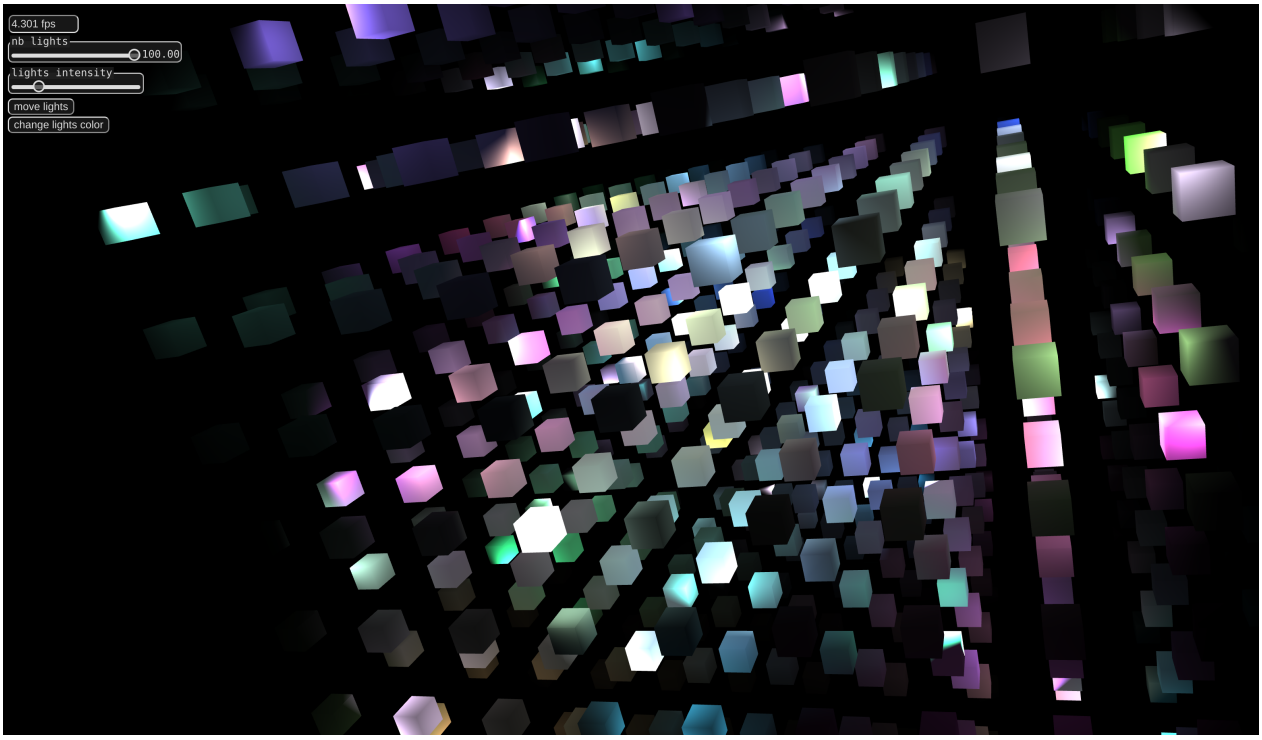
Programmation graphique 3D
M1 Informatique I3D - S1 Automne 2020–2021
Travaux Pratiques

