Lighting (2)

GPU Programming

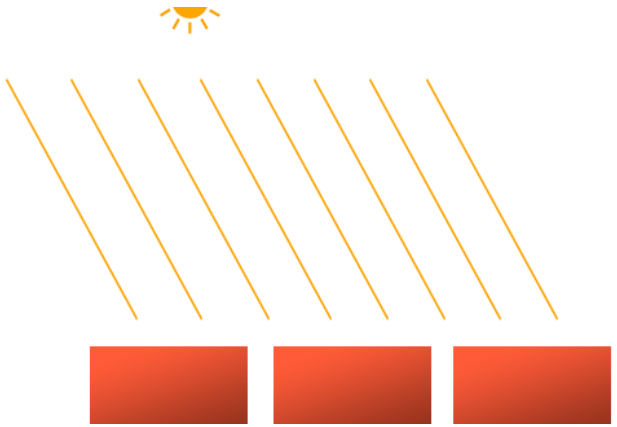
2022학년도
2학기

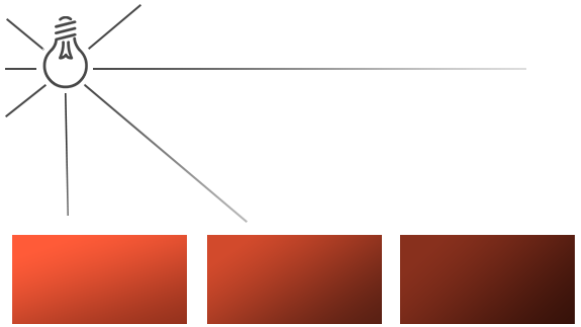
ⓒ nvidia

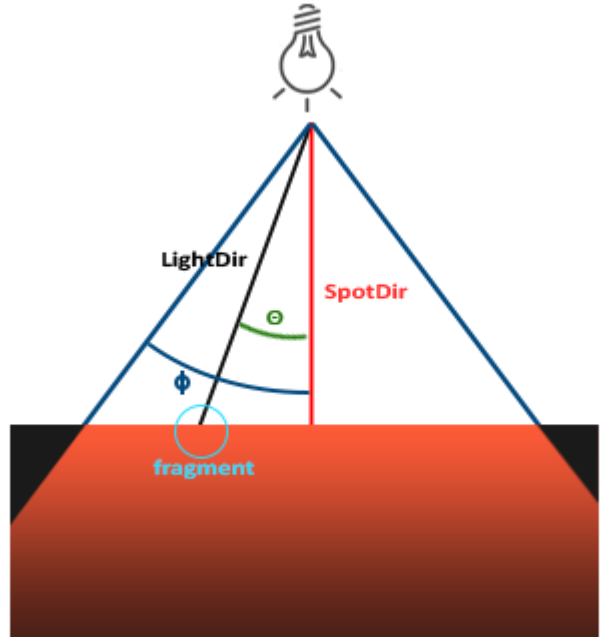
Light Casters

Light Casters

- Light caster
  - 물체에 빛을 발하는(cast) 광원(light source)
  - 다양한 종류의 light caster는 고품질의 환경을 만들어 내는 데 도움
- 본 과목에서는 세 가지 light caster를 소개
  - Directional light (방향이 있는 빛)
  - Point lights (점광원)
  - Spotlight (스포트라이트)

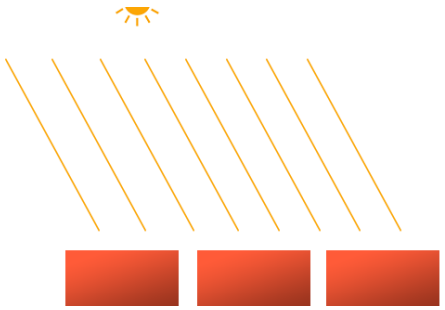






Directional Light

- 아주 먼 광원으로부터 오는 거의 평행한 광선(ray)을 표현 (예: 햇빛)
  - 물체의 위치, viewer의 위치, 광원의 위치에 상관 없이 ray의 방향이 동일하다고 가정
  - 모든 물체에 비슷하게 lighting 계산이 적용



- Directional light 구현 방법
  - 빛에 대한 위치 벡터 대신에 빛이 들어오는 방향 벡터(direction)를 정의
  - lightDir은 이를 반대 방향으로 (fragment→광원) 뒤집고 유닛 벡터로 정규화

```
struct Light {
    // vec3 position; // no longer necessary when using directional lights.
    vec3 direction;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
[...]
void main()
{
  vec3 lightDir = normalize(-light.direction);
  [...]
}
```

