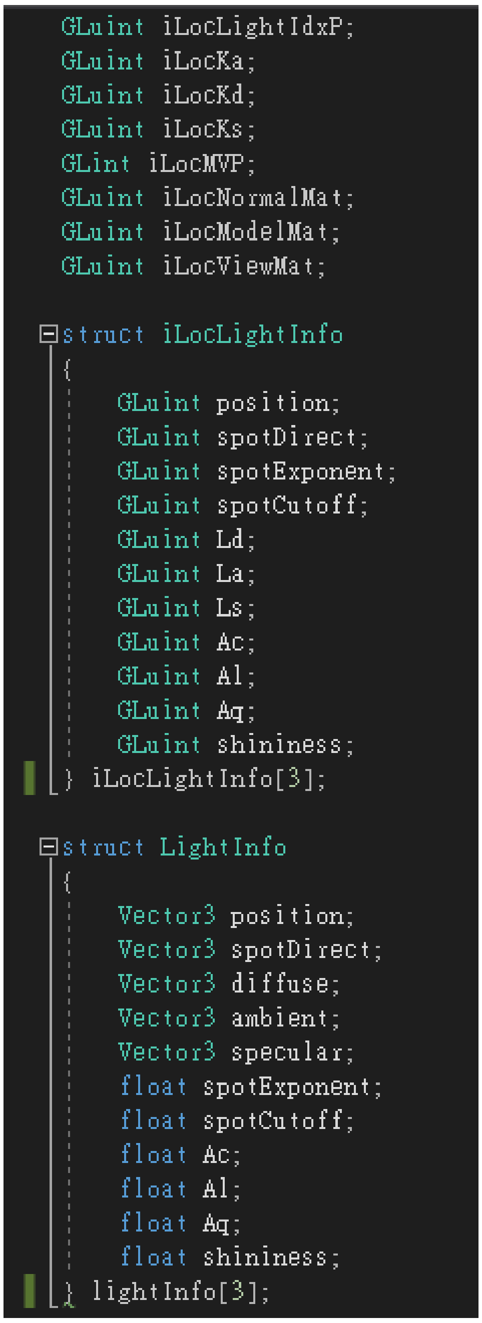
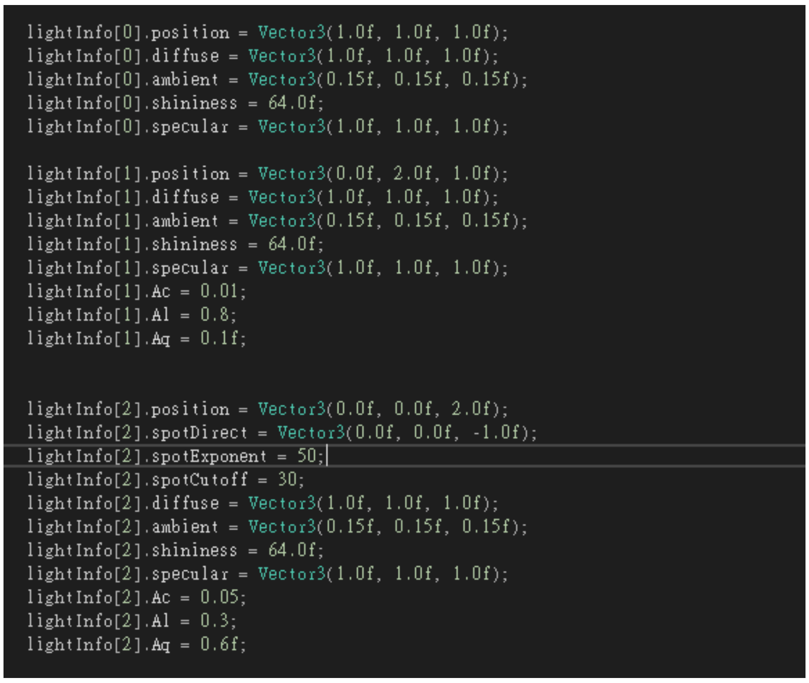
Assignment #2 Lighting

