

[Prog3D] TP sur les textures et rendu PBR



Adèle Imparato (22200570)

23 décembre 2022



1 Introduction

Le but de ce TP est de comprendre comment obtenir un rendu réaliste sur base d'une scène ou d'un objet 3D. Pour ce faire, ce TP se décompose en 2 sections, elles-mêmes découpées en 3 niveaux chacune.

La première section nous apprend à plaquer une texture sur un objet tout en rajoutant de la rugosité à celui-ci à l'aide d'une bump map (ou alternativement, une normal map). Enfin, elle nous apprend à manipuler la skybox, technique permettant elle aussi de texturer un objet mais cette fois en utilisant un cube 3D, soit 6 textures 2D reliées.

La deuxième section traite de rendu PBR. Dans celle-ci, on apprend à développer des nouveaux Shaders, rajouter un effet d'émission et d'ambiant occlusion, et pour finir, d'utiliser un modèle PBR connu et d'en développer les fonctionnalités.

Je vais à présent décomposer ce rapport en six sections, en expliquant pour chacune ce que j'ai réussi à réaliser, et ce qui n'a pas été. SPOILER : Ca n'a pas été.

2 Textures

2.1 Niveau 0 : Plaquage d'une texture 2D sur un plan 3D

Ce tout premier niveau consiste à plaquer une texture (donc une image en 2D) sur notre objet en 3D. J'ai deux codes différents pour ce niveau : "HAI719I_TP2_copie-niveau0" qui reprend le code du TP2, et "template-niv0" qui reprend le template donné en cours.

Dans la première version, on crée un buffer contenant les données (précédemment initialisées) de la texture qu'on va ensuite "bind" (lier) à une target de type GL_TEXTURE_2D. Ensuite, on charge et on génère la texture à l'aide de notre image texture, ici brique.png. Afin de s'assurer que celle-ci réagit de manière adaptée lorsqu'elle est étirée ou réduite, on définit certains de ses comportements à l'aide de la fonction glTexParameter(). On s'attaque ensuite au fragment et au vertex shader. Dans le fragment shader, il nous faut une variable uniforme (sampler2D) pour la



texture et un vecteur 2D (in) pour les coordonnées texture. Tandis que dans le vertex shader, il nous faut ajouter un vecteur 2D (out) pour le coordonnées texture (du même nom que le in dans le fragment shader).

Voici le résultat obtenu une fois la texture plaquée sur un simple triangle :

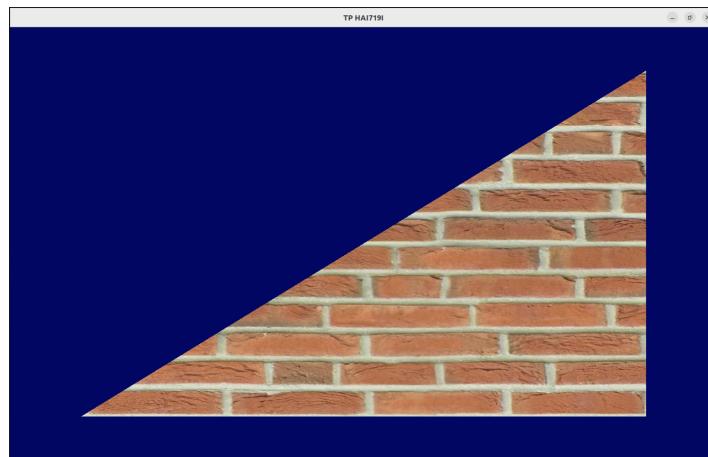


FIGURE 1 – Texture brique.png plaquée sur un objet triangulaire simple.

