

# Homework #6: Phong Reflection

김준호

## Abstract

본 과제에서는 Phong Shading 모델을 통해 per-pixel로 물체를 렌더링하는 프로그램을 작성한다.

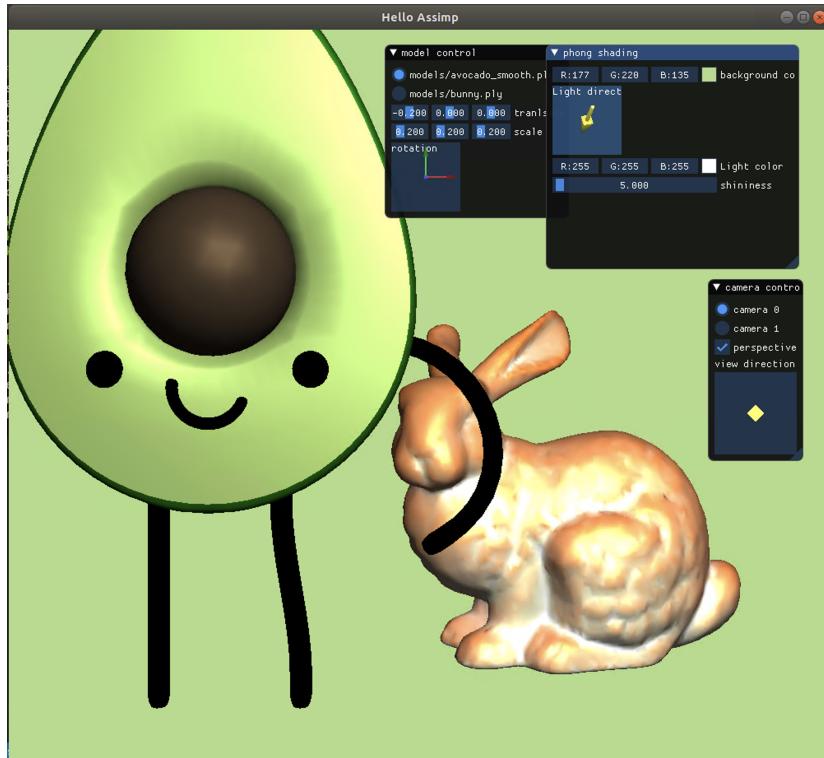


Fig. 1: 완성된 프로그램

## 1 과제 가이드

이번 과제에서는 per-vertex lighting으로 구현된 Phong shading model을 per-pixel lighting으로 구현한다. 미리 per-vertex lighting으로 구현된 vertex shader는 다음과 같다.

---

```
#version 120 // GLSL 1.20

attribute vec3 a_position; // per-vertex position (per-vertex input)
attribute vec3 a_color; // per-vertex color (per-vertex input)
attribute vec3 a_normal; // per-vertex normal (per-vertex input)
varying vec3 v_color;
uniform mat4 u_PVM;

// for phong shading
uniform vec3 u_light_position;
uniform vec3 u_light_color;
uniform float u_obj_shininess;
```

---

```

uniform vec3 u_camera_position;
uniform mat4 u_view_matrix;
uniform mat4 u_model_matrix;
uniform mat3 u_normal_matrix;

void main()
{
    gl_Position = u_PVM * vec4(a_position, 1.0f);

    // ambient 계산
    float ambientStrength = 0.2;
    vec3 ambient = ambientStrength * u_light_color;

    // world coordinate
    vec3 position_wc = (u_model_matrix * vec4(a_position, 1.0f)).xyz;
    vec3 normal_wc = normalize(u_normal_matrix * a_normal);

    // diffuse 계산
    vec3 light_dir = normalize(u_light_position - position_wc);
    float diff = max(dot(normal_wc, light_dir), 0.0);
    vec3 diffuse = diff * u_light_color;

    vec3 reflect_dir = reflect(-light_dir, normal_wc);
    vec3 view_dir = normalize(u_camera_position - position_wc);

    // specular 계산
    float rdotv = pow(max(dot(view_dir, reflect_dir), 0.0), u_obj_shininess);
    vec3 specular = rdotv * u_light_color;

    vec3 color = (ambient + diffuse + specular) * a_color;

    // fragment shader로 보내기
    v_color = color;
}

```

위 코드를 참고하여, Fig. 1과 같이 per-pixel lighting으로 구현된 Phong shading 프로그램을 완성하는 것이 이번 과제의 목표이다. 과제 진행을 위해, 아래 링크를 참고하도록 한다.

- 1) LearnOpenGL - Basic Lighting (link)
- 2) Phong Reflection (link)

### 1.1 과제 세부사항

- vertex.gls, fragment.gls 파일을 수정하여 per-pixel shading을 구현한다.
- main.cpp의 // TODO 부분을 수정하여 물체가 올바로 그려지도록 한다.

### 1.2 과제 유의사항

- Camera, Object 클래스는 이전 과제에서 본인이 작성한 코드를 사용하도록 한다.
- ImGui로 빛의 색, 방향, 배경 등을 조작하는 코드는 미리 구현되어 있다.

## 2 과제 제출방법(매우 중요!!)

- 본 과제는 개인과제이며, 각자 자신의 코드를 완성하도록 한다.
- 공지된 마감 시간까지 과제 코드를 가상대학에 업로드하도록 한다.
- 과제 코드는 Ubuntu 18.04 LTS 환경에서 make 명령으로 컴파일 가능하도록 작성한다.
- 과제 코드는 다음의 파일들을 하나의 압축파일로 묶어 tar.gz 파일 형식이나 표준 zip파일 형식으로만 제출하도록 한다. 이때, 압축파일의 이름은 반드시 'OOOOOOOO\_HW06.tar.gz (OOOOOOOO은 자신의 학번)'과 같이 자신의 학번이 드러나도록 제출한다.
  - 1) 소스코드 및 리소스 파일들
  - 2) Makefile
- 과제에 관한 질문은 오피스아워를 활용하도록 한다. 오피스아워 이외의 시간에 도움을 받으려면 교육조교(teaching assistant, TA)에게 메일로 약속시간을 정한 후, 교육조교가 있는 연구실로 방문하도록 한다.