FS

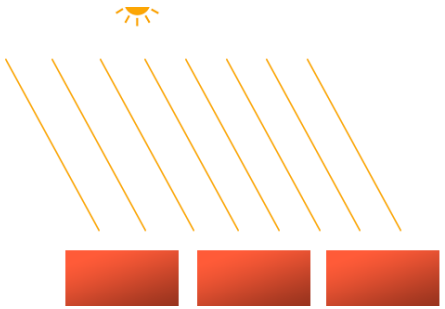
Directional Light

- Directional light 구현 방법 (cont.)
  - 빛의 방향을 CPU 코드에서 vec3 형태로 정해 셰이더로 넘겨 줌

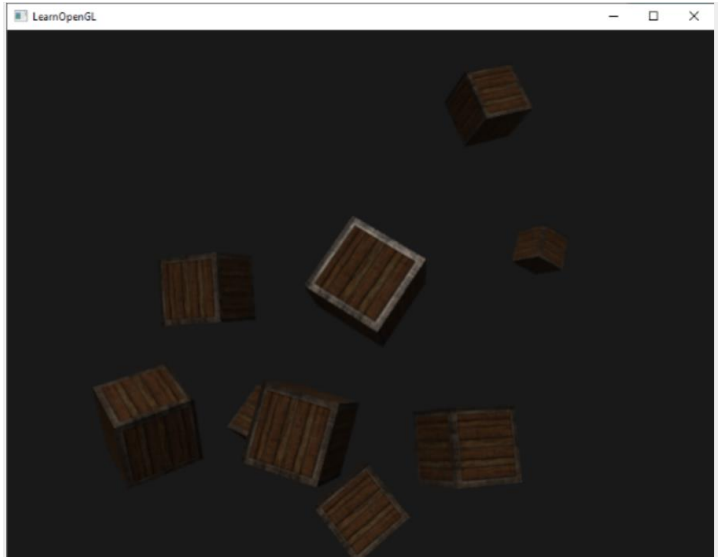
```
lightingShader.setVec3("light.direction", -0.2f, -1.0f, -0.3f);
```
CPU

  - 만약 point light와 directional light를 함께 사용할 경우에는, vec4 형태로 방향벡터를 만들어 lighting 계산시 구분 가능

```
if(lightVector.w == 0.0) // note: be careful for floating point errors
    // do directional light calculations
else if(lightVector.w == 1.0)
    // do light calculations using the light's position (as in previous chapters)
```
FS

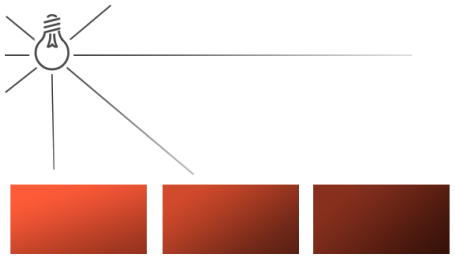


- 실행 결과
  - 지난 시간에 사용한 10개 컨테이너를 재사용



Point Lights

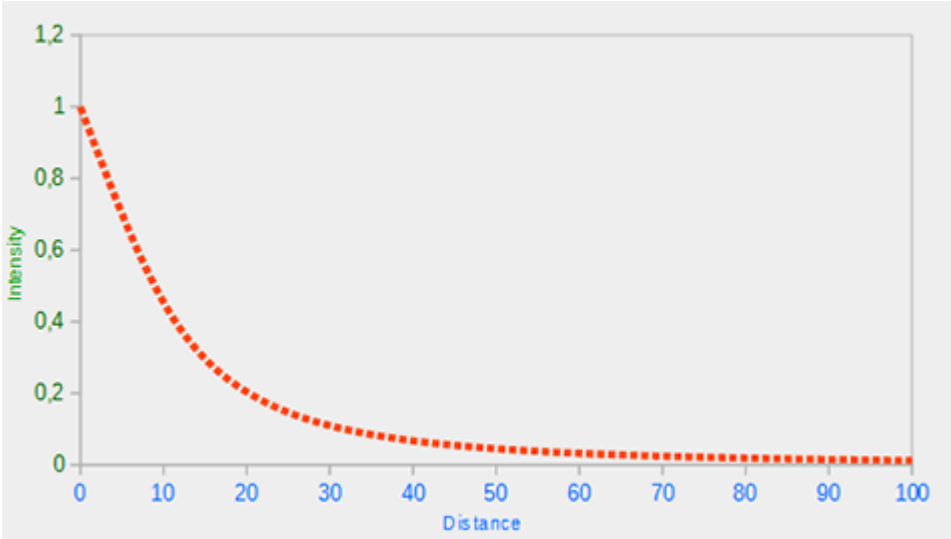
- World space 상의 어딘가에서 모든 방향으로 빛을 밝히는 광원 (예: 전구, 횃불 등)
  - Directional ray와 달리, 광원에서 멀어질수록 빛의 세기가 줄어드는 감쇠의 필요성 존재



- Attenuation(감쇠)
  - 광원 바로 옆에 있는 물체는 아주 밝지만, 멀리 떨어진 물체는 어둡게 처리
  - 빛의 세기는 광원 근처에서는 급격하게 줄지만, 멀어질수록 그 감쇠 속도는 느려짐

- 감쇠 공식
  - d는 거리, K_c, K_l, K_q는 각각 상수, 1차, 2차항

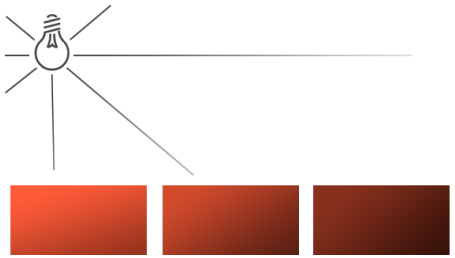
$$F_{att} = \frac{1.0}{K_c + K_l * d + K_q * d^2}$$



