HW3 說明檔

- **讀 texture**

因為不是每個 material 都有 map_Kd，所以將 Ka、kd、ks、Ns 這些 Set 好之後要判斷是否有在 mtl 檔案中讀到 map_kd，如果有的話就 new 一個 ImageTexture 物件，並將此物件作為 PhongMaterial 中的 MapKd，同時將 HaveMapKd(用來給之後 shader 判斷 material 的 Kd 是否有 map_Kd 可以取代)這個參數設為 1。

```
            getline(inmtlfile, mtlLine);
            istringstream sinNs(mtlLine);//用空格做切割
            sinNs >> map_Kd;
            cout << "map_kd :" << map_Kd << endl;
            if (map_Kd == "map_Kd") {
                mat->SetHaveMapKd(0);
                continue;
            }
            else {
                sinNs >> mapKdPath;
                cout << "have map_Kd path : " << mapKdPath << endl;
                ImageTexture* imgtex = new ImageTexture(mapKdPath);
                mat->SetMapKd(imgtex);
                mat->SetHaveMapKd(1);
                getline(inmtlfile, mtlLine);//讀空行
            }
```

- **PhongShadingDemoShaderProg 新增 locmapKd 和 locHaveMapKd**

在 fragment shader 中需要傳入 mapKd 和 haveMapKd。

mapKd：有的話需取代原本的 Kd。

haveMapKd：記錄此 material 有沒有 mapKd 的參數。

```
    // Texture data.
    // ------------------------------------------------------------
    // Add your data for supporting textures.
    // ------------------------------------------------------------
    GLint locMapKd;
    GLint locHaveMapKd;
```

```
    // Add your methods for supporting textures.
    // ------------------------------------------------------------
    GLint GetLocMapKd() const { return locMapKd; }
    GLint GetLocHaveMapKd() const { return locHaveMapKd; }
```

```
    // ------------------------------------------------------------
    // Add texture
    locMapKd = glGetUniformLocation(shaderProgId, "mapKd");
    locHaveMapKd = glGetUniformLocation(shaderProgId, "haveMapKd");
}
```

- **Vertex shader**

新增 layout (location = 2) in vec2 Texcoord 和 out vec2 iTexcorrd，並將 iTexcorrd 內插到 fragment shader。

```
#version 330 core

layout (location = 0) in vec3 Position;
layout (location = 1) in vec3 Normal;
layout (location = 2) in vec2 Texcoord;

// Transformation matrix.
uniform mat4 worldMatrix;
uniform mat4 normalMatrix;
uniform mat4 MVP;

// Data pass to fragment shader.
out vec3 iPosWorld;
out vec3 iNormalWorld;
out vec2 iTexcorrd;
```

```
void main()
{
    gl_Position = MVP * vec4(Position, 1.0);

    //pass vertex attributes.
    vec4 positionTmp = worldMatrix * vec4(Position, 1.0);
    iPosWorld = positionTmp.xyz / positionTmp.w;

    iNormalWorld = (normalMatrix * vec4(Normal, 0.0)).xyz;
    iTexcorrd = Texcoord;
}
```

- **Fragment shader**

多了 mapKd 和 haveMapKd，texColor 是查貼圖後得到的結果，用來取代原本的 Kd，applyKd 是最後的 Kd，需利用 haveMapKd 判斷現在這個 material 有沒有 map_kd 可以取代原來 Kd，有的話就用 texColor 去取代，沒有的話就用原來 mtl 檔案中讀到的 Kd 去做 lighting。

```
// Data from vertex shader.
in vec3 iPosWorld;
in vec3 iNormalWorld;
in vec2 iTexcorrd;

// ------------------------------------------------------------
// Add your uniform variables.
// ------------------------------------------------------------
uniform sampler2D mapKd;
uniform int haveMapKd;
```

```
void main()
{
    vec3 texColor = texture2D(mapKd, iTexcorrd).rgb;
    vec3 applyKd;
    if(haveMapKd == 1){
        applyKd = texColor;
    }
    else{
        applyKd = Kd;
    }
```

