Ambient and Diffuse Light

Lecture 16

Robb T. Koether

Hampden-Sydney College

Wed, Oct 2, 2019

Outline

1. Lighting
2. Ambient Light
3. Diffuse Light
4. Programming the Shaders
   - Ambient Lighting Model
   - Diffuse Lighting Model

Outline

Lighting

- The lighting model includes three “kinds” of light.

Lighting

- The lighting model includes three “kinds” of light.
- Ambient light
  - Source is from all directions.
  - Independent of the surface’s orientation.
  - Reflected equally in all directions.

Lighting

- The lighting model includes three “kinds” of light.
- Ambient light
  - Source is from all directions.
  - Independent of the surface’s orientation.
  - Reflected equally in all directions.
- Diffuse light
  - Source is from a specific direction.
  - Dependent on the surface’s orientation.
  - Reflected equally in all directions.

Lighting

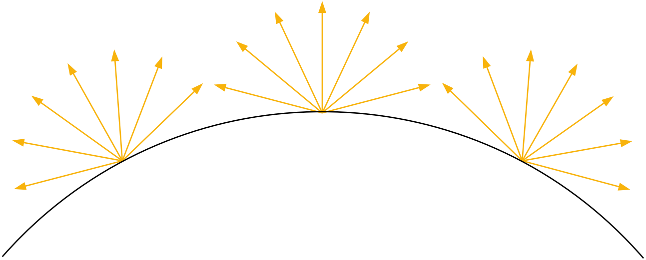
- The lighting model includes three “kinds” of light.
- Ambient light
  - Source is from all directions.
  - Independent of the surface’s orientation.
  - Reflected equally in all directions.
- Diffuse light
  - Source is from a specific direction.
  - Dependent on the surface’s orientation.
  - Reflected equally in all directions.
- Specular light
  - Source is from a specific direction
  - Dependent on the surface’s orientation.
  - Reflected more in some directions than in others.

Outline

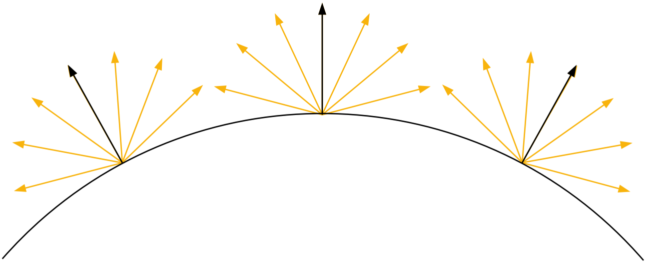
Ambient Light

Definition (Ambient Light)

Ambient light has no specific source. It illuminates all surfaces equally. The intensity of the reflected light is independent of the angle of incidence and the angle of reflection.



Ambient Light



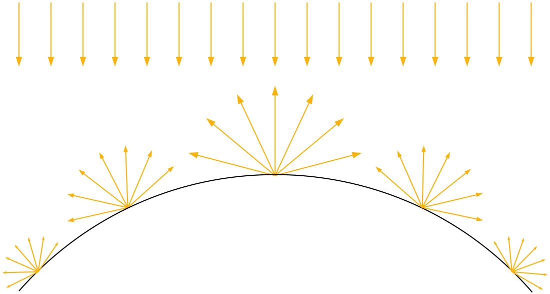
- Regardless of the orientation of the surface, the strength of the relection is the same.

Outline

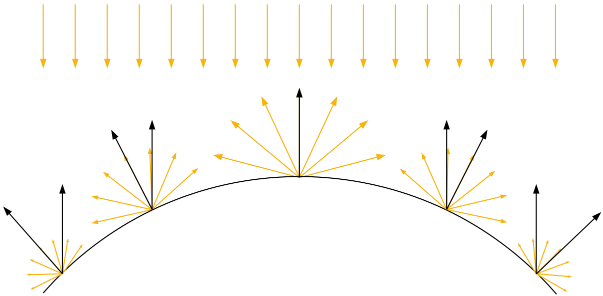
Diffuse Light

Definition (Diffuse Light)

Diffuse light is reflected equally in all directions, but it depends the direction of the light source. The intensity of the reflection depends on the angle of incidence, but does not depend on the angle of reflection.



Diffuse Light



- The angle of incidence varies over a curved surface.
- The more the light direction deviates from the normal, the weaker the reflection.

Outline

Outline

Ambient Lighting Model

- Let `color` be an RGB vector that represents the surface's inherent color.
- Let `ambient` be an RGB vector that represents the intensity of the ambient light.
- The ambient reflection is computed as

```
color*ambient
```

Outline

Diffuse Lighting Model

- Let
  - `diffuse` be an RGB vector that represents the intensity of the diffuse light.
  - `light_dir` be a unit vector indicating the direction of the light source.
  - `norm` be a unit vector normal to the surface.
- The diffuse reflection is computed as

```
color*diffuse*dot(light_dir, norm)
```

Diffuse Lighting Model

- Let
  - `diffuse` be an RGB vector that represents the intensity of the diffuse light.
  - `light_dir` be a unit vector indicating the direction of the light source.
  - `norm` be a unit vector normal to the surface.
- The diffuse reflection is computed as

```
color*diffuse*dot(light_dir, norm)
```

- If `dot(light_dir, norm)` < 0, then the diffuse contribution is 0.

