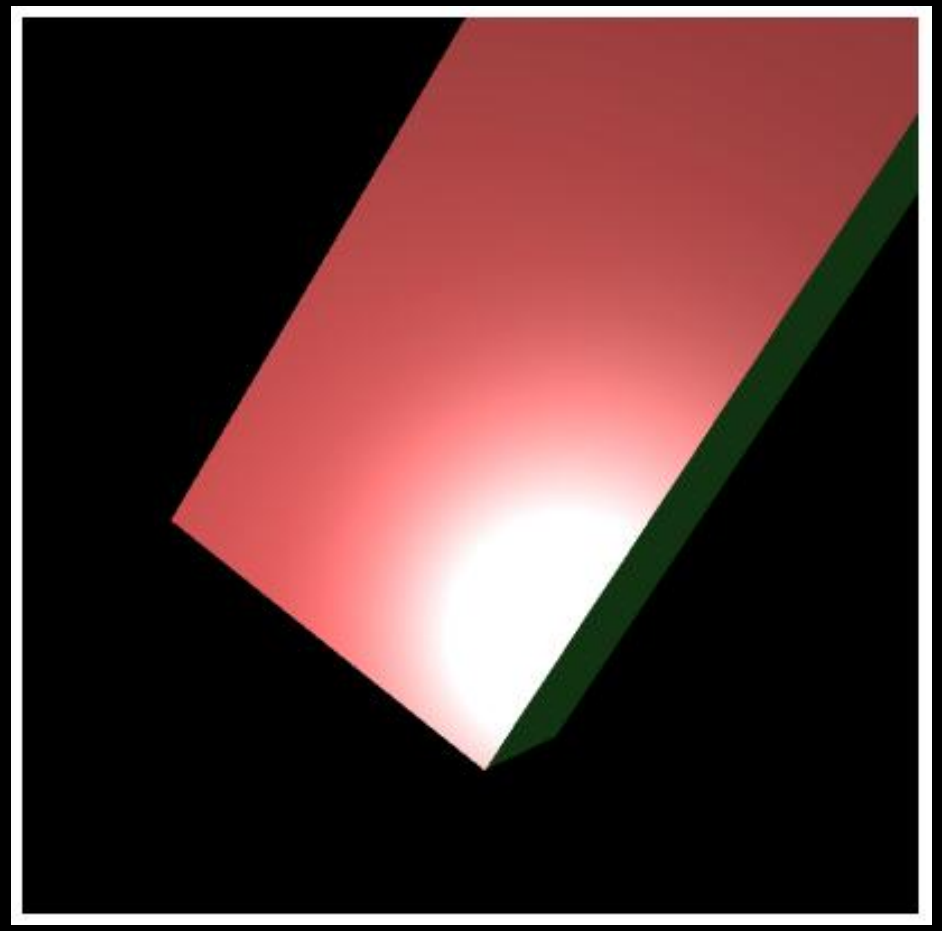
Lab 6

- Download the lab6 template
- You will implement phong shading (per-fragment illumination)
  - A much better illumination than Ex06-3 and Ex06-4
  - You can use mouse to rotate the objects and check the result
- Please this video, this is the result you should have
  - https://www.youtube.com/watch?v=gRnnbXuSjfk&ab_channel=Ko-ChihWang



- Please check “TODO” in the WebGL.js

- You only need to add code in main functions of vertex and fragment shaders (**do NOT change any code outside of main functions of the shaders**)
  - **Follow the TODOs and its explanation one by one to finish this practice**

- Hint:
  - the code you will write here is very similar to the shader code in ”Ex06-4”. The major difference is that most of the code in vertex shader in Ex06-4 is moved to fragment shader here.

```
var VSHADER_SOURCE = `
    attribute vec4 a_Position;
    attribute vec4 a_Color;
    attribute vec4 a_Normal;
    uniform mat4 u_MvpMatrix;
    uniform mat4 u_modelMatrix;
    uniform mat4 u_normalMatrix;
    varying vec3 v_Normal;
    varying vec3 v_PositionInWorld;
    varying vec4 v_Color;
    void main(){
        //TODO-1: transform "a_Position" to clip space and store in "gl_Position"
        //TODO-2: transform "a_Position" to world space and store its first three elements to "v_PositionInWorld"
        //TODO-3: transform normal vector "a_Normal" to world space using "u_normalMatrix" and store the result in "v_Normal",
        //        remember to renormalize the result before storing it to v_Normal
        //TODO-4: set "a_Color" to "v_Color"
    }
`;

var FSHADER_SOURCE = `
    precision mediump float;
    uniform vec3 u_LightPosition;
    uniform vec3 u_ViewPosition;
    uniform float u_Ka;
    uniform float u_Kd;
    uniform float u_Ks;
    uniform float u_shininess;
    varying vec3 v_Normal;
    varying vec3 v_PositionInWorld;
    varying vec4 v_Color;
    void main(){
        // let ambient and diffuse color are v_Color
        // (you can also input them from ouside and make them different)
        vec3 ambientLightColor = v_Color.rgb;
        vec3 diffuseLightColor = v_Color.rgb;
        // assume white specular light (you can also input it from ouside)
        vec3 specularLightColor = vec3(1.0, 1.0, 1.0);

        //TODO-5: calculate ambient light color using "ambientLightColor" and "u_Ka"


        vec3 normal = normalize(v_Normal); //normalize the v_Normal before using it, before it comes from normal vectors interpolation
        //TODO-6: calculate diffuse light color using "normal", "u_LightPosition", "v_PositionInWorld", "diffuseLightColor", and "u_Kd"


        vec3 specular = vec3(0.0, 0.0, 0.0);
        if(nDotL > 0.0) {
            //TODO-7: calculate specular light color using "normal", "u_LightPosition", "v_PositionInWorld",
            //       "u_ViewPosition", "u_shininess", "specularLightColor", and "u_Ks"
            //   You probably can store the result of specular calculation into "specular" variable

        }

        //TODO-8: sum up ambient, diffuse, specular light color from above calculation and put them into "gl_FragColor"
        gl_FragColor = ???
    }
`;
```

What You Should Do for “Submission”

Submission Instruction

- Create a folder
  - Put the html and js files in the folder
  - Zip the folder
  - Rename the zip file to your student ID
    - For example, if your student ID is “40312345s”, rename the zip file to “40312345s.zip”
  - Submit the renamed zip file to Moodle

- Make sure
  - you put all files in the folder to zip
  - You submit the zip file with correct name
- You won’t get any point if
  - the submitted file does not follow the naming rule,
  - TA cannot run your code,
  - or cannot unzip your zip file.