Point Lights

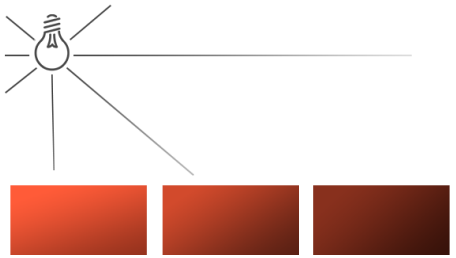
- 감쇠 공식의 항별로 적절한 값 선택
  - Point Light Attenuation | Ogre Wiki (ogre3d.org)
  - 특정한 반지름(거리)에 대해 현실적으로 광원을 시뮬레이션
  - 교재의 환경에서는 32~100 정도가 적당

Distance	Constant	Linear	Quadratic
7	1.0	0.7	1.8
13	1.0	0.35	0.44
20	1.0	0.22	0.20
32	1.0	0.14	0.07
50	1.0	0.09	0.032
65	1.0	0.07	0.017
100	1.0	0.045	0.0075
160	1.0	0.027	0.0028
200	1.0	0.022	0.0019
325	1.0	0.014	0.0007
600	1.0	0.007	0.0002
3250	1.0	0.0014	0.000007



Point Lights

- 감쇠의 구현
  - 아래 구현에서는 빛이 닿는 거리를 50이라고 가정하고, ambient에도 똑같이 감쇠 적용
  - Ambient lighting에 대해서는 거리에 비례하지 않도록 감쇠 식을 따로 적용할 수도 있음 (단, K_c 를 광원의 개수에 비례하도록 설정)



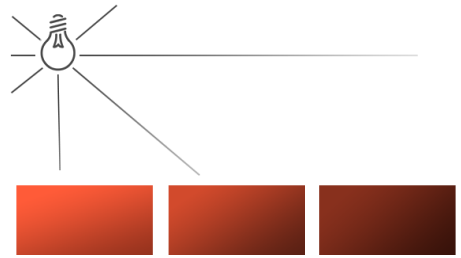
FS
```
struct Light {
    vec3 position;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;

    float constant;
    float linear;
    float quadratic;
};

float distance    = length(light.position - FragPos);
float attenuation = 1.0 / (light.constant + light.linear * distance +
                    light.quadratic * (distance * distance));
ambient  *= attenuation;
diffuse  *= attenuation;
specular *= attenuation;
```

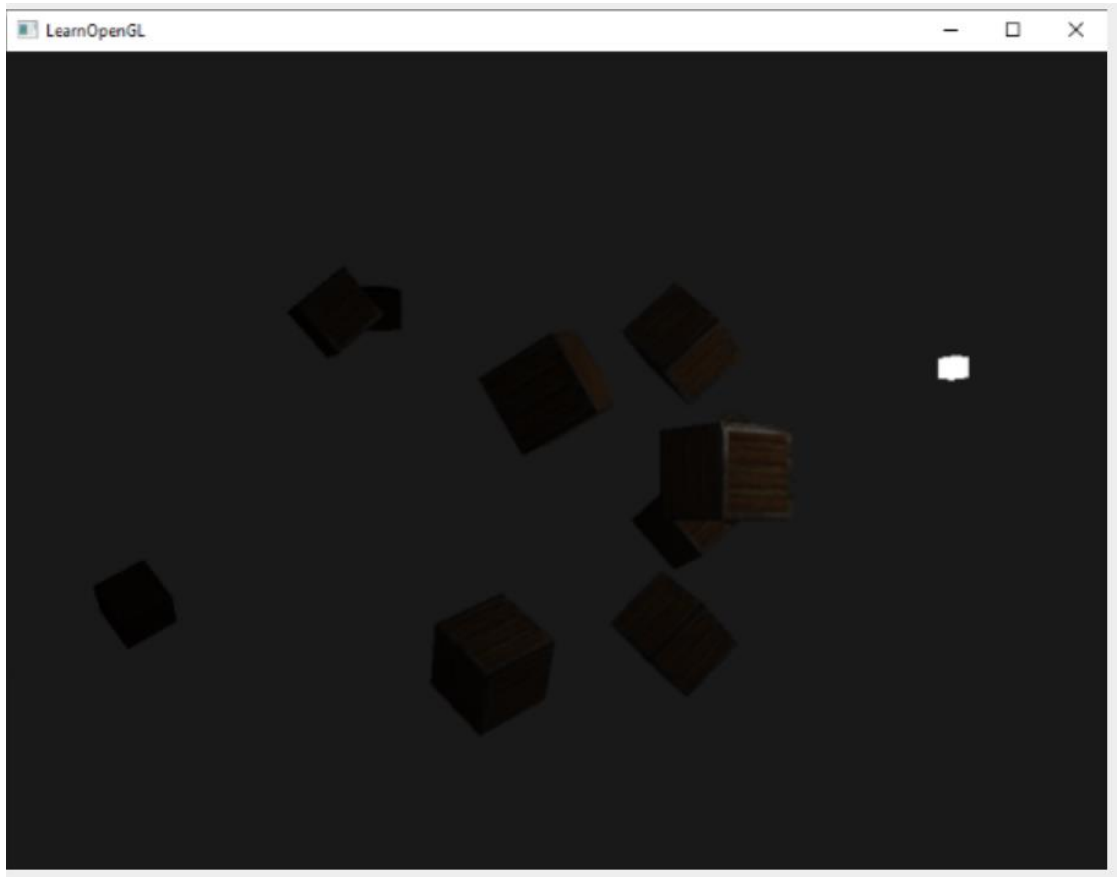
CPU
```
lightingShader.setFloat("light.constant",  1.0f);
lightingShader.setFloat("light.linear",    0.09f);
lightingShader.setFloat("light.quadratic", 0.032f);
```

