

# Homework #6: Phong Reflection

김준호

## Abstract

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

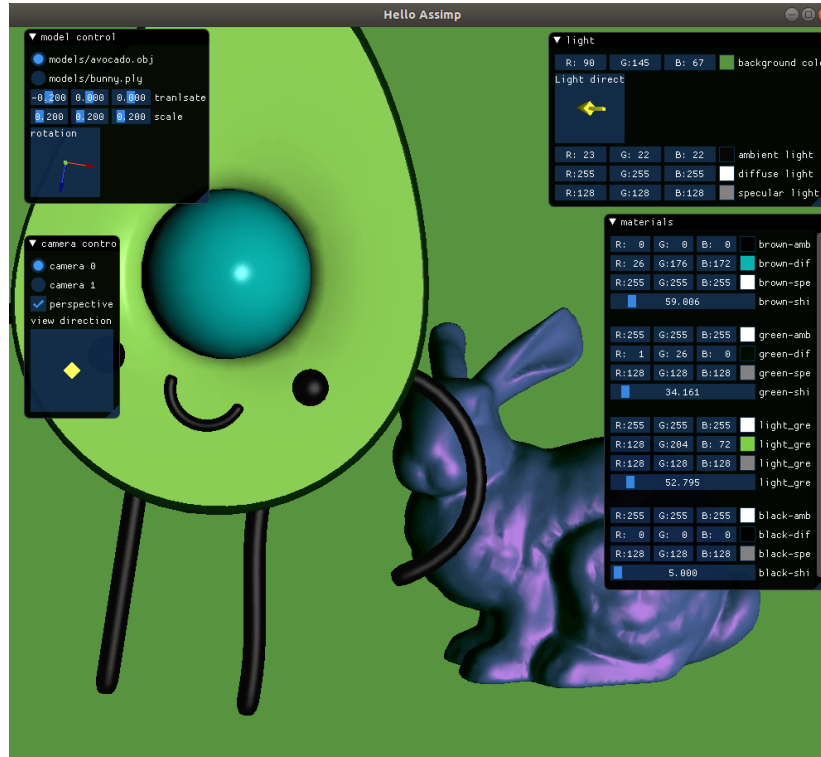


Fig. 1: 완성된 프로그램

## 1 과제 가이드

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

```
#version 120                                // GLSL 1.20

attribute vec3 a_position;                  // per-vertex position (per-vertex input)
attribute vec3 a_normal;                    // per-vertex normal (per-vertex input)

uniform mat4 u_PVM;

// for phong shading
uniform mat4 u_model_matrix;
uniform mat3 u_normal_matrix;

uniform vec3 u_light_position;
uniform vec3 u_light_ambient;
```

```

uniform vec3 u_light_diffuse;
uniform vec3 u_light_specular;

uniform vec3 u_obj_ambient;
uniform vec3 u_obj_diffuse;
uniform vec3 u_obj_specular;
uniform float u_obj_shininess;

uniform vec3 u_camera_position;
uniform mat4 u_view_matrix;

varying vec3 v_color;

vec3 directional_light()
{
    vec3 color = vec3(0.0);

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

    vec3 light_dir = normalize(u_light_position);

    // ambient
    color += (u_light_ambient * u_obj_ambient);

    // diffuse
    float ndotl = max(dot(normal_wc, light_dir), 0.0);
    color += (ndotl * u_light_diffuse * u_obj_diffuse);

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

    float rdotv = max(dot(view_dir, reflect_dir), 0.0);
    color += (pow(rdotv, u_obj_shininess) * u_light_specular * u_obj_shininess);

    return color;
}

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

```

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.glsl, fragment.glsl 파일을 수정하여 per-pixel shading을 구현한다.
- main.cpp, Mesh.cpp, Model.cpp의 *// TODO* 부분을 수정하여 물체가 올바르게 그려지도록 한다.

### 1.2 과제 유의사항

- Camera, Mesh 클래스는 이전 과제에서 본인이 작성한 코드를 사용하도록 한다.
- Mesh 클래스는 이전 과제의 Object 클래스에 대응된다. 여러 재질을 가진 하나의 물체를 구현하기 위해, 여러 개의 Mesh를 가진 Model 클래스를 완성한다. 보다 자세한 방법은 LearnOpenGL - Mesh, Model을 참고한다.
- 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 가능**하도록 작성한다.
- 과제 코드는 다음의 파일들을 하나의 압축파일로 묶어 **tar.gz** 파일 형식이나 표준 **zip**파일 형식으로만 제출하도록 한다. 이때, 압축파일의 이름은 반드시 'OOOOOOOOO\_HW06.tar.gz (OOOOOOOOO은 자신의 학번)'과 같이 자신의 학번이 드러나도록 제출한다.
  - 1) 소스코드 및 리소스 파일들
  - 2) Makefile