Computer Graphics HW#4

공과대학 컴퓨터공학부

2020-19422 권신영

- Specification
  - Phong shading

    Complete Phong Illumination Model을 사용하였다.

$$I_{\text{out}} = k_a I_a + \frac{I_{\text{light}}}{d^2}\left[k_d \max(N \cdot L, 0) + k_s (R \cdot V)^n\right]$$

```
for (int i = 0; i <num_lights; ++i) {

    vec3 light_dir = normalize(light_pos[i] - frag_pos);

    // ambient
    vec3 ambient = ambient_strength * light_color[i];

    // diffuse
    float diff = max(dot(norm, light_dir), 0); // 뒤에서오는거 처리
    vec3 diffuse = diff * diffuse_strength * light_color[i];

    // attenuation
    float distance = length(light_pos[i] - frag_pos);
    float attenuation = 1 / (1 + 0.05 * distance + 0.01 * distance * distance);
    // 거리 및 거리 제곱에 의한 감쇠 (더 자연스럽게)

    // specular
    vec3 specular = vec3(0);

    if (diff > 0) {
        vec3 view_dir = normalize(view_pos - frag_pos);
        vec3 reflect_dir = reflect(-light_dir, norm);
        float spec = pow(max(dot(view_dir, reflect_dir), 0), shininess);
        specular = specular_strength * spec * light_color[i];
    }

    // I(out) = ka * Ia + ...을 구현
    result += (ambient + attenuation * (diffuse + specular));
}
```

(shader.py)

Attenuation의 경우, 거리에 의한 2차 다항식을 사용하여 자연스럽게 보정하였다.

■ Parameter Selection

```
# 할 일) 클래스 별로 구분해서 적용하기
# 각 재질에 따라

if shape.type == "Box":
    shader['ambient_strength'] = 0.1
    shader['specular_strength'] = 0.1
    shader['diffuse_strength'] = 0.2
    shader['shininess'] = 12

elif shape.type == "Cornell Box":
    shader['ambient_strength'] = 0.05
    shader['specular_strength'] = 0.3
    shader['diffuse_strength'] = 0.5
    shader['shininess'] = 2

elif shape.type == "Tape": # Tape
    shader['ambient_strength'] = 0.1
    shader['specular_strength'] = 0.3
    shader['diffuse_strength'] = 0.4
    shader['shininess'] = 9

else:
    shader['ambient_strength'] = 1
    shader['specular_strength'] = 1
    shader['diffuse_strength'] = 1
    shader['shininess'] = 50
```

(render_hw4.py – def update)

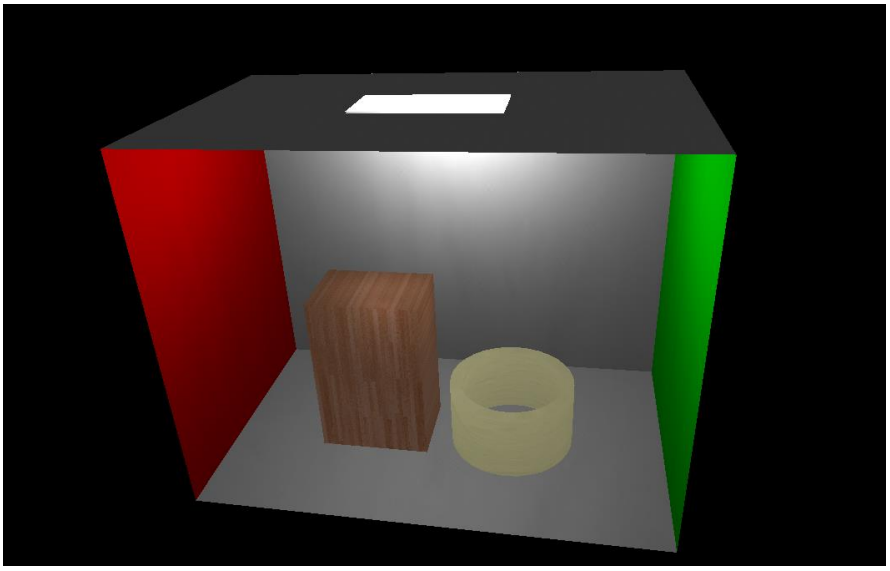
물체에 따라 다른 계수를 적용하였다. 수업 자료 RenderingBasics.pdf의 참고 자료를 사용하여 각 재질이 실제와 비슷해 보이도록 조정하였다.

■ Texture mapping

왼쪽부터 차례대로 Tape, Box, Paper texture이다.

Tape는 원통형 모양의 테이프에, Box는 큐브 형태의 박스에, Paper는 Cornell Box

의 벽면에 적용되었다.



