

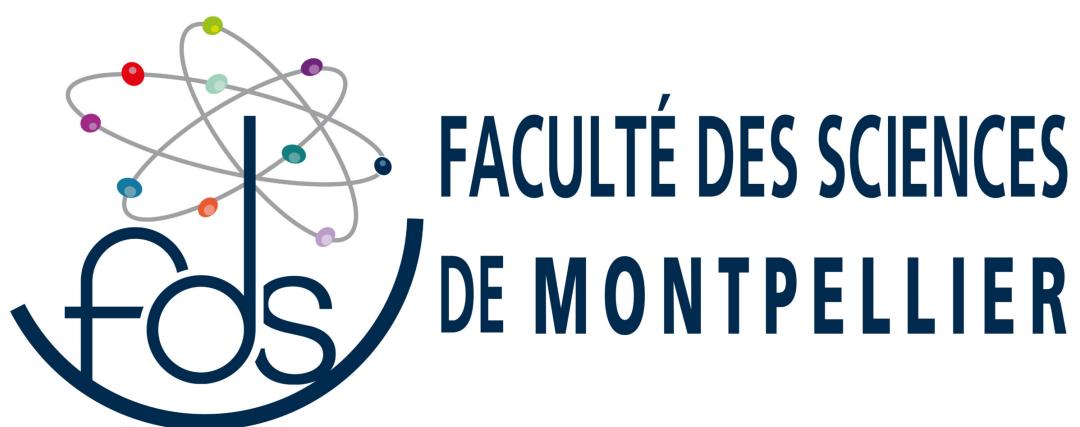
UNIVERSITÉ DE MONTPELLIER

M1 - Programmation 3D
Compte Rendu - Texture

Etudiant :
Guillaume Bataille
Numéro étudiant : 21916619

Encadrant :
BEUGNON SEBASTIEN

Année 2022-2023



0.1 Contexte et Objectif

Nous avons eu un cours sur les textures et leurs applications sous diverses formes.

L'objectif de ce tp est donc de se familiariser avec le texturing.

Sommaire

0.1	Contexte et Objectif	1
1	Workflow	3
2	Gestion de l'éclairage via phong	4
3	Texture	6
4	NormalMap	7
5	SkyBox	8
6	Reflexion	9
7	PBR	10
8	Conclusion et difficulté	12

1. Workflow

Git et Github m'ont servi comme dépôt en ligne pour pouvoir avancer sur différentes machines et d'avoir un historique de progression.

VSCode est l'IDE que j'ai utilisé pour ce tp

learnopengl.com m'a servi d'aide et de guide complémentaire pour effectuer le tp Le code source provient du git fourni par Mr Beugnon et la compilation/build c'est faite via cmake

2. Gestion de l'éclairage via phong

En effet dans le code de base nous avons un visuel comme celui-ci : Nous constatons qu'il n'y a pas

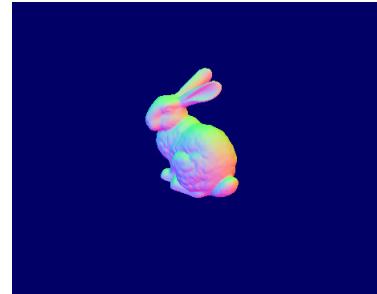


FIGURE 2.1 – Bunny.obj de base

une gestion de l'éclairage, donc il va nous falloir implémenter phong avec ses composantes ambiante speculaire et diffuse. Cela nous servira par la suite pour des textures avec du relief.