Distance	Constant	Linear	Quadratic
7	1.0	0.7	1.8
13	1.0	0.35	0.44
20	1.0	0.22	0.20
32	1.0	0.14	0.07
50	1.0	0.09	0.032
65	1.0	0.07	0.017
100	1.0	0.045	0.0075
160	1.0	0.027	0.0028
200	1.0	0.022	0.0019
325	1.0	0.014	0.0007
600	1.0	0.007	0.0002
3250	1.0	0.0014	0.000007

Point Lights

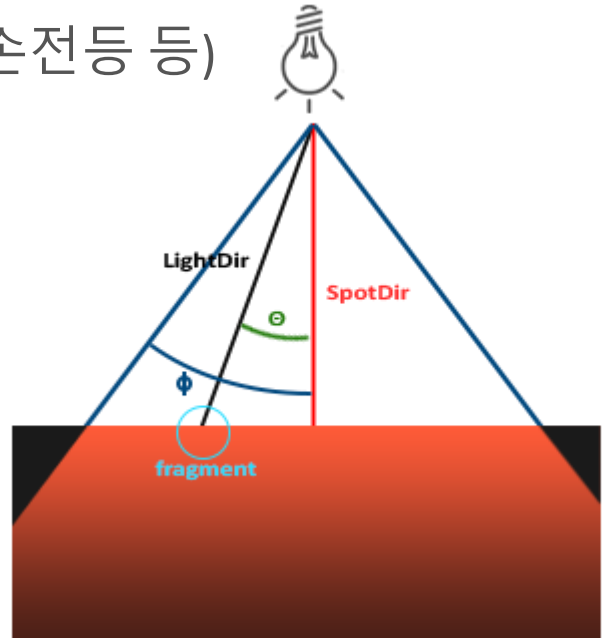
- 감쇠 구현 결과
  - 전체 구현 코드: Code Viewer. Source code: src/2.lighting/5.2.light_casters_point/light_casters_point.cpp (learnopengl.com)





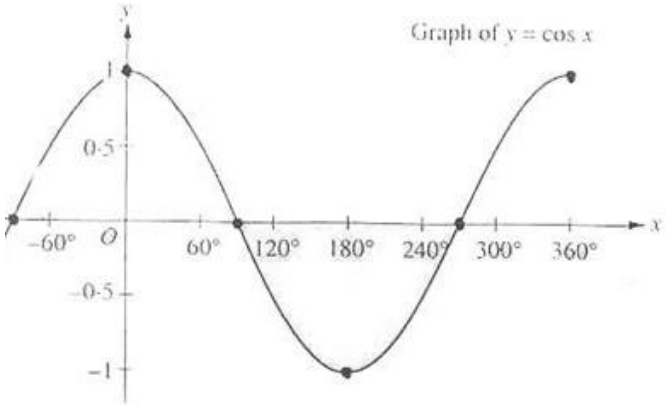
Spotlight

- World space 상의 어딘가에서 특정 방향으로 빛을 밝히는 광원 (예: 가로등, 손전등 등)
  - Spotlight 방향의 특정 반지름 내부의 물체만 밝아짐
  - 각 fragment가 spotlight의 원뿔 내에 존재하는지 판단하여 lighting 계산



- Spotlight 계산에 필요한 요소
  - LightDir: fragment→광원 방향의 벡터
  - SpotDir: spotlight가 겨누고 있는 방향
  - Phi(φ): spotlight의 반지름을 지정하는 cutoff 각.
    이 각 내부에 있어야만 spotlight의 빛을 받음
  - Theta(θ): LightDir 벡터와 SpotDir 벡터 사이의 각도 (θ < φ)

- Spotlight 계산
  - LightDir과 SpotDir 벡터를 내적하여 cosθ를 구한 후,
    이를 cutoff(cosφ)와 비교 (cosθ > cosφ)



Spotlight

- Flashlight 구현 예제
  - Viewer의 위치에서 보는 spotlight (위치와 방향이 viewer와 동기화)

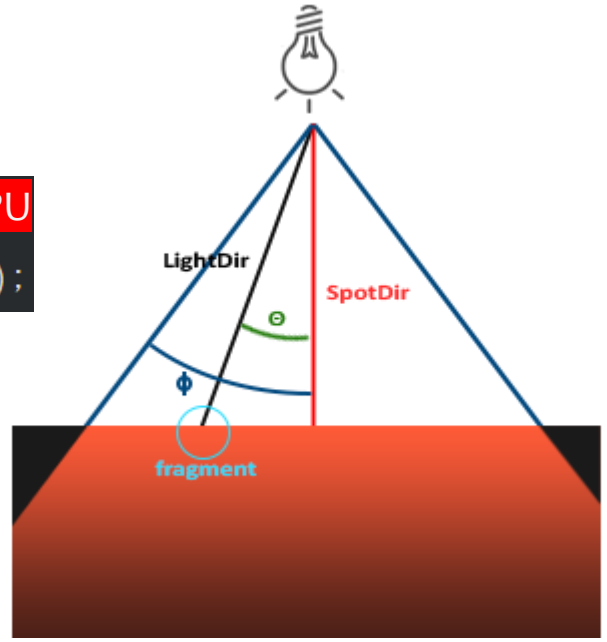
```
struct Light {
    vec3  position;
    vec3  direction;
    float cutOff;
    ...
};
```