- **主要畫圖部分**

1. RenderSceneCB()

   從 GetHaveMapKdValue 函式中取的 HaveMapKd 的值，設定給 shader，並且 bind texture。

```
//Material properties.
for (int Count = 0; Count < pMesh->GetNumSubMeshes(); Count++) {
    glUniform3fv(phongShadingShader->GetLocKa(), 1, glm::value_ptr(pMesh->GetKa_of_SubMeshes(Count)));
    glUniform3fv(phongShadingShader->GetLocKd(), 1, glm::value_ptr(pMesh->GetKd_of_SubMeshes(Count)));
    glUniform3fv(phongShadingShader->GetLocKs(), 1, glm::value_ptr(pMesh->GetKs_of_SubMeshes(Count)));
    glUniform1f(phongShadingShader->GetLocNs(), pMesh->GetNs_of_SubMeshes(Count));

    //glUniform1i(phongShadingShader->GetLocHaveMapKd(), 0);
    glUniform1i(phongShadingShader->GetLocHaveMapKd(), pMesh->GetHaveMapKdValue(Count));
    pMesh->GetMapKd_of_SubMeshes(Count)->Bind(GL_TEXTURE0);
    glUniform1i(phongShadingShader->GetLocMapKd(), 0);


    pMesh->Draw(pMesh->Get_subMeshes(Count));
}
```

2. TriangleMesh::Draw(const SubMesh& obj)

   因為需要用到 position、normal、texture coordinate，所以在 glEnableVertexAttribArray 需要設定 index 0、1、2，在 glVertexAttribPointer 中 index=1 時，起始位置要設為 GLvoid(*)12，因為前面有 position 的 xyz 位置; index=2 時，起始位置要設為 GLvoid(*)24，因為前面有 position 和 normal 的 xyz 位置。

```
void TriangleMesh::Draw(const SubMesh& obj) {
    glEnableVertexAttribArray(0);
    glEnableVertexAttribArray(1);
    glEnableVertexAttribArray(2);
    glBindBuffer(GL_ARRAY_BUFFER, vbo);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 32, 0);
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 32, (const GLvoid*)12);
    glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, 32, (const GLvoid*)24);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, obj.iboId);
    glDrawElements(GL_TRIANGLES, obj.vertexIndices.size(), GL_UNSIGNED_INT, 0);
    glDisableVertexAttribArray(0);
    glDisableVertexAttribArray(1);
    glDisableVertexAttribArray(2);
}
```

- **讓 Skybox 旋轉**

在 skybox class 中新增 SetworldMatrix，在 RenderSceneCB 中多加 skybox 的旋轉，將算好的 WorldMatrix 設定至 skybox class 中，並在 skybox Render function 中的 MVP 多乘上算好的 WorldMatrix。

```
// Render skybox. --------------------------------------------------------------
if (skybox != nullptr) {
    // ----------------------------------------------------
    // Add your code to rotate the skybox.
    // ----------------------------------------------------
    static float curSkyboxRotationY = 0.0f;
    const float SkyboxrotStep = 0.005f;
    curSkyboxRotationY += SkyboxrotStep;
    glm::mat4x4 S = glm::scale(glm::mat4x4(1.0f), glm::vec3(1.5f, 1.5f, 1.5f));
    glm::mat4x4 R = glm::rotate(glm::mat4x4(1.0f), glm::radians(curSkyboxRotationY), glm::vec3(0, 1, 0));
    skybox->SetworldMatrix(S * R);
    skybox->Render(camera, skyboxShader);
}
// -----------------------------------------------------------------------------
```

```
public:
    // Skybox Public Methods.
    Skybox(const std::string& texImagePath, const int nSlices,
           const int nStacks, const float radius);
    ~Skybox();
    void Render(Camera* camera, SkyboxShaderProg* shader);

    void SetRotation(const float newRotation) { rotationY = newRotation; }
    void SetworldMatrix(const glm::mat4x4 worldMat) { worldMatrix = worldMat; }

    ImageTexture* GetTexture() { return panorama; };
    float GetRotation() const  { return rotationY; }

private:
    // Skybox Private Methods.
    static void CreateSphere3D(const int nSlices, const int nStacks, const float radius,
                    std::vector<VertexPT>& vertices, std::vector<unsigned int>& indices);

    // Skybox Private Data.
    GLuint vboId;
    GLuint iboId;
    std::vector<VertexPT> vertices;
    std::vector<unsigned int> indices;

    SkyboxMaterial* material;
    ImageTexture* panorama;
    glm::mat4x4 worldMatrix;
    float rotationY;
};
```

```
void Skybox::Render(Camera* camera, SkyboxShaderProg* shader)
{
    glEnableVertexAttribArray(0);
    glEnableVertexAttribArray(1);

    glBindBuffer(GL_ARRAY_BUFFER, vboId);
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(VertexPT), 0);
    glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, sizeof(VertexPT), (const GLvoid*)12);

    shader->Bind();

    // Set transform.
    // -------------------------------------------------------
    // TODO: modify code here to rotate the skybox.
    glm::mat4x4 MVP = camera->GetProjMatrix() * camera->GetViewMatrix() * worldMatrix;
```