(texture가 각 물체에 적용된 모습)

이 texture를 적용하기 위해, 각 primitive마다 (u, v) coordinate를 세팅해주었다.

```
self.text_coords = [
    0.0, 0.0,
    1.0, 0.0,
    1.0, 1.0,
    0.0, 1.0,
    0.0, 0.0,
    1.0, 0.0,
    1.0, 1.0,
    0.0, 1.0,
]
```

(primitive의 class Cube에 대한 uv coordinate)

- Lights

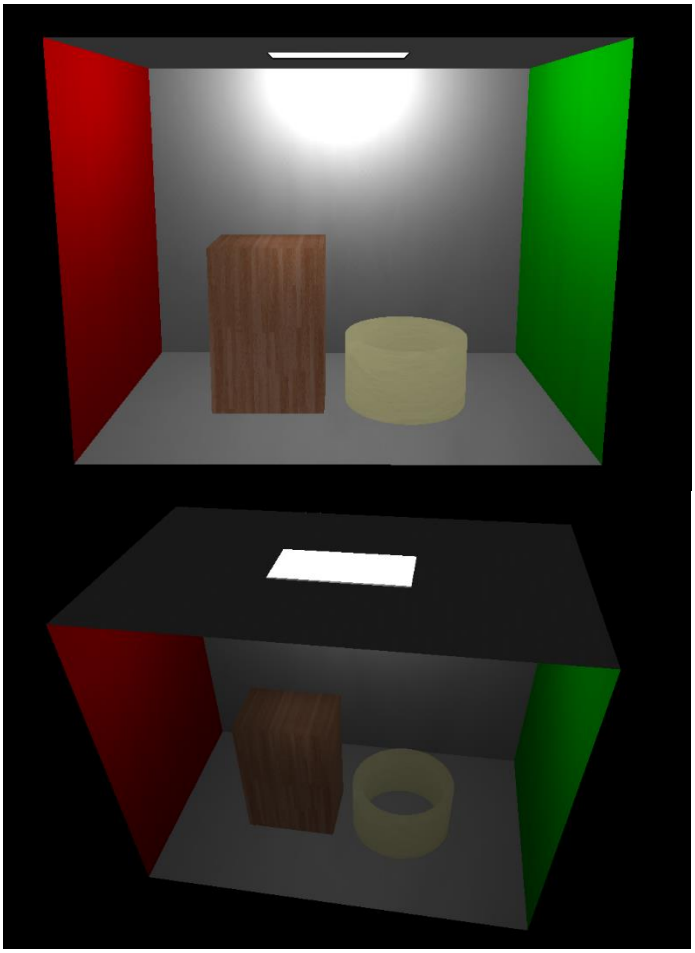
```
def update(self, dt):
    # view_projection 행렬 계산
    view_projection_matrix = self.proj_mat @ self.view_mat

    # 광원 위치와 색상
    # (x, y, z) = (9.5, 4.5, 12.5) -> 4.75, 2.25, 12.5
    light_positions = []

    for i in range(10):
        for j in range(10):
            x = -4.75 + 9.5 * i / 9
            z = -2.25 + 4.5 * j / 9

            light_positions.append(Vec3(x, 12.5, z))

    light_colors = [Vec3(0.01 * self.lights, 0.01 * self.lights, 0.01 * self.lights)] * 100
```

(render_hw4.py – def update)

Cornell box의 상단에 뚫린 구멍에 광원을 위치시켰다. 면 광원을 근사적으로 구현하기 위해, 구멍을 가로, 세로 각각 10등분하여 각 위치에 작은 점광원을 배치시켰고, 총 100개의 광원이 존재한다.

K키를 통해 광원의 빛 세기를 조절할 수 있다. 최초 상태는 가장 센 상태이다.

- Result Image



Hw2와 동일하게 TrackballViewer가 적용되어 있다. 위 사진이 최초 실행 시 모습이고, K키를 통해 광원을 5단계로 조절할 수 있다. Report에 실행 화면 동영상이 첨부되어 있다.

- 실행법
  - conda env create -f environment.yml
  - conda activate snu_graphics로 가상환경 실행
  - python main.py로 pyglet window를 실행
  - **K키를 통해 광원의 세기 조작 가능**

- Reference

https://reminder-by-kwan.tistory.com/169 - Phong shading

https://people.eecs.ku.edu/~jrmiller/Courses/672/InClass/3DLighting/MaterialProperties.html (수업 자료)

https://blogs.igalia.com/itoral/2017/07/06/working_lights_shadows_parti_phong_reflection_model/ - Phong shading