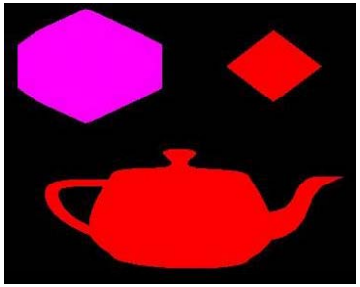
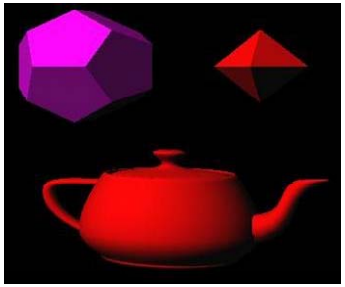
Lighting

321190
2014년 봄학기
5/9/2014
박경신

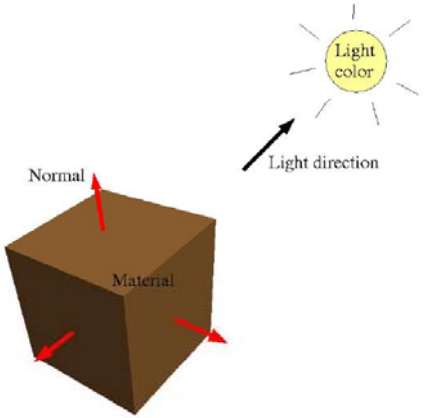
OpenGL Lighting

- OpenGL의 조명에는 3가지 요소가 필요
  - 광원 (Lights)
  - 재질 (Materials)
  - 면의 법선벡터 (Normals)



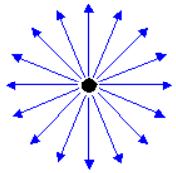


OpenGL Lighting



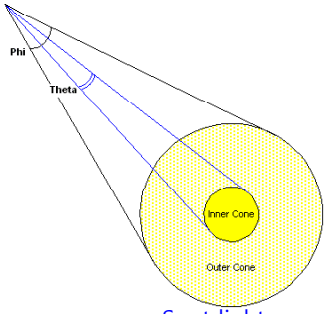
OpenGL Lighting

- OpenGL에서 제공하는 광원
  - 환경광/주변광 (ambient lights)
  - 점광원 (point lights)
  - 방향성광원/평행광원/원거리광원 (directional lights)
  - 점적광원 (spot lights)



Point light

Directional light



Spot light

OpenGL Light Sources

- 광원은 위치(position)와 색(ambient/diffuse/specular color)을 갖는다.
- 광원의 강도 (intensity)는 색의 강도에 의해 결정된다.

```
struct lightSource {
  vec4 position;
  vec4 diffuse;
  vec4 specular;
  float constantAttenuation, linearAttenuation, quadraticAttenuation;
  float spotCutoff, spotExponent;
  vec3 spotDirection;
};
```

OpenGL Light Source Position

- 방향성광원 infinite light (혹은 directional light)과 점광원 local light (혹은 positional light)
  - Infinite light source의 모든 빛은 같은 방향을 갖는다.
  - Local light source는 공간의 특정지점에서 오는 빛이다.
  - 광원의 위치의 4번째 값이 0이면 infinite light (directional light)이고, 1이면 local light (point light)이다.
- 예제:

```
if (light0.position.w == 0.0)  { // directional light
    attenuation = 1.0; // no attenuation
    lightDirection = normalize(vec3(light0.position));
}
else  { // point or spot light (or other kind of light)
    ...
}
```

OpenGL Light Source Position

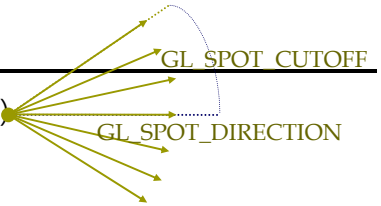
- 광원의 위치는 현재 변환 (transformation)에 영향을 받는다.
- Light은 세계 좌표계 (world coordinates system)에서 정의될 수 있도록 카메라 변환 후에 지정하는 것이 좋다.
- Light은 또한 객체와 같이 변환 행렬에 의해 움직일 수 있다.