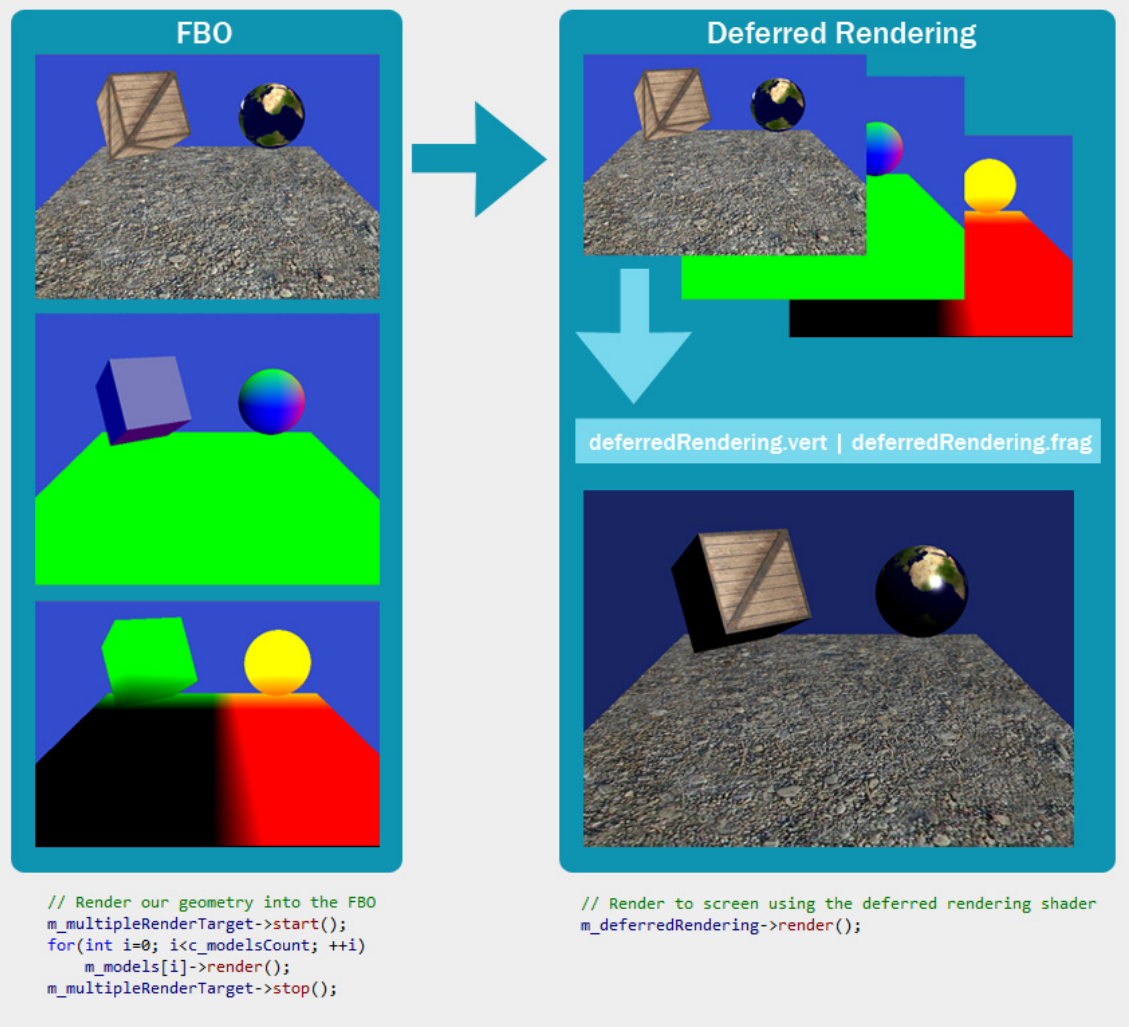
TP 7 : Rendu Offscreen - FBO
Application #2 : Deffered Shading

Introduction

En partant du programme de base qui fait du shading "classique" (ou "forward shading"), on se rend compte que si on ajoute **beaucoup de sources lumineuse** dans la scène, **les performances chutent drastiquement** (voir FPS). Et oui, ça fait beaucoup de calcul pour tous les fragments !



L'idée principale du "deffered shading" est de **séparer le rendu en 2 passes** : "visibilité" et "lighting/shading". On calcule d'abord ce qui est visible puis on fait ensuite le lighting/shading avec un shader complexe **que sur les éléments visibles** de la scène 3D.



1ère passe de rendu : Pas de shading ! Juste une passe de visibilité ! Pour stocker l'information de géométrie dans la texture attachée au FBO : position, normal, kd (materiaux).

- Faire le rendu de la scène (modèle 3D), en rendu offscreen dans un FBO
- Remplir le G-buffer (texture attachées au FBO)

2ème passe de rendu : Faire le shading/lighting dans le fragment shader.

- Faire le rendu d'un quad fullscreen
- Lire les données calculées à la 1ère passe de rendu, depuis les textures

Exercice 1 : Deferred shading

Etape 1 : rendu offscreen (G-buffer)

- Créer un FBO
- Créer et attacher 3 textures pour exporter : les positions, les normales et les couleurs de votre objet 3D, sans appliquer de transformations.(InternalFormat = gl.RGBA32F, type = gl.FLOAT)
- Il faudra utiliser 3 COLOR ATTACHMENT différents
- Faire le rendu de la scène en faisant avant un bind de votre FBO et en initialisant le viewport à la taille du FBO
- Vertex shader :
  1. exporter vos positions et normales ***dans le repère de la caméra***
  2. écrire gl_Position comme d'habitude.

- Fragment shader : dans votre fragment shader, vous aurez à utiliser la technique MRT, "multiple render target", en exportant vos données dans 3 variables de sorties (correspondant aux 3 COLOR ATTACHMENT du FBO):
    1. layout (location=0) out vec3 position;
    2. layout (location=1) out vec3 normal;
    3. layout (location=2) out vec3 color; // pour stocker par exemple le composant diffuse "Kd"

Etape 2 : rendu Classique avec lighthing

Faire le rendu de la scène avec un quad en plein écran, en pensant à binder/activer les 3 textures précédentes du FBO afin de pour récupérer vos positions et normales dans votre shader.

- Vertex shader :
    1. rendu d'un quad plein écran (reprenez mon shader de TP précédent qui dessine un quad de manière procédural avec un triangle 2 fois plus grand que l'écran)
    2. générer en sortie une coordonnées "uv" de textures entre 0 et 1
- Fragment shader : Le calcul se fera au Fragment Shader : chaque fragment (pixel) pour aller lire les infos dans les textures et faire le calcul de lighting, dans l'espace de coordonnées exportés à la première étape du FBO, soit ici "l'espace caméra".