



MASTER INFORMATIQUE, SYNTHÈSE
D'IMAGES ET CONCEPTION GRAPHIQUE

Projet : Moteur 3D

Florian AUBERVAL

Encadrant :

Maxime MARIA

8 Janvier 2023

1 Introduction

Afin de compléter cette UE, nous avons pour projet de réalisé au choix différentes méthodes d'effet 3D plus ou moins complexe vu en cours.

Pour mener à bien ce projet, nous allons reprendre la base d'application de visualisation de scène interactive qui a été développée au fil des TP. Nous utilisons la technologie OpenGL, version 4, pour la programmation de notre moteur de rendu. Et dans ce cadre, nous utiliserons le langage de programmation C++, ainsi que l'environnement de programmation Visual Studio Community, version 2022.

Le dépôt du projet est disponible sur :

https://github.com/Me-k-01/Projet_Moteur_3D

2 Les différents labworks

Durant les 6 TP, a été demandée la réalisation de 6 espaces de travail lab_work. Par défaut, le projet s'ouvre sur le dernier labwork, celui du projet avec la scène Sponza version Crytek.

Voici un bref descriptif de ce qui a pu être codé en relation avec ces différents TP:

2.1 LabWork1

Dans le LabWork de ce premier TP, je me suis initié à OpenGL par l'affichage d'un premier triangle rouge.

2.2 LabWork2

Dans cet espace de travail, a été réalisé l'affichage d'un cube coloré selon ses sommets, et animé selon un mouvement sinusoïdal. Un menu de paramètre a également été implémenté.

2.3 LabWork3

Dans ce troisième TP, j'ai créé un cube 3D, que j'ai placé dans la scène grâce aux matrices de changement d'espace. Un slider de FOVY a été ajouté et la méthode handleEvents a été implémentée pour l'interactivité clavier souris.

2.4 LabWork4

Dans le cadre de ce quatrième TP, a d'abord été affiché le maillage d'un lapin (cet affichage est accessible via #define MODEL_1 en haut du fichier LabWork), puis celui d'une scène représentant une salle de classe. La caméra peut bouger indépendamment de la lumière, qui est placée au-dessus de la table. Nous pouvons utiliser les deux méthodes de calcul de l'éclairage (commenter ou

décommenter `#define USE_BLINN_PHONG` à l'intérieur du fragment shader pour passer de la méthode d'éclairage de Phong à celle de Blin-Phong).

2.5 LabWork5

Dans ce cinquième TP, l'affichage d'un lapin avec ses différentes textures a été réalisé dans la première partie (accessible de nouveau avec `#define MODEL_1`).

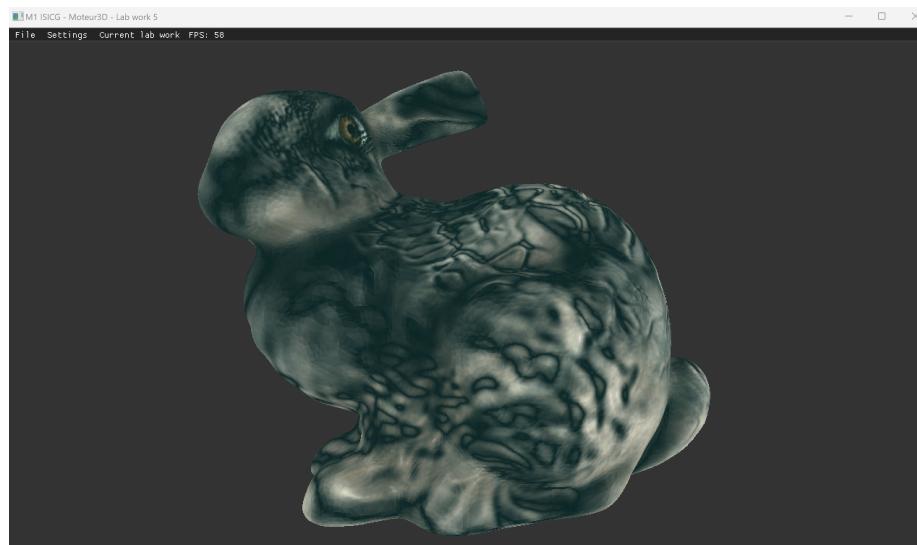


Figure 1: Démonstration des normales map sur le lapin.

Dans la seconde partie, nous affichons la scène de Sponza, en utilisant les normals map, et prenant en compte la transparence des feuilles..



