

Programmation 3D

Compte-rendu TP4

Textures et rendu PBR

Louis Jean

Master 1 IMAGINE

Université de Montpellier

N° étudiant : 21914083

[github](#)

30 novembre 2023

Table des matières

| | | |
|---|--|----|
| 1 | Introduction | 2 |
| 2 | Plaquage d'une texture 2D sur un plan 3D | 2 |
| 3 | Normal mapping | 4 |
| 4 | Création d'une skybox | 5 |
| 5 | Environment mapping | 6 |
| 6 | Rendu PBR | 9 |
| 7 | Conclusion | 12 |

1 Introduction

Ce compte-rendu présente une exploration des techniques essentielles de programmation 3D, avec un focus sur le plaquage de textures et le rendu basé sur la physique (PBR). Nous abordons des sujets allant du simple mappage de textures 2D sur des modèles 3D à des concepts plus complexes tels que le normal mapping, la création de skybox, l'environnement mapping, en terminant par le rendu PBR. L'objectif est de se familiariser avec ces méthodes cruciales à l'informatique graphique d'aujourd'hui, contribuant à l'amélioration du réalisme et de la profondeur dans les rendus 3D.

Veuillez trouver toutes les commandes et instructions nécessaires au fonctionnement du programme dans le README.

Tous les modèles de ce TP ont été empruntés ici, dans une banque de modèles fournie par KhronosGroup.

2 Plaquage d'une texture 2D sur un plan 3D

La première étape, d'apparence toute simple, était de plaquer une texture 2D sur un maillage de plan. Une fois le maillage et la texture téléchargés et après m'être familiarisé avec le code, j'ai pu charger une texture. Au début, j'ai fait cela en brut, directement dans **Material.cpp**, de cette manière :

```
1 m_texture = loadTexture2DFromFilePath("./data/
→ TwoSidedPlane_BaseColor.png");
```

Voici le résultat obtenu.

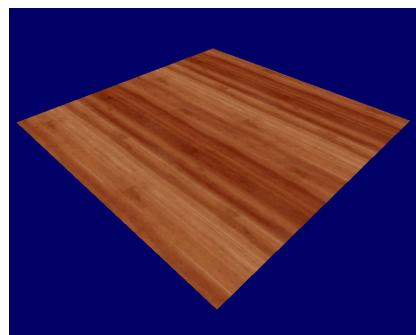
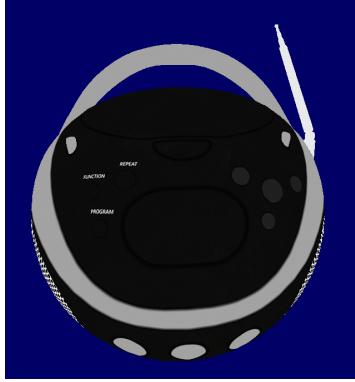
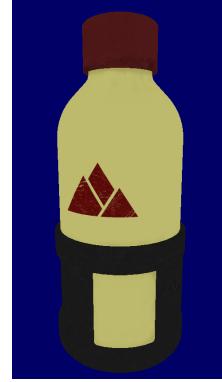


Figure 1: Texture plaquée sur un maillage de plan

J'ai aussi essayé d'autres modèles pour voir les rendus possibles et les limites d'une simple texture.



(a) Maillage d'une BoomBox avec une simple texture



(b) Maillage d'une bouteille d'eau avec une simple texture



(c) Maillage d'un barramundi avec une simple texture

Figure 2: Rendus de maillages avec une simple texture

À partir d'ici, j'ai compris que le TP allait être assez lourd à utiliser, j'ai donc voulu mettre en place un moyen rapide et pratique de changer de type de rendu sans avoir à modifier trop de lignes de code à chaque fois. Pour ce faire, j'ai inspecté le code, et me suis rendu compte que les fichiers **Context.h** et **Context.cpp** servaient à partager des informations du programme entre plusieurs fichiers. J'ai donc créé une **Enum renderingType** dans **Context.h**, qui liste chacune des méthodes de rendus explorées dans le TP, à savoir Unlit (pas d'éclairage), Phong, Reflective et PBR. En créant deux shaders par procédé de rendu et en itérant sur **Context::rendering_type**, j'ai pu m'en sortir à résumer à une seule ligne dans **main.cpp** le changement de mode de rendu.

3 Normal mapping

L'étape suivante consistait à rajouter du détail en prenant en compte une normal map. C'était la première fois que j'avais affaire à Assimp, et l'introduction fut longue. Après quelques recherches, j'ai compris que Assimp calculait les tangentes et bitangentes. Après avoir chargé la texture de normal map et en suivant la méthode du cours qui permet de transformer les normales en passant de l'espace tangent à l'espace monde, il a fallu implémenter la méthode de Blinn-Phong afin de pouvoir observer les faux reliefs créés. Pour cela, j'ai créé une lumière que j'ai passé aux shaders, et j'ai aussi passé la position de la caméra, pour ensuite faire une implémentation des plus classiques de Phong dans `phong_fragment.glsl`, et pouvoir apprécier les résultats.



