

# CSE 167

# Discussion #4

**Fiat Lux (Let there be light)**

# Materials/Lights

## Materials

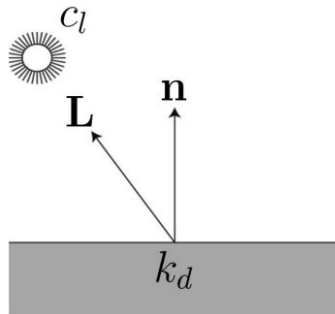
- Ambient
  - $k_a$
- Diffuse
  - $k_d$
- Specular
  - $k_s$
- Shininess
  - $\alpha$

## Lights

- Ambient
  - $L_a$
- Diffuse
  - $L_d$
- Specular
  - $L_s$

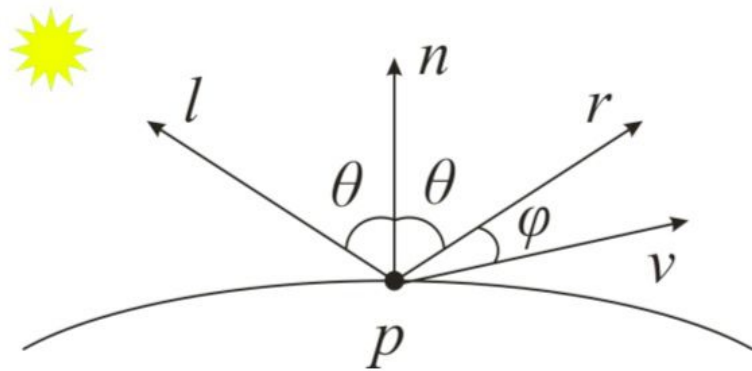
# Light Calculations

- $a * b$ 
  - Component-wise multiplication
    - $\langle a.x, a.y, a.z \rangle * \langle b.x, b.y, b.z \rangle = \langle a.x*b.x, a.y*b.y, a.z*b.z \rangle$
- $n \cdot l$ 
  - Dot product between surface normal and light vector
  - Light vector points from the vertex toward the light source



# Light Calculations

- $r \cdot v$ 
  - Dot product between reflection vector and view vector
  - Reflection vector can be calculated using the GLSL **reflect()** function
  - View vector is camera's **e** minus vertex position



# Directional Light

- Direction
- Ambient
- Diffuse
- Specular

*directional\_lights-1*

$$\sum_{n=0} [k_a L_a + k_d L_d * \max(0, (\vec{n} \cdot \vec{l})) + k_s L_s * \max(0, (\vec{r} \cdot \vec{v})^\alpha)]$$

# Point Light

- Position
- Ambient
- Diffuse
- Specular

$$\sum_{n=0}^{point\_lights-1} [k_a L_a + \frac{1}{cd^2} (k_d L_d * \max(0, (\vec{n} \cdot \vec{l}))) + k_s L_s * \max(0, (\vec{r} \cdot \vec{v})^\alpha)]$$

# Spotlight

- Position
- Spot direction
- Spot cutoff
- Spot exponent
- Ambient
- Diffuse
- Specular

$$\sum_{n=0}^{spotlights-1} \left[ k_a L_a + \frac{(\overrightarrow{spot\_dir} \cdot \overrightarrow{-l})^{spot\_exp}}{cd^2} (k_d L_d * \max(0, (\overrightarrow{n} \cdot \overrightarrow{l}))) + k_s L_s * \max(0, (\overrightarrow{r} \cdot \overrightarrow{v})^\alpha) \right]$$

# OpenGL Pipeline, again

- Transferring data from CPU to GPU is slow
- Do it once and don't have the overhead
  - `glGetUniformLocation`
  - `glUniform*`
- Once in the GPU, send data from shader to shader without revisiting CPU
  - Vertex shader's `out` → Fragment shader's `in`
  - (Some interpolation will happen)



# OpenGL Pipeline, again

CPU application

```
draw(shaderProgram)
```

```
varID = glGetUniformLocation  
(shaderProgram, "var");  
(where is this variable?)
```

```
glUniform1i  
(varID, 5);  
(set this variable to 5)
```

Vertex Shader

```
uniform int var;  
out int fragVar;
```

Send interpolated out vars  
into fragment shader's in vars

Fragment Shader

```
in int fragVar;  
layout(location = 0)  
out vec4 color;
```

Send color  
to be displayed

...

GPU shader program

# Directional Light

## DirectionalLight.cpp

```
void DirectionalLight::bind(GLuint shaderProgram)
{
    GLuint lightAttrID = glGetUniformLocation(shaderProgram,
        "dirLight.on");
    glUniform1i(lightAttrID, on);
    lightAttrID = glGetUniformLocation(shaderProgram,
        "dirLight.dir");
    glUniform3fv(lightAttrID, 1, &direction[0]);
}
```

## shader.vert

```
...
uniform struct DirLight {
    bool on;
    vec3 dir;
} dirLight;

out DirLight fragDirLight;
...
void main()
{
    fragDirLight = dirLight;
    ...
}
```

# Directional Light

## shader.vert

```
...
uniform struct DirLight {
    bool on;
    vec3 dir;
} dirLight;

out DirLight fragDirLight;
...
void main()
{
    fragDirLight = dirLight;
    ...
}
```

## shader.frag

```
...
in struct DirLight {
    bool on;
    vec3 dir;
} fragDirLight;
...
void main()
{
    if (fragDirLight.on)
    {
        //Do directional light calculation
    }
    ...
}
```