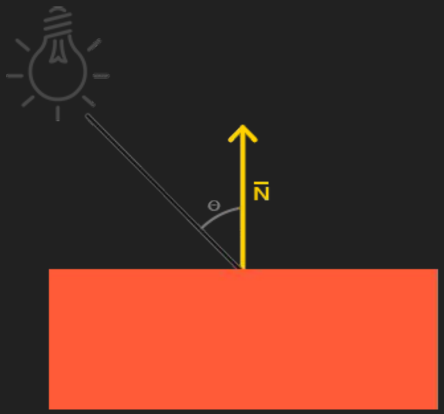
Diffuse Component

```
float3 P = i.vertInWorldCoords.xyz;

float3 N = normalize(i.normal);

float3 L = normalize(_WorldSpaceLightPos0.xyz - P);

float3 Kd = _Color.rgb;

float3 lightColor = _LightColor0.rgb;

float3 diffuseComponent = Kd * lightColor * max(dot(N,L), 0);
```

$$\textit{diffuse} = K_d \times \textit{lightColor} \times \max(N \cdot L, 0)$$

where:

- K_d is the material's diffuse color,
- *lightColor* is the color of the incoming diffuse light,
- N is the normalized surface normal,
- L is the normalized vector toward the light source, and
- P is the point being shaded.



Specular Component

```
float3 Ks = _SpecularColor.rgb;

float3 V = normalize(_WorldSpaceCameraPos - P);

float3 H = normalize(L + V);

float3 specularComponent = Ks * lightColor * pow(max(dot(N, H), 0), _Shininess);
```