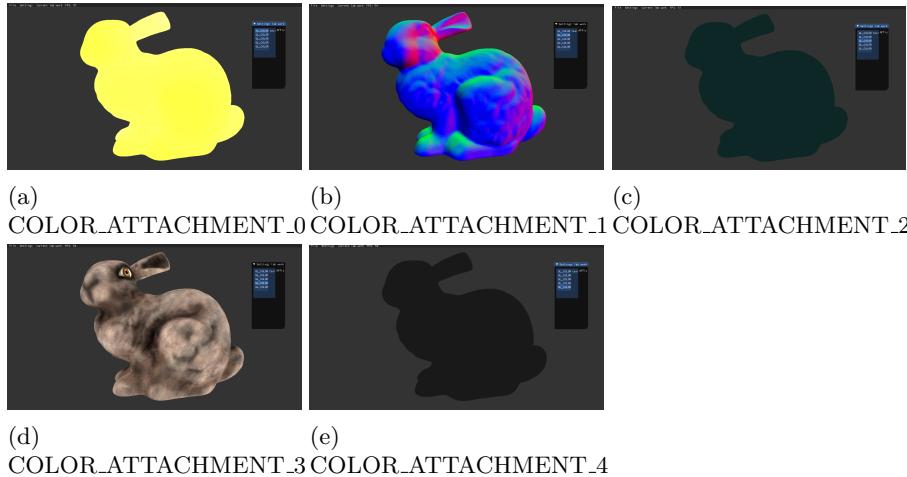
Figure 2: Démonstration de la transparence des feuilles sur la scène Sponza.

2.6 LabWork6

Le programme de l'espace de travail 6 fonctionne jusqu'au shading pass.

2.6.1 géométry pass

La geometry pass ainsi que le sélecteur de COLOR_ATTACHMENT fonctionne.
Voici les affichages des contenus du G-buffer:



2.6.2 shading pass

Le quad est bien affiché, mais le modèle du lapin n'est pas présent dessus.

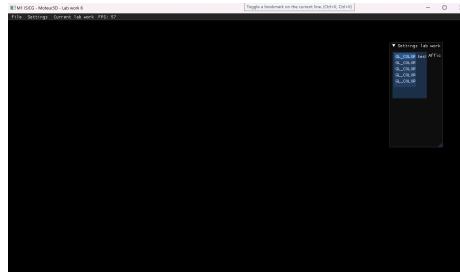


Figure 4: Affichage obtenu avec le shading pass

3 Projet final

Puisque le dernier passe du dernier TP n'est pas fonctionnel, le projet a été réalisé avec le forward rendering..

3.0.1 Lumière

J'ai d'abord commencé par mettre en place un système pour gérer, plusieurs lumières en même temps. Pour le moment on ne fait qu'additionner les couleurs entre eux. Pour plus de réalisme, comme montré dans ce tutoriel, on obtiendrait un meilleur rendu si on atténuaient la lumière selon sa distance.

Par la suite, j'ai ajouté la possibilité de changer la couleur des lumières, pour mieux remarquer l'influence des deux lumières. Dans ce même but, j'ai également animé leurs déplacements. Cela donne une ambiance un peu plus vivante à la scène.



Figure 5: Démonstration des lumières colorées.

3.0.2 La génération de texture procédurale

Pour ce projet, j'ai décidé de faire de la génération procédurale de texture animée. Tout cela est calculé dans le fragments shader.

J'ai donc dû utiliser les algorithmes de génération de bruit de Voronoi, et de Perlin. J'ai repris un algorithme de ce site pour comprendre comment les bruits dynamiques fonctionnent. Puis j'ai implémenté dans les shaders du moteur 3D cet algorithme.

J'ai également récupéré d'autres algorithmes de bruits, sur le site de shader-toy. Tel que celui qui combine bruit de Perlin et bruit de Voronoi. Il contenait une seconde version de Voronoi, avec un paramètre en plus.

J'ai donc remplacé les textures ambiantes de la scène par ces bruits. Et j'ai rendu le tout dynamique en ajoutant au menu des paramètres qui permettent de modifier en temps réel l'allure du bruit appliqué sur les objets et son intensité.