105072123 黃海茵


```
GLuint iLocLightIdxP;
GLuint iLocKa;
GLuint iLocKd;
GLuint iLocKs;
GLint iLocMVP;
GLuint iLocNormalMat;
GLuint iLocModelMat;
GLuint iLocViewMat;

struct iLocLightInfo
{
    GLuint position;
    GLuint spotDirect;
    GLuint spotExponent;
    GLuint spotCutoff;
    GLuint Ld;
    GLuint La;
    GLuint Ls;
    GLuint Ac;
    GLuint Al;
    GLuint Aq;
    GLuint shininess;
} iLocLightInfo[3];

struct LightInfo
{
    Vector3 position;
    Vector3 spotDirect;
    Vector3 diffuse;
    Vector3 ambient;
    Vector3 specular;
    float spotExponent;
    float spotCutoff;
    float Ac;
    float Al;
    float Aq;
    float shininess;
} lightInfo[3];
```
	宣告 shader 會用到的所有變數，以及建立等等會用到的 struct。

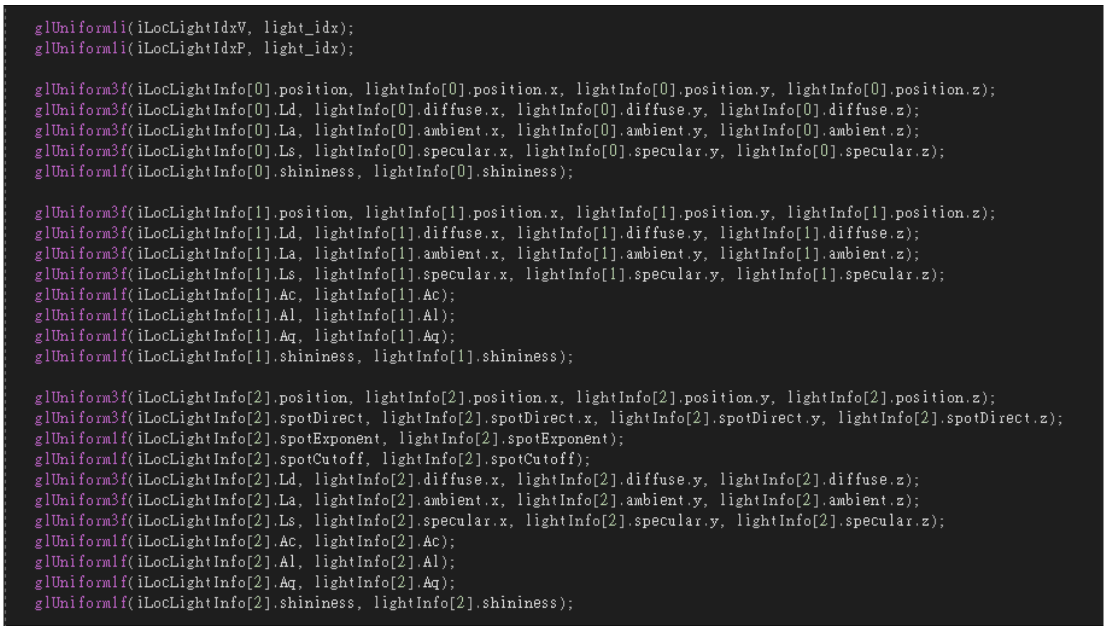


```
lightInfo[0].position = Vector3(1.0f, 1.0f, 1.0f);
lightInfo[0].diffuse = Vector3(1.0f, 1.0f, 1.0f);
lightInfo[0].ambient = Vector3(0.15f, 0.15f, 0.15f);
lightInfo[0].shininess = 64.0f;
lightInfo[0].specular = Vector3(1.0f, 1.0f, 1.0f);

lightInfo[1].position = Vector3(0.0f, 2.0f, 1.0f);
lightInfo[1].diffuse = Vector3(1.0f, 1.0f, 1.0f);
lightInfo[1].ambient = Vector3(0.15f, 0.15f, 0.15f);
lightInfo[1].shininess = 64.0f;
lightInfo[1].specular = Vector3(1.0f, 1.0f, 1.0f);
lightInfo[1].Ac = 0.01;
lightInfo[1].Al = 0.8;
lightInfo[1].Aq = 0.1f;


lightInfo[2].position = Vector3(0.0f, 0.0f, 2.0f);
lightInfo[2].spotDirect = Vector3(0.0f, 0.0f, -1.0f);
lightInfo[2].spotExponent = 50;
lightInfo[2].spotCutoff = 30;
lightInfo[2].diffuse = Vector3(1.0f, 1.0f, 1.0f);
lightInfo[2].ambient = Vector3(0.15f, 0.15f, 0.15f);
lightInfo[2].shininess = 64.0f;
lightInfo[2].specular = Vector3(1.0f, 1.0f, 1.0f);
lightInfo[2].Ac = 0.05;
lightInfo[2].Al = 0.3;
lightInfo[2].Aq = 0.6f;
```