CPU

```
lightingShader.setVec3("light.position",  camera.Position);
lightingShader.setVec3("light.direction", camera.Front);
lightingShader.setFloat("light.cutOff",   glm::cos(glm::radians(12.5f)));
```

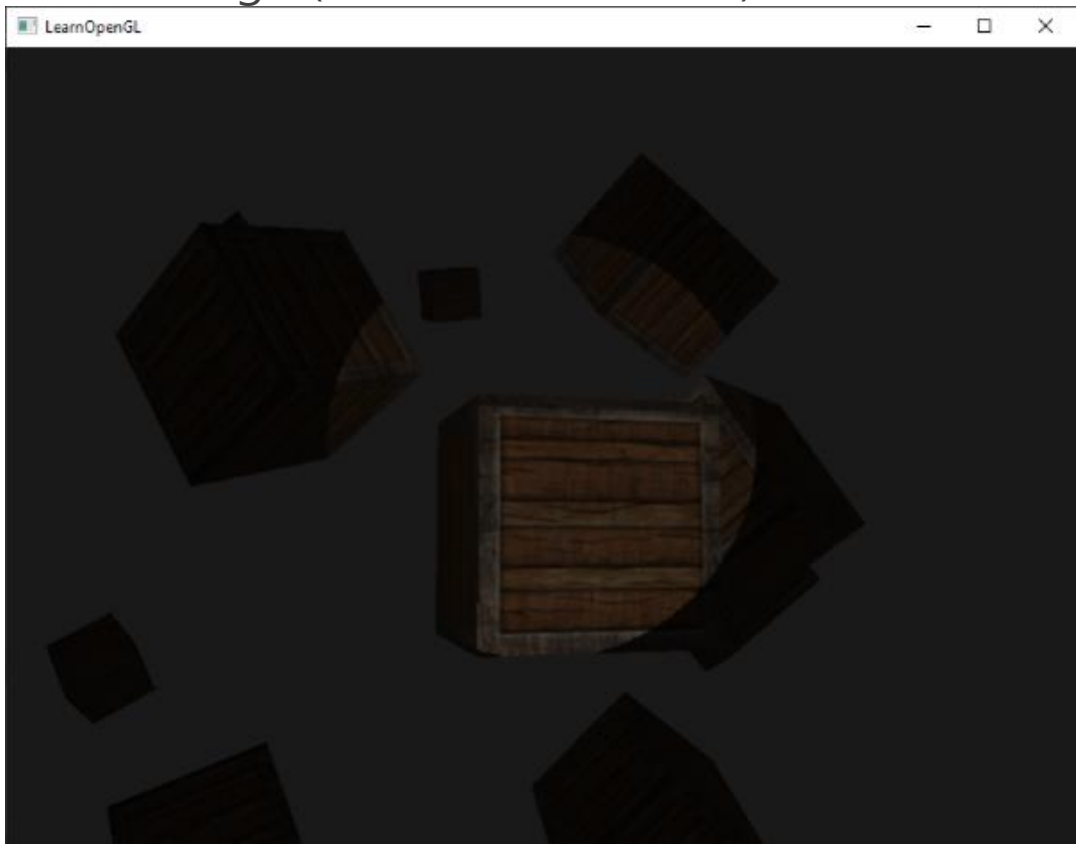
FS

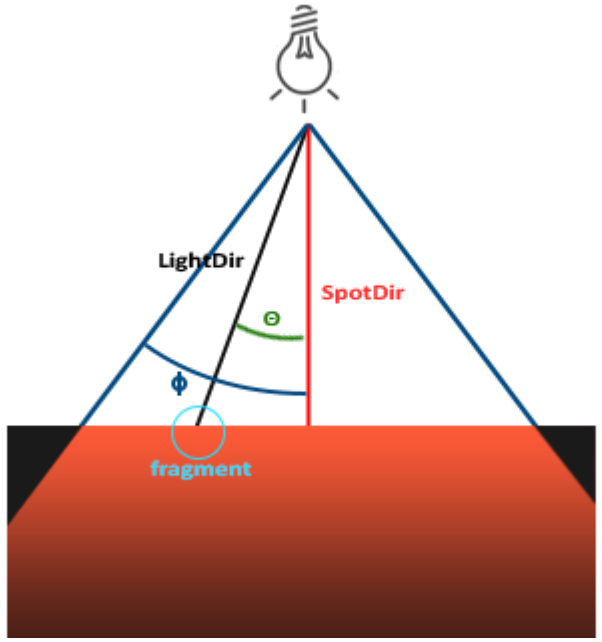
```
float theta = dot(lightDir, normalize(-light.direction));

if(theta > light.cutOff)
{
  // do lighting calculations
}
else  // else, use ambient light so scene isn't completely dark outside the spotlight.
  color = vec4(light.ambient * vec3(texture(material.diffuse, TexCoords)), 1.0);
```