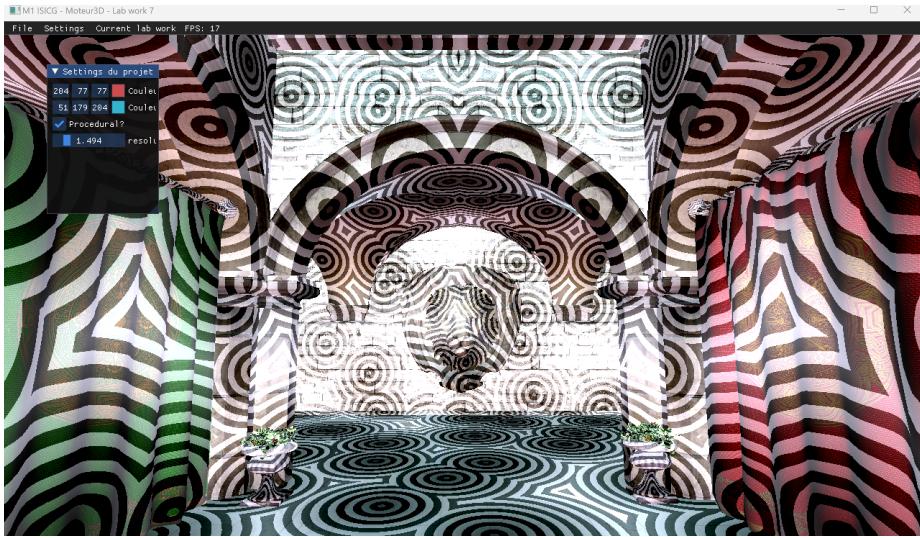


Figure 6: Démonstration de l'ajout d'un bruit sur toute les textures.

J'ai aussi remarqué que ce système ne fonctionnait pas bien sur la scène de conférence. Il n'y a qu'une seule texture affecté. C'est pourquoi j'ai changé de scène pour le Sponza de Crytek.

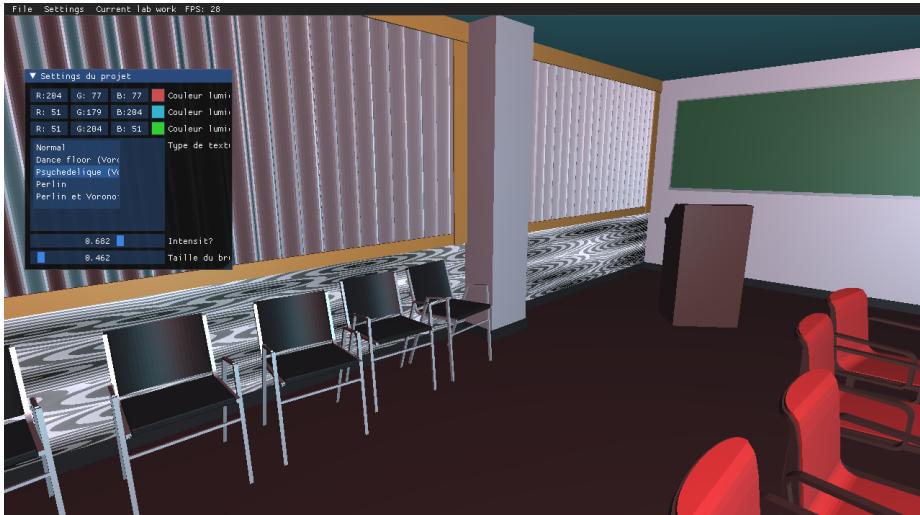


Figure 7: Bug dans salle de conférence.

3.0.3 Skybox

Par la suite, j'ai entamé la création d'une skybox. Je me suis inspiré de la fonction de chargement de texture de la classe triangle_mesh_model, tel que conseiller dans la fin du TP5. Et je me suis appuyer sur ce document. Cependant je n'ai pas encore et le temps de finir sa création et d'ajouter les buffers de vao pour son affichage.

4 Références utilisées pour le développement

- La gestion de plusieurs lumière (sans implémentation de l'atténuation selon la distance)
<https://learnopengl.com/Lighting/Multiple-lights>
- <https://learnopengl.com/Advanced-OpenGL/Cubemaps>
- Pour pouvoir passer une liste de vec3 au shader. J'ai consulté cette solution :
<https://gamedev.stackexchange.com/questions/86621/how-can-i-pass-a-stdvectorvector3f-to-a-shader>
- <https://thebookofshaders.com/12>
- Pour essayer de résoudre un bug que j'avais avec mon bruit, j'ai du tester plusieurs algorithme de bruit de ce site. <https://www.shadertoy.com>
- <https://www.shadertoy.com/view/MdGSzt>