↑　初始化三種 light 的所有 data



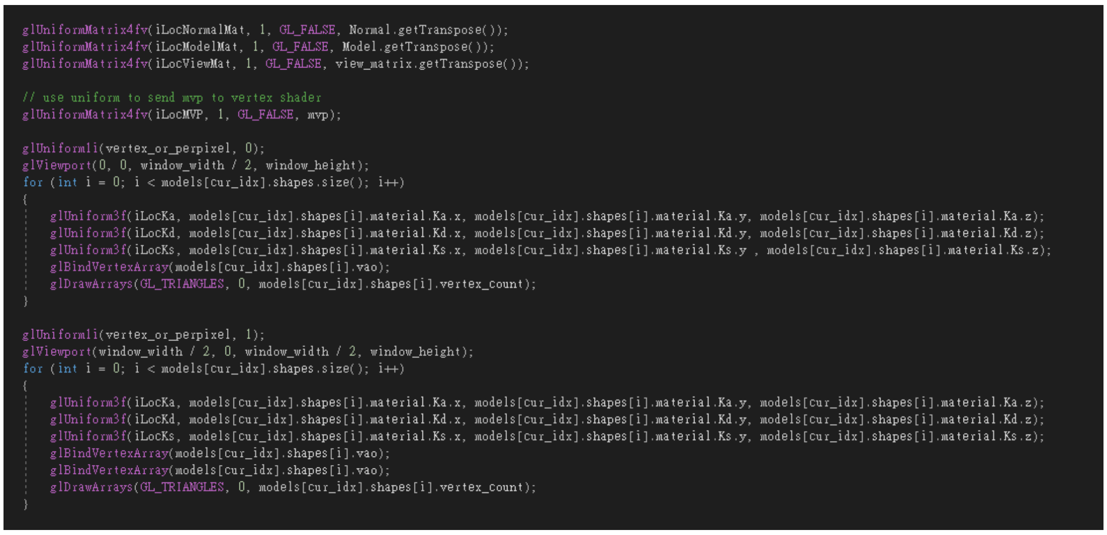
```
glUniform1i(iLocLightIdxV, light_idx);
glUniform1i(iLocLightIdxP, light_idx);

glUniform3f(iLocLightInfo[0].position, lightInfo[0].position.x, lightInfo[0].position.y, lightInfo[0].position.z);
glUniform3f(iLocLightInfo[0].Ld, lightInfo[0].diffuse.x, lightInfo[0].diffuse.y, lightInfo[0].diffuse.z);
glUniform3f(iLocLightInfo[0].La, lightInfo[0].ambient.x, lightInfo[0].ambient.y, lightInfo[0].ambient.z);
glUniform3f(iLocLightInfo[0].Ls, lightInfo[0].specular.x, lightInfo[0].specular.y, lightInfo[0].specular.z);
glUniform1f(iLocLightInfo[0].shininess, lightInfo[0].shininess);

glUniform3f(iLocLightInfo[1].position, lightInfo[1].position.x, lightInfo[1].position.y, lightInfo[1].position.z);
glUniform3f(iLocLightInfo[1].Ld, lightInfo[1].diffuse.x, lightInfo[1].diffuse.y, lightInfo[1].diffuse.z);
glUniform3f(iLocLightInfo[1].La, lightInfo[1].ambient.x, lightInfo[1].ambient.y, lightInfo[1].ambient.z);
glUniform3f(iLocLightInfo[1].Ls, lightInfo[1].specular.x, lightInfo[1].specular.y, lightInfo[1].specular.z);
glUniform1f(iLocLightInfo[1].Ac, lightInfo[1].Ac);
glUniform1f(iLocLightInfo[1].Al, lightInfo[1].Al);
glUniform1f(iLocLightInfo[1].Aq, lightInfo[1].Aq);
glUniform1f(iLocLightInfo[1].shininess, lightInfo[1].shininess);

glUniform3f(iLocLightInfo[2].position, lightInfo[2].position.x, lightInfo[2].position.y, lightInfo[2].position.z);
glUniform3f(iLocLightInfo[2].spotDirect, lightInfo[2].spotDirect.x, lightInfo[2].spotDirect.y, lightInfo[2].spotDirect.z);
glUniform1f(iLocLightInfo[2].spotExponent, lightInfo[2].spotExponent);
glUniform1f(iLocLightInfo[2].spotCutoff, lightInfo[2].spotCutoff);
glUniform3f(iLocLightInfo[2].Ld, lightInfo[2].diffuse.x, lightInfo[2].diffuse.y, lightInfo[2].diffuse.z);
glUniform3f(iLocLightInfo[2].La, lightInfo[2].ambient.x, lightInfo[2].ambient.y, lightInfo[2].ambient.z);
glUniform3f(iLocLightInfo[2].Ls, lightInfo[2].specular.x, lightInfo[2].specular.y, lightInfo[2].specular.z);
glUniform1f(iLocLightInfo[2].Ac, lightInfo[2].Ac);
glUniform1f(iLocLightInfo[2].Al, lightInfo[2].Al);
glUniform1f(iLocLightInfo[2].Aq, lightInfo[2].Aq);
glUniform1f(iLocLightInfo[2].shininess, lightInfo[2].shininess);
```