OpenGL Light Sources

- 거리에 따른 빛의 세기 감쇠 (attenuation)
  - Physical attenuation: $I(P) = \frac{I(P_0)}{\|P - P_0\|^2}$
  - OpenGL attenuation:
    - Default a = 1, b = 0, c = 0 $I(P) = \frac{I(P_0)}{a + bd + cd^2}$

```
vec3 vertexToLightSource = vec3(light0.position - worldPosition);
float distance = length(vertexToLightSource);
lightDirection = normalize(vertexToLightSource);
attenuation = 1.0 / (light0.constantAttenuation
                + light0.linearAttenuation * distance
                + light0.quadraticAttenuation * distance * distance);
```

OpenGL Light Sources



- 점적/집중 광원 (spot light source)
  - 방향지정 벡터 (spot light direction)
  - 절단 각도 (spot light cutoff)
  - 높은 계수 (spot light exponent) 를 사용하면 보다 많은 spot light 생김

```
// light attenuation에 이어서..
if (light0.spotCutoff <= 90.0) { // spotlight
float clampedCosine
      = max(0.0, dot(lightDirection, normalize(light0.spotDirection)));
if (clampedCosine < cos(radians(light0.spotCutoff))) // outside of spotlight
   cone {
      attenuation = 0.0;
} else  {
       attenuation = attenuation * pow(clampedCosine,light0.spotExponent);
}
```

OpenGL Multiple Lights

- OpenGL에서는 GL_LIGHT0, GL_LIGHT1, .. 등으로 지정함으로써 여러 개의 광원을 동시에 활성화할 수 있다.
- 최대 8개까지의 광원을 사용할 수 있다.
- 너무 많은 light을 사용하면 계산량이 늘어나게 되고 따라서 렌더링 속도를 늦출 수 있다.

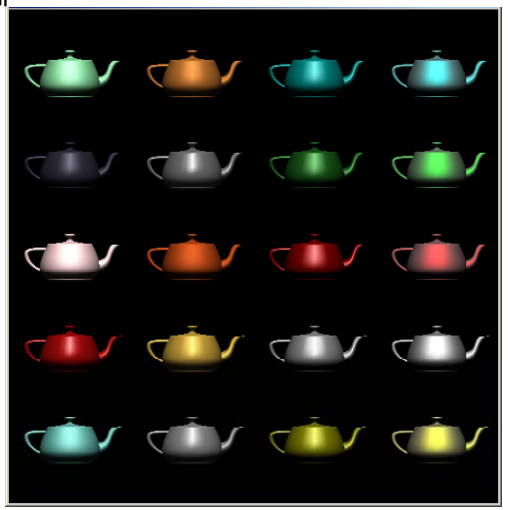
OpenGL Materials

- 재질 속성은 표면이 빛을 어떻게 반사하는지를 나타내준다.
- 재질은 기본적으로 표면의 색을 말한다.
- 조명이 활성화되면, glColor는 무시되고 재질(material)이 대신 사용된다.

```
struct material
{
   vec4 ambient;
   vec4 diffuse;
   vec4 specular;
   float shininess;
};
```

Example

- Teapot의 재질예제



OpenGL Light Color

- OpenGL에서 광원은 색(color)과 강도(intensity)를 가지고 있으며, 기본값은 백색이다.
- 우리가 보는 객체의 색은 빛의 색 (light color)과 재질의 색 (material color) 둘 간의 합성으로 결정된다.
  - 빛의 색은 얼마나 RGB 빛이 객체에서 빛나는지를 말한다.
  - 재질의 색은 얼마나 RGB 빛이 객체에서 반사되는지를 말한다.
  - White light * red material = red
  - Red light * white material = red
  - Red light * blue material = black
  - Yellow light * cyan material = green

OpenGL Ambient/Diffuse/Specular Light

- OpenGL 조명 모델에서는 3가지 형태의 반사광을 고려함.
- Ambient light (환경광): 다른 일반 표면에서 반사되어 나오는 빛. 장면을 전반적으로 밝게 함.
- Diffuse light (난반사광): 특정 방향으로 진행하다가 표면에 닿으면 모든 방향으로 동일하게 반사됨. 관찰자의 위치와 무관함. 가장 일반적인 형태임.
- Specular light (정반사광): 특정 방향으로 진행하다가 표면에 닿으면 한 방향으로 강하게 반사됨. 반짝이는 표면을 모델링할 때 이용됨.

