計算機圖學 HW2 Report 111062678 卓榮祥

完成清單：

- ✓ Directional light
- ✓ Point light
- ✓ Spot light
- ✓ Per-pixel lighting / Per-vertex lighting
- ✓ Side-by-side viewport
- ✓ Switch lights & models
- ✓ Dynamic light position, cutoff, shininess
- ✓ Report

Directional light：

```
vec3 position = vec3(1, 1, 1) + light_move;

vec3 direction = vec3(0, 0, 0);


vec3 direction_normalize = normalize(position.xyz - direction);

vec3 Rp = normalize(reflect( -direction_normalize , vertex_normal));


vec3 diffuse = Ip * Kd  * max(dot(vertex_normal, direction_normalize), 0);
vec3 specular = Ip * Ks * pow(max(dot( Rp, viewpoint_vec ), 0), Shininess ) ;


FragColor = vec4(ambient + intensity * diffuse + specular , 1.0f);
```

Point light：

```
vec4 position = v * vec4(vec3(0, 2, 1) + light_move, 1);
//vec3 position = vec3(0, 2, 1) + light_move;

vec3 direction_normalize = normalize(position.xyz - eye_position.xyz );

vec3 Rp = normalize(reflect( -direction_normalize , vertex_normal));


vec3 diffuse = Ip * Kd  * max(dot(vertex_normal, direction_normalize), 0);
vec3 specular = Ip * Ks * pow(max(dot(Rp, viewpoint_vec), 0), Shininess );

float dist = length(position.xyz - eye_position.xyz);
float Fatt = min((1.0 / (0.01 + 0.8 * dist + 0.1 * (dist * dist))), 1);

FragColor = vec4(ambient + Fatt * (intensity * diffuse + specular) , 1.0f);
```

Spot light：

```
vec4 position = v * vec4(vec3(0, 0, 2) + light_move, 1);
//vec3 position = vec3(0, 0, 2) + light_move;
vec3 direction = vec3(0, 0, -1);

float spotlight_effect = 0;

//float cutoff  = 30;

vec3 direction_normalize = normalize(position.xyz - eye_position.xyz );
if ( dot(-direction_normalize, direction) > cos(cutoff * M_PI / 180))
    spotlight_effect = pow(max( dot(-direction_normalize, direction), 0 ), 50);
else
    spotlight_effect = 0;

float dist = length(position.xyz - eye_position.xyz);
float Fatt = min((1.0 / (0.05 + 0.3 * dist + 0.6 * (dist * dist))), 1);

vec3 Rp = normalize(reflect( -direction_normalize , vertex_normal));

vec3 diffuse = Ip * Kd  * max(dot(vertex_normal, direction_normalize), 0);
vec3 specular = Ip * Ks * pow(max(dot(Rp, viewpoint_vec), 0), Shininess );

FragColor = vec4(ambient + Fatt * spotlight_effect * (ambient + diffuse + specular) , 1.0f);
```

Per-pixel lighting / Per-vertex lighting：

基本上如果做完 Per-vertex lighting 也相當於把 Per-pixel lighting 完成了，因為他們的 lighting equation 是相同的，差別只在於 Per-vertex lighting 是在 vertex shader 中計算，而 Per-pixel lighting 是在 fragment shader 中計算。

Side-by-side viewport：

```
lighting_model = 0;
glUniform1i(uniform.iLocLightingModel, lighting_model);
glViewport(0, 0, WINDOW_WIDTH / 2, WINDOW_HEIGHT);
glDrawArrays(GL_TRIANGLES, 0, models[cur_idx].shapes[i].vertex_count);


lighting_model = 1;
glUniform1i(uniform.iLocLightingModel, lighting_model);
//glBindVertexArray(models[cur_idx].shapes[i].vao);
glViewport(WINDOW_WIDTH / 2, 0, WINDOW_WIDTH / 2, WINDOW_HEIGHT);
glDrawArrays(GL_TRIANGLES, 0, models[cur_idx].shapes[i].vertex_count);
```

利用 glViewport()設定畫在兩個 800X800 的 viewport 上，並把 lighting_model 傳到 vertex shader 來辨別是哪個 lighting model。

```
// Call back function for window reshape
void ChangeSize(GLFWwindow* window, int width, int height)
{
    //glViewport(0, 0, width / 2, height);

    //glViewport(width / 2, 0, width / 2, height);
    // [TODO] change your aspect ratio
    WINDOW_WIDTH = width;
    WINDOW_HEIGHT = height;

    proj.aspect = ((float)width / 2) / (float)height;
    setPerspective();

}
```

每次對 windows 做縮放時需要重新調整 projection aspect 以維持 model 比例。

Switch lights & models：

```
case GLFW_KEY_L:
    models[cur_idx].light_move = Vector3(0, 0, 0);
    light_idx = (light_idx + 1) % 3;
    break;
```

修改 KeyCallback 來完成 switch light 的功能，再將 light_idx 透過 uniform 傳入 vertex shader 中以辨別 light 的狀態。

Dynamic light position, cutoff, shininess：

Dynamic light position

```
case LightEdit:
    models[cur_idx].light_move = models[cur_idx].light_move + move ;
```

cutoff

```
case LightEdit:
    if (light_idx == 2) {
        cutoff += yoffset;
    }
    else {
        intensity += yoffset;
    }
```

shininess

```
case ShininessEdit:
    shininess += yoffset;
    break;
```

透過修改 KeyCallback、scroll_callback、cursor_pos_callback function

來完成，一樣將修改後的值傳到 vertex shader 中做計算。