(a) Maillage d'une BoomBox avec une normal map et Phong



(b) Maillage d'un bouteille d'eau avec une normal map et Phong



(c) Maillage d'un barramundi avec une normal map et Phong

Figure 3: Rendus de maillages avec une normal map et Phong

4 Creation d'une skybox

Pour faire une skybox, j'ai choisi de ne pas charger de maillage de cube, mais plutot de definir tous les sommets a la main. Mon approche consistait a charger une image par face, en utilisant **GL_TEXTURE_CUBE_MAP**. J'ai eu des difficultes a charger chaque texture, et aprs quelques recherches sur le site learnopengl.com j'ai trouv une fonction **loadSkybox** ici qui permet de faire cela. Je l'ai place dans **Texture.cpp**. J'ai donc pu terminer d'crire ma fonction **initSkybox** que j'ai place dans **main.cpp**. Il est important de noter que j'ai plac des attributs concernant la skybox dans **Context.h** et **Context.cpp**, facilitant l'utilisation de cette derniere dans la prochaine phase.



Figure 4: Rendu de la skybox

5 Environment mapping

Le but de cette partie était de faire de l'environment mapping, c'est-à-dire faire réfléchir l'environnement (à partir de la texture) sur un maillage. J'ai donc passé la texture de la skybox à **reflective_fragment.glsl**, et j'ai appliqué la logique très simple et bien expliquée sur la même page internet citée plus haut.



Figure 5: Rendu d'un maillage de BoomBox réfléchissante



Figure 6: Rendu d'un maillage de bouteille d'eau réfléchissante



Figure 7: Rendu d'un maillage de bouteille d'eau réfléchissante de près



Figure 8: Rendu d'un maillage de barramundi réfléchissant

6 Rendu PBR

Passons à la partie rendu PBR, de loin celle qui m'a le plus plu. J'ai adoré comprendre comment approcher les lois de la physique pour obtenir un rendu réaliste. J'ai fait le choix de récupérer les paramètres (ambient occlusion, roughness, metalness) directement depuis la texture disponible pour chaque modèle, en passant la texture chargée au shader, et en récupérant chaque composante selon les canaux r, g et b. Cela impliquait un petit choix de modèles, car tous les modèles ne sont pas définis de la même manière (ils n'ont pas tous un fichier regroupant les trois composantes). À l'aide de votre cours et de cette page, j'ai réussi à mettre en place mon **pbr_fragment.glsl**. J'ai cependant rencontré des problèmes quant au positionnement de la lumière dans ma scène, ne sachant pas vraiment où la mettre pour obtenir le meilleur résultat.

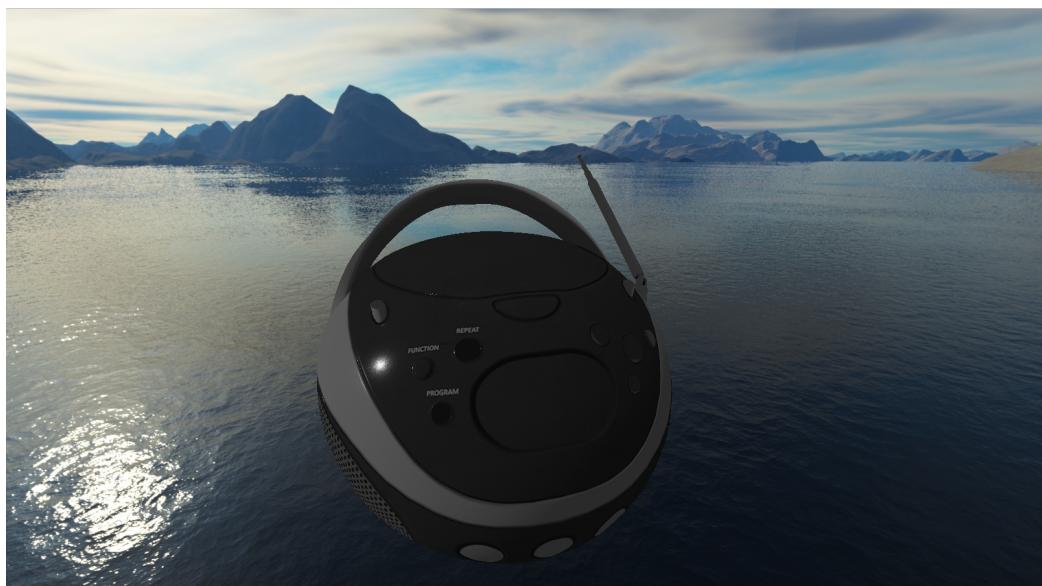


Figure 9: Rendu PBR d'un maillage de BoomBox



Figure 10: Rendu PBR d'un maillage de bouteille d'eau



Figure 11: Rendu PBR d'un maillage de corset



Figure 12: Rendu PBR d'un maillage de barramundi

7 Conclusion

Ce compte-rendu a été une exploration approfondie de la programmation 3D, couvrant du plaquage de textures au rendu PBR. Chaque étape a renforcé ma compréhension et mes compétences en informatique graphique. L'expérience acquise, en particulier dans le rendu PBR, est précieuse pour mes futures aventures dans la visualisation 3D. Mes recherches m'ont introduit au rendu réaliste, durant lesquelles j'en ai appris plus sur le subsurface scattering, et cela m'a vraiment intéressé, j'aimerai le mettre en place dans un futur proche. Merci pour le temps et l'attention que vous avez consacrés à la lecture de ce compte-rendu.