Spotlight

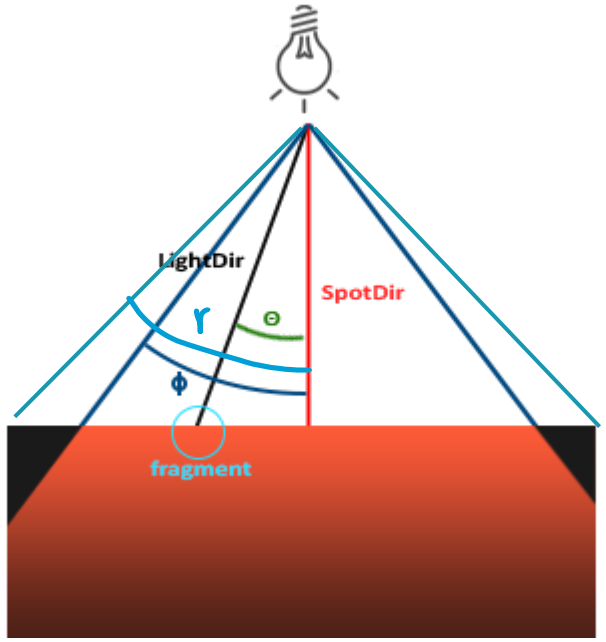
- Flashlight 구현 결과
  - 전체 코드: Code Viewer. Source code: src/2.lighting/5.3.light_casters_spot/light_casters_spot.cpp (learnopengl.com)
  - Hard edge (딱 떨어지는 외곽선) 때문에 다소 부자연스러워 보임





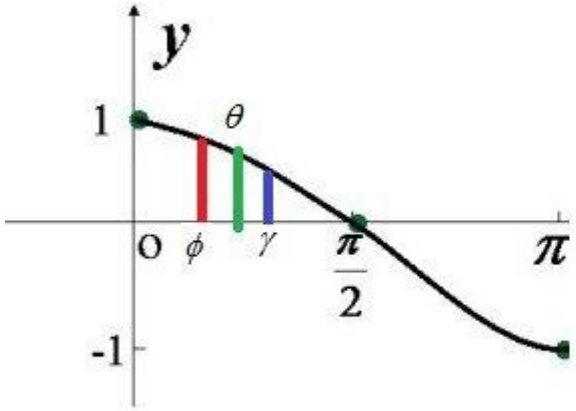
Spotlight

- Smooth/Soft edges
  - 안쪽/바깥쪽 원뿔을 가지는 spotlight 구현
  - 안쪽으로부터 바깥쪽 뿔로는 점차적으로 흐려짐



- Smooth/Soft edges 계산을 위한 공식
  - I=현재 fragment에서의 강도 [0.0, 1.0]
  - ϵ=φ−γ, 안쪽(φ)과 바깥쪽 원뿔(γ)의 cosine 값 차이

$$I = \frac{\theta - \gamma}{\epsilon}$$



$$f(\theta,\phi,\gamma) = \begin{cases} \geq 1.0, & 0 \leq \theta \leq \phi \\ (0.0,1.0), & \phi < \theta < \gamma \\ \leq 0.0, & \theta \geq \gamma \end{cases}$$

$$(0 < \phi < \gamma < \frac{\pi}{2})$$

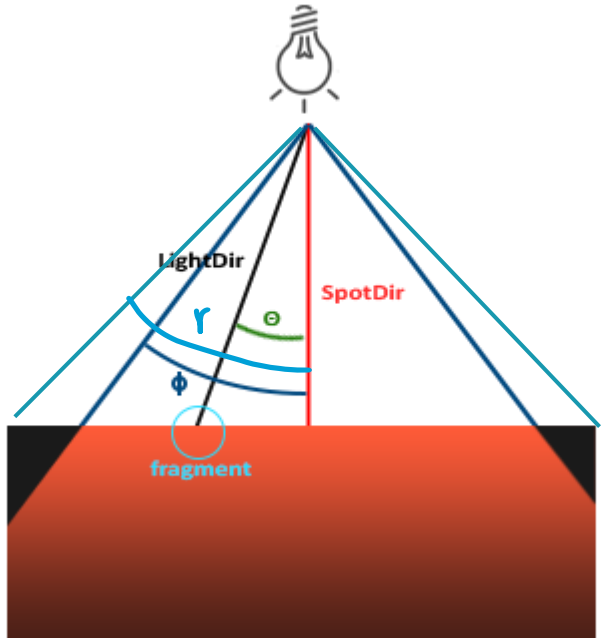
Spotlight

- Soft edge를 가지도록 flashlight 수정

FS

```
float theta     = dot(lightDir, normalize(-light.direction));
float epsilon   = light.cutOff - light.outerCutOff;
float intensity = clamp((theta - light.outerCutOff) / epsilon, 0.0, 1.0);
...
// we'll leave ambient unaffected so we always have a little light.
diffuse  *= intensity;
specular *= intensity;
```