OpenGL Ambient Light

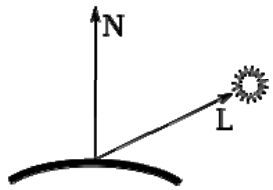
- 환경광은 간접적으로 들어오는 빛을 시뮬레이션함.
- 표면의 방향과는 상관없이 사방에 일정한 밝기의 빛이 고르게 퍼져있다고 가정함.
- 환경광 효과는 광원의 환경광색과 재질의 환경광색으로 지정됨.

```
vec3 ambientLighting
   = vec3(scene_ambient) * vec3(mymaterial.ambient);
```

OpenGL Diffuse Light

- 특정 방향으로 진행하다가 표면에 닿으면 모든 방향으로 동일하게 반사되는 특징을 가짐.
- 관찰자의 위치와 무관하며, 가장 일반적인 형태임.

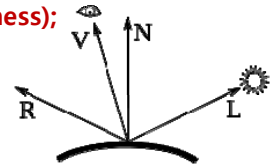
```
vec3 diffuseReflection
   = attenuation * vec3(light0.diffuse) * vec3(mymaterial.diffuse)
   * max(0.0, dot(normalDirection, lightDirection));
```



OpenGL Specular Light

- 정반사광은 표면에 반짝이는 highlights를 생성함.
- 정반사광의 3가지 속성:
  - 재질의 정반사 색
  - 광원의 정반사 색
  - 재질의 반짝임 (shininess) – Shiness값이 높으면 작은 면적의 highlight이 생성

specularReflection
= attenuation * vec3(light0.specular) * vec3(mymaterial.specular)
* **pow(max(0.0, dot(reflect(-lightDirection, normalDirection), viewDirection)), mymaterial.shininess);**

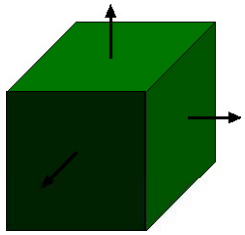


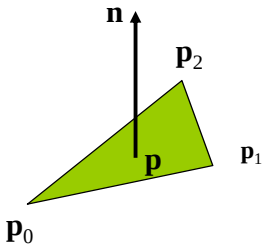
Surface Orientation

- Lighting에서 빛이 표면에서 얼마나 반사되는지 계산하기 위해서 OpenGL은 표면이 어느 방향을 향하고 있는지 알아야 한다.
- 표면의 방향 (surface orientation)과 빛의 방향은 빛의 난반사광 (diffuse light)의 밝기를 결정짓는다.
- 표면의 방향 (surface orientation), 빛의 방향 (light direction), 관찰자를 향하는 방향벡터 (direction to the viewer)는 정반사광 (specular light)의 밝기를 결정짓는다.

Surface Normal

- 표면의 방향 (orientation)은 표면에 수직을 이루는 법선벡터에 의해 정의된다.
- 면의 법선 벡터는 (벡터의 길이가 1인) 단위벡터 (unit vector) 이어야 한다.