Diffuse Lighting Model

- Let
  - `diffuse` be an RGB vector that represents the intensity of the diffuse light.
  - `light_dir` be a unit vector indicating the direction of the light source.
  - `norm` be a unit vector normal to the surface.
- The diffuse reflection is computed as

```
color*diffuse*dot(light_dir, norm)
```

- If `dot(light_dir, norm)` < 0, then the diffuse contribution is 0.
- Why?

Outline

Coordinate Systems

- Computing ambient shading is simple, but we must be very careful when computing diffuse shading.

Coordinate Systems

- Computing ambient shading is simple, but we must be very careful when computing diffuse shading.
- When computing the dot product $\mathbf{L} \cdot \mathbf{N}$ of the light vector $\mathbf{L}$ and the normal $\mathbf{N}$, we must be sure that they are in the same coordinate system.

Coordinate Systems

- Computing ambient shading is simple, but we must be very careful when computing diffuse shading.
- When computing the dot product $\mathbf{L} \cdot \mathbf{N}$ of the light vector $\mathbf{L}$ and the normal $\mathbf{N}$, we must be sure that they are in the same coordinate system.
- The light vector is initially in world coordinate system.

Coordinate Systems

- Computing ambient shading is simple, but we must be very careful when computing diffuse shading.
- When computing the dot product $\mathbf{L} \cdot \mathbf{N}$ of the light vector $\mathbf{L}$ and the normal $\mathbf{N}$, we must be sure that they are in the same coordinate system.
- The light vector is initially in world coordinate system.
- The normal vector is initially in model coordinates, so it must be transformed to world coordinates.

Coordinate Systems

- To make the vectors compatible, we must transform **N** into world coordinates, using the model matrix.
- However, vectors do not transform in the same way as vertices.

Coordinate Systems

- Let $\mathbf{n}$ be a vector and $\mathbf{v} = B - A$ be a vector from a point B on a surface to a point A on the surface.
- Let $\mathbf{M_1}$ be the model matrix and let $\mathbf{M_2}$ be the transformation matrix for the normal vector.
- In model coordinates, we have $\mathbf{n}^{\mathsf{T}}\mathbf{v} = 0$, because $\mathbf{n} \perp \mathbf{v}$.
- Therefore, we need to have $(\mathbf{M}_2\mathbf{n})^{\mathsf{T}}\,\mathbf{M}_1\mathbf{v} = 0$ as well.

Coordinate Systems

- We have

$$\begin{aligned}(\mathbf{M}_2\mathbf{n})^T\mathbf{M}_1\mathbf{v} &= \left(\mathbf{n}^T\mathbf{M}_2^T\right)\mathbf{M}_1\mathbf{v}\\ &= \mathbf{n}^T\left(\mathbf{M}_2^T\mathbf{M}_1\right)\mathbf{v}\end{aligned}$$

- If $\mathbf{M}_2^T\mathbf{M}_1$ is the identity matrix, then

$$\begin{aligned}\mathbf{n}^T\left(\mathbf{M}_2^T\mathbf{M}_1\right)\mathbf{v} &= \mathbf{n}^T\mathbf{I}\mathbf{v}\\ &= \mathbf{n}^T\mathbf{v}\\ &= 0.\end{aligned}$$

- Therefore, $\mathbf{M}_2\mathbf{n} \perp \mathbf{M}_1\mathbf{v}$.

- Therefore, $\mathbf{M}_2\mathbf{n} \perp \mathbf{M}_1\mathbf{v}$ provided that

$$\mathbf{M}_2 = \left(\mathbf{M}_1^T\right)^{-1}.$$

Coordinate Systems

- It so happens that for any translation matrix **T** and for any rotation matrix **R**,

$$\left(\mathbf{T}^{\mathsf{T}}\right)^{-1} = \mathbf{T},$$
$$\left(\mathbf{R}^{\mathsf{T}}\right)^{-1} = \mathbf{R}.$$

- However, for scaling matrices **S**, in general

$$\left(\mathbf{S}^{\mathsf{T}}\right)^{-1} \neq \mathbf{S}.$$

The `ModelStack` Class

The `ModelStack` Class

```
void setModelLoc(GLuint m_loc);
void setNormalLoc(GLuint n_loc);
```

- The function `setModelLoc()` will store the uniform location of the model matrix (returned by `glGetUniformLocation()`) in the `ModelStack` class.
- In the `setNormalLoc()` will store the uniform location of the normal matrix (also returned by `glGetUniformLocation()`) in the `ModelStack` class.

The `ModelStack` Class

The `ModelStack` Class

```
GLfloat* invtrans(mat4 m);
void toShader();
```

- The function `invtrans()` (private) will take a 4×4 matrix `m` and return a pointer to 3×3 matrix that is the inverse transpose of the upper-left 3×3 submatrix of `m`.
- The function `toShader()` will transfer the model matrix to the shaders. If `setNormalLoc()` was called, then it will also transfer the inverse transpose of the normal matrix to the shaders.

The `ModelStack` Function

```
model_stack.toShader();
```

- The above function call is typical of what we should use when using the model stack or programming the lighting model.

Outline

Assignment

Assignment

- Read pp. 359 - 368, Classic Lighting Model.