姓名: 成文瑄
學號: 0716004

首先先用 createshader 說要建立兩個 shader，前面的參數是 shader 檔的位置， 後面的參數是它是哪種 shader。再來用 createprogram 來把建立的 shader 連接起來。

```
    GLuint vert = createShader("Shaders/vertexShader.vert", "vertex");
    GLuint frag = createShader("Shaders/fragmentShader.frag", "fragment
");

    program = createProgram(vert, frag);
```

接著要 create vao 和 vbo。vbo 是一個封裝頂點的各種資訊的東西，這樣傳進去 shader 的時候就不需要一個 vertex 一個 vertex 的傳進去，而是可以一次送一堆 vertex 的資料進去。但是如果 vbo 的物件太多的話，每次使用的時候都要 bind, 設各個屬性之類的很麻煩，所以要用 vao 來封裝 vbo，每次使用的時候只要 bind vao 就好。所以首先要先產生 vao，glgenvertexarray 是產生出 vao 的名字(id)，接著再用 glbindvertexarray 來把剛才產生的 vao 名字 bind 到一個 vao object 上，所以 vao 就建好了。

```
    // Set up VAO
    glGenVertexArrays(1, &VAO);
    glBindVertexArray(VAO);
```

接著要產生出在這個 vao 管理之下 vbo（在 vao 創立後產生的 vbo 預設都會綁在它下面）。首先一樣用 glgenbuffer、glbindbuffer 來產生 vbo 名字、bind vbo。接著是把資料用 glbufferdata 送進 vbo，第二個參數是這個 vbo 要多大，第三個參數是傳進去的 data，第四個參數是要如何畫它。再來用 glenablevertexattribarray 來開一個要傳進 shader 的通道（這裡的數字要和 shader 中 layout 後面的數字相同。

```
layout(location=2) in vec2 texcoord_to_vert;
```

再來是用 glvertexattribpointer 來告訴 shader 要怎麼讀這個 vbo，第一個參數是用哪個通道，第二個是一次會傳幾個東西（要從 vbo 拿幾個），第五個是下一個頂點要隔多少，第六個是 offset（從 vbo 的哪裡開始讀）。再來就是用完 vbo 後把 vbo、vao unbind 掉。

```
glGenBuffers(1, &VBO[2]);
    glBindBuffer(GL_ARRAY_BUFFER, VBO[2]);
    glBufferData(GL_ARRAY_BUFFER, sizeof(float) * (model-
>texcoords.size()), model->texcoords.data(), GL_STATIC_DRAW);
```

```
    glEnableVertexAttribArray(2);
    glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, sizeof(float) * 2,
(GLvoid*)0);
    glBindBuffer(GL_ARRAY_BUFFER, 0);

    // Unbind VAO
    glBindVertexArray(0);
```

接著要開始畫的時候，在把東西傳進 shader 之前要 use program，接著就是建立一些要傳進去的東西（像是 mat4、texture 等等)，然後傳進去前要先用 glgetuniformlocation 來找到這個變數在 shader 中的位置（第二個參數和 shader 中的變數名要對起來），

```
uniform mat4 P;
```

然後在依照變數的 type 用 gluniform...傳進去 shader（用 uniform 傳的是各個頂點都相同、或是只在一個 shader 被使用的資料）。

```
glm::mat4 P = getP();
    GLint pmatLoc = glGetUniformLocation(program, "P");
    glUniformMatrix4fv(pmatLoc, 1, GL_FALSE, &P[0][0]);
```

如果是 texture 的話，要先 active 要用的 texture，然後把它跟 texture 的資料 bind 起來，在像上面一樣傳進 shader。

```
glActiveTexture(GL_TEXTURE1);
    glBindTexture(GL_TEXTURE_2D, modeltexture);
    GLint textLoc = glGetUniformLocation(program, "Texture");
    glUniform1i(textLoc, 1);
```

接著也把 vao bind 起來這樣 shader 可以拿到在裡面 vbo 的資料。接著就是用 gldrawarrays 來畫圖！

```
    glBindVertexArray(VAO);
    glDrawArrays(GL_QUADS, 0, 4 * model->fNum);
```

最後用完之後就把大家都 unbind 掉。

```
glBindVertexArray(0);
    glBindTexture(GL_TEXTURE_2D, 0);
    glActiveTexture(0);
    glUseProgram(0);
```

再來是 vertex shader 的部分，拿到傳進來的變數之後（in 和 uniform 的部分），在 main 裡一定要有的是 gl_position（也是輸出），是頂點輸出的位置（把傳進來的三維位置加上一維後做 model view、projection)，out 表示要輸出給 fragment shader 的東西（名字和 fragment shader 裡的要對起來），這裡直接把送進來的 normal、texture 座標送給 shader。

```
layout(location=0) in vec3 position_to_vert;
layout(location=1) in vec3 normal_to_vert;
layout(location=2) in vec2 texcoord_to_vert;

uniform mat4 P;
uniform mat4 V;
uniform mat4 M;

out vec2 textcoord_to_frag;
out vec3 normal_to_frag;

void main()
{
    gl_Position = P * V * M * vec4(position_to_vert, 1.0);
    textcoord_to_frag = texcoord_to_vert;
    normal_to_frag = normal_to_vert;
}
```

在 fragment shader 中，拿到由 vertex shader 中來的頂點位置後，會貼貼圖、算顏色，做內插。texture2D 表示 return 給定的座標上的 texture 顏色，然後輸出 output 顏色。

```
in vec2 textcoord_to_frag;
in vec3 normal_to_frag;
out vec4 color_out;

void main()
{

    color_out = mix(texture2D(Texture, textcoord_to_frag), texture2D(Te
xture2, textcoord_to_frag), 0.3 * text2);
    color_out = color_out + vec4( n, n, n, 1.0f) ;
}
```

問題的話因為助教的 ppt 寫的還滿詳細的，再 google 後，基本上沒遇到什麼問題。

Bonus 的話我做了四個按鍵：
按鍵 s 會讓音樂盒停下來，實作方法就是讓轉的角度+0

```
    if (stop)
        angle = angle;
    else
        angle += 0.1;
```

按鍵 j 會讓月亮伊布開始跳，做法是再乘上一個 y 是 sin 函數的 translate 矩陣

```
    if (jump) {
        M = glm::translate(M, glm::vec3(0, 1.3 * sin(theta_j / 180 * pi
) + 1.3, 0));
        theta_j = theta_j + 0.25;
    }
```

按鍵 t 會讓月亮伊布身上混合一個星星的貼圖，再傳一個貼圖進去，用 mix 來混合兩個貼圖。

```
color_out = mix(texture2D(Texture, textcoord_to_frag), texture2D(Textur
e2, textcoord_to_frag), 0.3 * text2);
```

按鍵 i 則是讓月亮伊布的顏色由暗變亮，透過加上一個 rgb 顏色都是一樣的向量來達成（n 的值每次會加一點點）。

```
if (increment) {
        GLint Loc2 = glGetUniformLocation(program, "n");
        glUniform1f(Loc2, float(incre));
        //cout << incre << endl;
        incre = incre + 0.00005;
        if (incre > 0.9)
            incre = 0;
    }
```

```
color_out = color_out + vec4( n, n, n, 1.0f) ;
```