$\mathbf{n} \cdot (\mathbf{p} - \mathbf{p}_0) = 0$

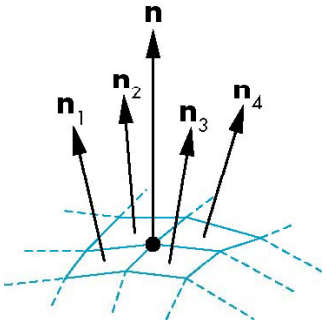
$\mathbf{n} = (\mathbf{p}2 - \mathbf{p}0) \times (\mathbf{p}1 - \mathbf{p}0)$

normalize $\mathbf{n} \leftarrow \mathbf{n} / |\mathbf{n}|$

Vertex Normal

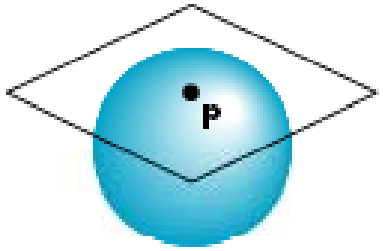
- 정점의 법선벡터
  - 그 정점을 공유하는 근접한 면의 법선벡터의 평균치를 계산하여 사용함.

$$n = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$$



Normal to Sphere

- Implicit function of Sphere:
  - $f(x,y.z)=0$
- Unit Sphere:
  - $f(\mathbf{p})=\mathbf{p}\cdot\mathbf{p}-1=0$
- 구의 법선벡터
  - $n = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]^T = \mathbf{p}$



OpenGL Normal

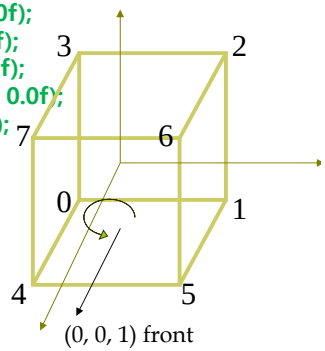
- OpenGL에서 면의 법선벡터는 glNormal 함수로 지정
- OpenGL의 조명 계산에서는 각 정점의 법선벡터를 사용함
- 예제:

```
// Vertice Positions
squareVertices.push_back(glm::vec3(-1.5f, -1.5f, 0.0f));
squareVertices.push_back(glm::vec3( 1.5f, -1.5f, 0.0f));
squareVertices.push_back(glm::vec3( 1.5f,  1.5f, 0.0f));
squareVertices.push_back(glm::vec3(-1.5f,  1.5f, 0.0f));
// Vertices Normals
squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
squareNormals.push_back(glm::vec3(0.0f, 0.0f, 1.0f));
```

OpenGL Normal

- Shaded Cube를 그린다.

```
glm::vec3 frontNormal = glm::vec3(0.0f, 0.0f, 1.0f);
glm::vec3 backNormal = glm::vec3(0.0f, 0.0f, -1.0f);
glm::vec3 leftNormal = glm::vec3(-1.0f, 0.0f, 0.0f);
glm::vec3 rightNormal = glm::vec3(1.0f, 0.0f, 0.0f);
glm::vec3 bottomNormal = glm::vec3(0.0f, -1.0f, 0.0f);
glm::vec3 topNormal = glm::vec3(0.0f, 1.0f, 0.0f);
```



OpenGL Normal

```
// Front face
vbo.addData(&glm::vec3(-size, -size,  size), sizeof(glm::vec3));
vbo.addData(&frontNormal[0], sizeof(glm::vec3)); // vertex normal
vbo.addData(&glm::vec3( size, -size,  size), sizeof(glm::vec3));
vbo.addData(&frontNormal[0], sizeof(glm::vec3)); // vertex normal
vbo.addData(&glm::vec3( size,  size,  size), sizeof(glm::vec3));
vbo.addData(&frontNormal[0], sizeof(glm::vec3)); // vertex normal
vbo.addData(&glm::vec3(-size, -size,  size), sizeof(glm::vec3));
vbo.addData(&frontNormal[0], sizeof(glm::vec3)); // vertex normal
vbo.addData(&glm::vec3( size,  size,  size), sizeof(glm::vec3));
vbo.addData(&frontNormal[0], sizeof(glm::vec3)); // vertex normal
vbo.addData(&glm::vec3(-size,  size,  size), sizeof(glm::vec3));
vbo.addData(&frontNormal[0], sizeof(glm::vec3)); // vertex normal
```