↑　每次做 Render 前都傳入 shader 更新的 light data 參數



```
glUniformMatrix4fv(iLocNormalMat, 1, GL_FALSE, Normal.getTranspose());
glUniformMatrix4fv(iLocModelMat, 1, GL_FALSE, Model.getTranspose());
glUniformMatrix4fv(iLocViewMat, 1, GL_FALSE, view_matrix.getTranspose());

// use uniform to send mvp to vertex shader
glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);

glUniform1i(vertex_or_perpixel, 0);
glViewport(0, 0, window_width / 2, window_height);
for (int i = 0; i < models[cur_idx].shapes.size(); i++)
{
    glUniform3f(iLocKa, models[cur_idx].shapes[i].material.Ka.x, models[cur_idx].shapes[i].material.Ka.y, models[cur_idx].shapes[i].material.Ka.z);
    glUniform3f(iLocKd, models[cur_idx].shapes[i].material.Kd.x, models[cur_idx].shapes[i].material.Kd.y, models[cur_idx].shapes[i].material.Kd.z);
    glUniform3f(iLocKs, models[cur_idx].shapes[i].material.Ks.x, models[cur_idx].shapes[i].material.Ks.y , models[cur_idx].shapes[i].material.Ks.z);
    glBindVertexArray(models[cur_idx].shapes[i].vao);
    glDrawArrays(GL_TRIANGLES, 0, models[cur_idx].shapes[i].vertex_count);
}

glUniform1i(vertex_or_perpixel, 1);
glViewport(window_width / 2, 0, window_width / 2, window_height);
for (int i = 0; i < models[cur_idx].shapes.size(); i++)
{
    glUniform3f(iLocKa, models[cur_idx].shapes[i].material.Ka.x, models[cur_idx].shapes[i].material.Ka.y, models[cur_idx].shapes[i].material.Ka.z);
    glUniform3f(iLocKd, models[cur_idx].shapes[i].material.Kd.x, models[cur_idx].shapes[i].material.Kd.y, models[cur_idx].shapes[i].material.Kd.z);
    glUniform3f(iLocKs, models[cur_idx].shapes[i].material.Ks.x, models[cur_idx].shapes[i].material.Ks.y, models[cur_idx].shapes[i].material.Ks.z);
    glBindVertexArray(models[cur_idx].shapes[i].vao);
    glBindVertexArray(models[cur_idx].shapes[i].vao);
    glDrawArrays(GL_TRIANGLES, 0, models[cur_idx].shapes[i].vertex_count);
}
```

↑ 讓 shader 在 vertex 和 pixel 接受到材質，就能夠顯示出正確的顏色。

因為這次要分成左右兩邊不同的模式，所以要用 viewport 來控制，將整個 window 切成左右兩部分，再將 model 分別畫在左右兩邊。

參數設定完整的話，在 shader 中三種燈光的公式，就都能夠套用講義中以及網路上查到的方式來實作。

然後鍵盤和滑鼠的控制和上次的作業一差不多，就不多做說明。