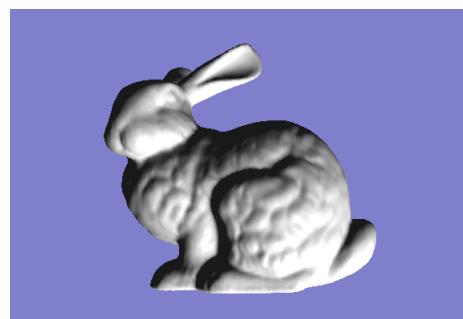


FIGURE 2.2 – Bunny effrayant avec la light en dessous

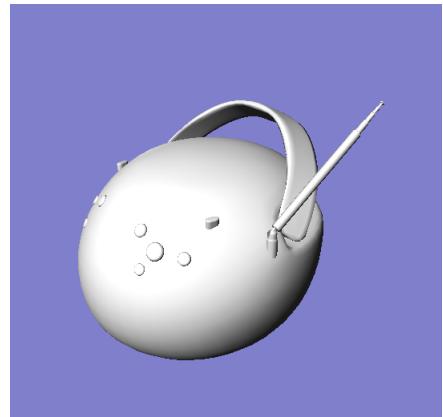


FIGURE 2.3 – Phong 1

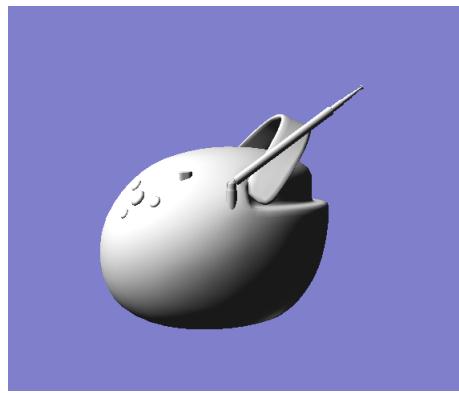


FIGURE 2.4 – Phong 2

3. Texture

Afin de pouvoir plaquer des textures, j'ai suivi les tutos sur learnopengl. J'ai donc du envoyer le nécessaire aux shaders (comme la texture ou les normales etc..) puis dans le fragment shader on a utilisé les uv afin de chercher dans l'image des textures quelle couleur on applique.



FIGURE 3.1 – Texture appliquée sur la Boombox

4. NormalMap

Maintenant qu'on a de quoi appliquer une texture et de quoi avoir une lumière intéressante (phong), on cherche à utiliser des normalmap. Le but ici va être d'envoyer une texture comme la précédent, sauf qu'on va faire varier les normales du mesh en fonction de cette normalmap.

Cela aura pour effet de donner du relief (plus ou moins précis) sur un mesh à partir d'une texture 2D noir et blanc. Pour ce faire il va falloir manipuler les normales et les réorienter selon la map. Cela est rendu possible avec une matrice de passage TBN (pour tangente,bitangente et normale) ce qui permet de projeter les coordonnées sur l'espace des normales souhaité.

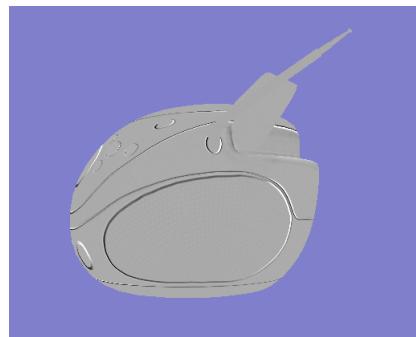


FIGURE 4.1 – Application de la normalmap sur la boombox



FIGURE 4.2 – Application de la normalmap + texture sur la boombox

On peut voir qu'il y a un gain de précision via la normalmap, d'autant plus que la lumière, réagissant aux normales s'adapte à ses changements de relief pour donner plus de détail (bouton, partie différentes de l'appareil, relief sur les cotés, etc..)

5. SkyBox

Une partie compliquée qui m'a prise du temps, la skybox est une boîte contenant les textures d'un ciel (6 textures) et dès qu'on est à l'intérieur, on a l'illusion d'être entouré d'un ciel. Pour pouvoir l'effectuer, il faut faire une boîte englobante contenant la scène courante (camera + scène) et avec ce que nous avons appris pour le projet de raytracing, nous avons pu construire à la main (même si c'est pas parfait) la boîte englobante.

Par la suite j'ai suivi un tuto expliquant comment charger les coordonnées du cube via des buffers, charger la texture des cube et l'envoyer côté fragment shader.

Ensuite, on n'affiche la skybox uniquement si la ou on regarde il n'y a pas d'objets/meshes.



FIGURE 5.1 – Visuel sans mesh de la skybox

6. Reflexion

Ensuite, le tp nous demande d'effectuer un objet 3D réfléchissant. C'est quelque chose que l'on a déjà vu en raytracing. Il faut compute le vecteur camera, et lorsque ce vecteur touche une surface, on utilise la normale pour trouver le rayon réfléchi.



FIGURE 6.1 – Reflexion sur la boombox



FIGURE 6.2 – Reflexion sur bunny

7. PBR

Pour le PBR (physic based rendering), j'avoue avoir eu beaucoup de mal à comprendre comment m'y prendre. J'ai donc suivi un tutoriel expliquant comment s'y prendre. J'ai essayé de comprendre comment les différentes fonctions comme celle de Schlick GGX fonctionne mais la seule chose que j'ai retenu c'est que le PBR se base sur la rugosité. Plus on est rugueux, plus la luminosité renvoyé sera atténue et large. A l'inverse, moins on l'est et plus la luminosité renvoyé sera concentré et intense



FIGURE 7.1 – Bunny rugueux = 0.8



FIGURE 7.2 – Bunny brillant = 0.2



FIGURE 7.3 – Metallique rugueux



FIGURE 7.4 – Metallique brillant

8. Conclusion et difficulté

Pour conclure, l'utilisation de texture peut s'avérer très utile de part son implémentation peut couteuse et l'effet "beau" qu'il peut rajouter.

Concernant les difficultés, j'ai trouvé ce tp particulièrement difficile. J'ai passé ne nombreuses heures avec des collègues de classe pour pouvoir résoudre plusieurs soucis qu'on a rencontré. C'est peut être parce que la programmation graphique n'est pas encore aquise de mon côté..
