TP5

Shadow maps

Les librairies suivantes sont utilisées dans BaseGL:

- OpenGL, for accessing your graphics processor
- GLEW, for accessing modern OpenGL extensions
- GLFW to interface OpenGL and the window system of your operating system
- GLM, for the basic mathematical tools (vectors, matrices, etc.).

Pour compiler BaseGL, nous utiliserons cmake:
cd <path-to-BaseGL>
mkdir build cd build cmake .. cd .. cmake --build build

Pour lancer le programme : cd <path-to-BaseGL>

Initialement, le programme charge un maillage et l'affiche devant un plan. Les interactions basiques sont implémentées e.g., navigation camera (appuyer sur 'h' pour afficher l'aide). L'idée Générale de ce programme est :

- De définir et manipuler des objets C++ représentant les entités de l'application dans Main.cpp comme variables globales
- Les intialiser dans init ()
- Les utiliser dans render () pour synthétiser une image, en les attachant aux structures définies dans le code du shader code, en les passants de variables des classes C++ au shaders GPU (par exemple les matrices modelview et projection).

But du TP

Vous apprendrez à :
- Comment utiliser des Frame Buffer Objects, en particulier dans le contexte des shadow map
- Comment mettre en place des shadow maps pour plusieurs lumières dans votre scène 3D
- Comment filtrer les ombres ou calculer des ombres douces

Actuellement, le modèle d'éclairage implémenté dans le fragment shader. Il y a une seule lumière ponctuelle dans la scène, et le BRDF est un modèle standard de Phong Diffus + Spéculaire.

Analysez la structure de la fonction

```
void Scene::render() ;
```

Dans la première passe, pour chaque lumière de la scène, un objet Frame Buffer est attaché (voir classe), et la géométrie de la scène est rendue du point de vue de la lumière (vous devrez implémenter les paramètres de la caméra plus tard), avec seulement la profondeur comme sortie active. Le shader utilisé est à cette étape est shadomMapShaderProgramPtr.

Dans la seconde passe, le shader principal shaderProgramPtr est utilisé, et la géométrie est rendue à l'écran. Plus tard, vous utiliserez les shadow maps calculées lors de la première passe pour rendre les ombres en temps réel.

1) Lumières

Ajouter un nombre NL > 1 de lumières dans la scene.
Il faut adapter le code dans le fragment shader, et en particulier **il faut utiliser un array de lumières comme variable uniform**.

Votre fragment shader devrait contenir les lignes suivantes :

```
struct LightSource {     vec3 position;
    // etc ...
}; int number_of_lights = 3; // for example

uniform LightSource lightSources[ 3 ];
```

2) Construction des shadow maps

a) Définir une caméra appropriée pour chaque lumière de votre scène. Ce calcul peut s'effectuer dans

```
void Light::setupCameraForShadowMapping(glm::vec3 scene_center , float scene_radius)
```

Vous pouvez utiliser les fonctions suivantes de la librairie GLM :
`glm::ortho<float>  /  glm::perspective<float> glm::lookAt`

Pour voir à quoi ressemblent vos shadow maps, exécutez le programme et appuyez sur «s». Il enregistrera autant de shadow maps que de lumières dans les fichiers PPM.

b) Analysez le shader de shadow map (FragmentShader_shadowMap.glsl / VertexShader_shadowMap.glsl).

Ce shader est assez trivial : il restitue simplement la géométrie du point de vue de la lumière (dont vous venez de calculer la caméra) et affiche la profondeur.

3) Rendre les ombres

Vous allez maintenant utiliser les shadow maps que vous avez calculées afin de rendre les ombres en temps réel. Cette étape devrait être la plus longue. Vous pouvez vous référer aux diapositives du cours.

Il y a plusieurs étapes requises :

- Donnez la texture de la carte des ombres ainsi que la caméra correspondante de la carte des ombres au shader principal.

- Étant donné un fragment, pour chaque lumière, testez si le fragment se trouve dans l'ombre ou s'il est éclairé par la lumière.

Implémentez une heuristique simple pour éviter le célèbre artefact d'auto-ombrage.

4) Bonus : Shadow filtering

Maintenant que vous savez comment tester si un point 3D est dans une ombre ou non, vous pouvez implémenter des schémas de filtrage simples pour calculer les ombres douces.