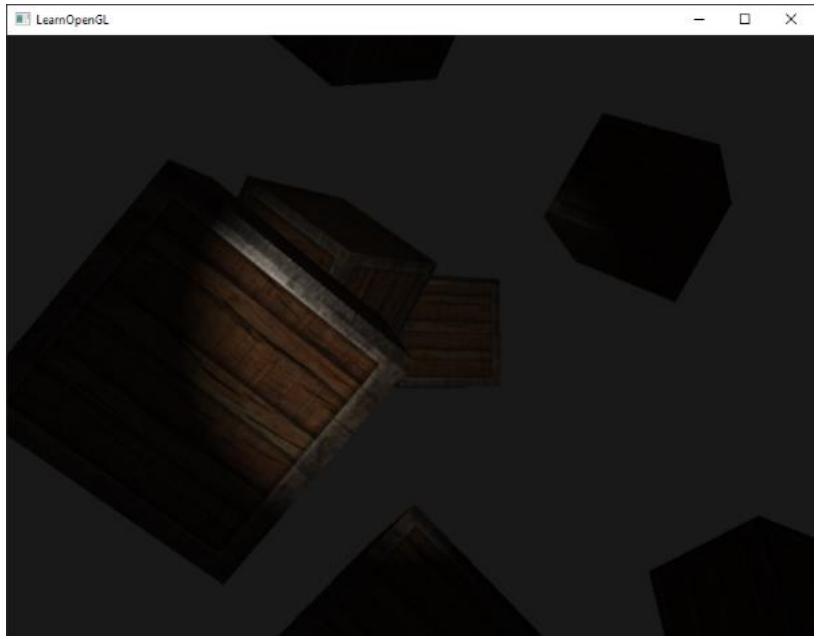
$$I = \frac{\theta - \gamma}{\epsilon}$$



CPU

```
lightingShader.setFloat("light.outerCutOff", glm::cos(glm::radians(17.5f)));
```

- 실행 결과
  - 전체 코드: Code Viewer. Source code: src/2.lighting/5.4.light_casters_spot_soft/light_casters_spot_soft.cpp (learnopengl.com)
  - 호러 게임에서 사용 가능!



ⓒ nvidia

Multiple Lights

Multiple Lights

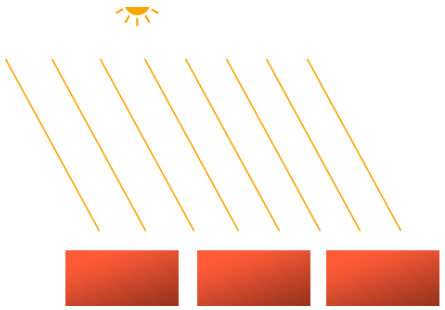
- 6개 광원 사용 예제
  - 1개의 directional light (태양과 같은 빛)
  - 4개의 point light (장면 전체에 빛 산란)
  - 1개의 flashlight

- Lighting 계산을 GLSL의 함수들로 캡슐화
  - C의 함수 사용법과 유사 (Prototype을 호출 전에 선언)
  - calDirLight() 및 calPointLight()

Directional Light

```
struct DirLight {
    vec3 direction;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
uniform DirLight dirLight;
```

FS

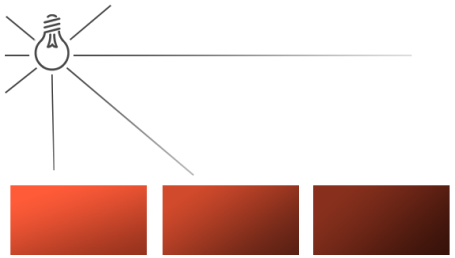


```
vec3 CalcDirLight(DirLight light, vec3 normal, vec3 viewDir)
{
    vec3 lightDir = normalize(-light.direction);
    // diffuse shading
    float diff = max(dot(normal, lightDir), 0.0);
    // specular shading
    vec3 reflectDir = reflect(-lightDir, normal);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
    // combine results
    vec3 ambient  = light.ambient  * vec3(texture(material.diffuse, TexCoords));
    vec3 diffuse  = light.diffuse  * diff * vec3(texture(material.diffuse, TexCoords));
    vec3 specular = light.specular * spec * vec3(texture(material.specular, TexCoords));
    return (ambient + diffuse + specular);
}
```

Point Light

```
struct PointLight {
    vec3 position;

    float constant;
    float linear;
    float quadratic;

    vec3 ambient;
    vec3 diffuse;
    vec3 specular;
};
#define NR_POINT_LIGHTS 4
uniform PointLight pointLights[NR_POINT_LIGHTS];
```

- NR_POINT_LIGHTS 개수만큼 PointLight 배열 생성



FS

```
vec3 CalcPointLight(PointLight light, vec3 normal, vec3 fragPos, vec3 viewDir)
{
    vec3 lightDir = normalize(light.position - fragPos);
    // diffuse shading
    float diff = max(dot(normal, lightDir), 0.0);
    // specular shading
    vec3 reflectDir = reflect(-lightDir, normal);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), material.shininess);
    // attenuation
    float distance    = length(light.position - fragPos);
    float attenuation = 1.0 / (light.constant + light.linear * distance +
                               light.quadratic * (distance * distance));
    // combine results
    vec3 ambient  = light.ambient  * vec3(texture(material.diffuse, TexCoords));
    vec3 diffuse  = light.diffuse  * diff * vec3(texture(material.diffuse, TexCoords));
    vec3 specular = light.specular * spec * vec3(texture(material.specular, TexCoords));
    ambient  *= attenuation;
    diffuse  *= attenuation;
    specular *= attenuation;
    return (ambient + diffuse + specular);
}
```

Putting It All Together

- 최종 코드
  - 최적화 여지는 남아 있음 (반사 벡터, diffuse 및 specular term, material texture sampling 등의 중복 계산)

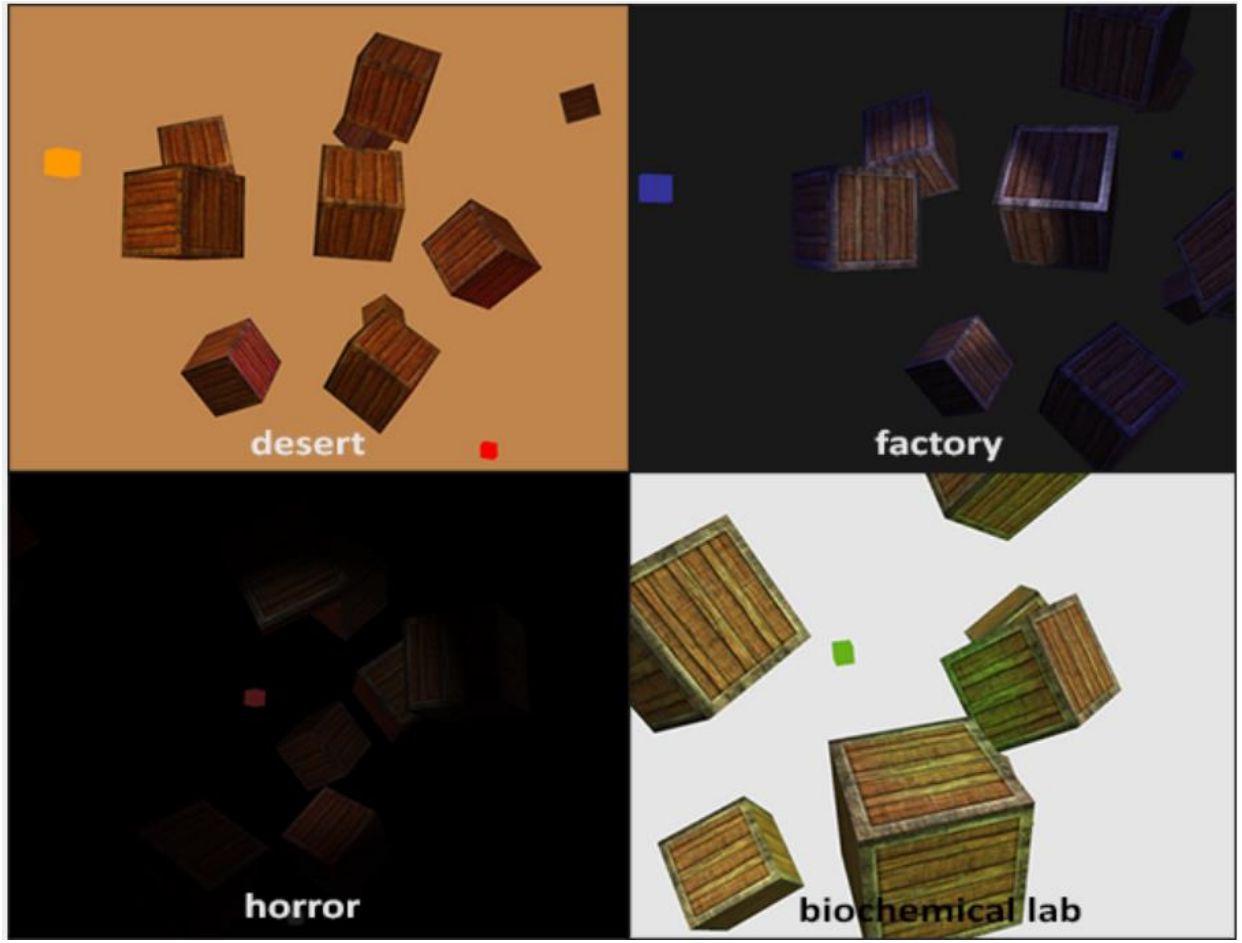
FS
```
void main()
{
    // properties
    vec3 norm = normalize(Normal);
    vec3 viewDir = normalize(viewPos - FragPos);

    // phase 1: Directional lighting
    vec3 result = CalcDirLight(dirLight, norm, viewDir);
    // phase 2: Point lights
    for(int i = 0; i < NR_POINT_LIGHTS; i++)
        result += CalcPointLight(pointLights[i], norm, FragPos, viewDir);
    // phase 3: Spot light
    //result += CalcSpotLight(spotLight, norm, FragPos, viewDir);

    FragColor = vec4(result, 1.0);
}
```

  - 각 광원별로 아래와 같이 구조체의 멤버를 하나하나 다 채워줘야 함

CPU
```
lightingShader.setFloat("pointLights[0].constant", 1.0f);
```

Putting It All Together

- 실행 결과
  - 광원과 배경색의 설정 변경만으로 아래와 같이 다양한 환경을 시뮬레이션 할 수 있음



ⓒ nvidia

마무리

마무리

- 이번 시간에는 기본 lighting의 두번째 시간으로, 아래와 같은 내용을 살펴보았습니다.
  - 여러가지 type의 light caster
  - 빛의 감쇠 (attenuation)
  - 여러 광원을 동시에 배치하여 lighting을 수행하는 방법

Directional lighting

Point lighting

Spotlight

attenuation | Learn OpenGL ES

- 이어서 model loading 파트로 넘어갑니다.