J'ai obtenu un résultat similaire en manipulant le code template :

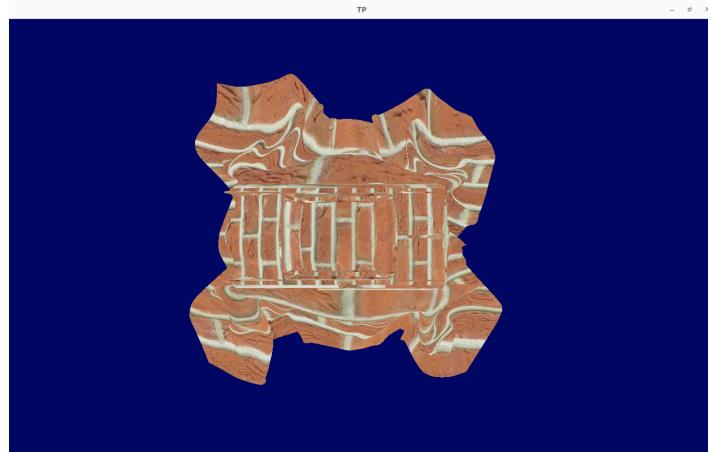


FIGURE 2 – Texture brique.png plaquée sur l'objet ToyCar.

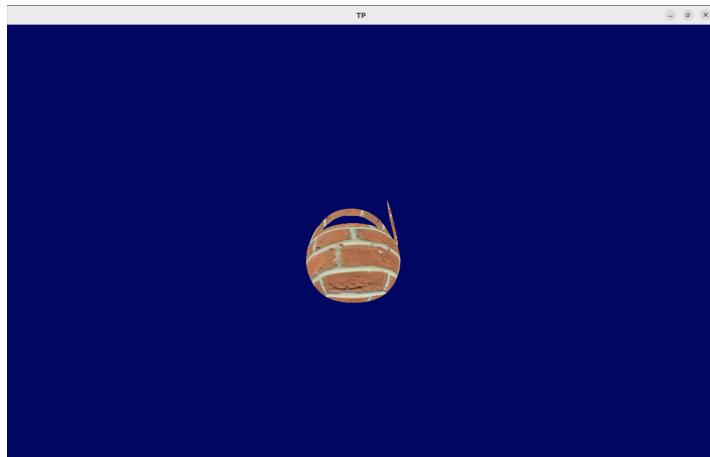


FIGURE 3 – Texture brique.png plaquée sur l'objet BoomBox.

2.2 Niveau 1 : Normal Mapping

Le but de ce deuxième niveau est de rajouter de la rugosité à notre texture, pour la rendre encore plus réaliste. Pour comprendre, cette section, j'ai suivi les explications données à cette source : <https://learnopengl.com/Advanced-Lighting/Normal-Mapping>. J'ai d'abord commencé par tenter de générer une bump map, sans succès, puis ensuite je me suis focalisée sur la normale map. L'idée de celle-ci est la suivante : afin de donner un effet de relief à notre texture, on utilise une seconde texture correspond à la normale map. Celle-ci est similaire à l'image texture à part que ses couleurs sont différentes. Cela est du au fait que chaque couleur code en réalité un vecteur normal. Le concept derrière la normale map est donc de modifier les normales pour chaque fragment plutôt que de considérer une seule normale commune. Ces normales vont donc être désorientées par rapport à la normale initiale et influencer la lumière en conséquence (note : il faut donc rajouter un modèle de lumière à notre code, comme par exemple Phong). C'est donc ça qui va nous donner un effet de relief, bien qu'en réalité l'objet initial peut avoir une surface plane. Malheureusement, j'ai beau avoir compris l'idée derrière la normale map, je ne suis pas parvenue à l'intégrer dans mon code, je n'ai donc aucun résultat à montrer.

2.3 Niveau 2 : Environnement Mapping

TODO



3 Rendu PBR

3.1 Niveau 0

TODO

3.2 Niveau 1

TODO

3.3 Niveau 2

TODO

4 Organisation

Vous vous demandez certainement comment j'obtiens si peu de résultats pour un si gros travail. Ce qu'il s'est passé c'est que j'ai été très démotivée par ce travail car dès le premier TP je ne suis parvenue à rien. Il m'a fallu environ 8 heures de travail pour obtenir une texture simple. Sachant que cette étape ne consistait qu'1/6 du travail final, je me demandais sérieusement comment j'allais arriver au bout de ce TP.

Par la suite, à chaque fois que je me replongeais dans ce travail, je passais plusieurs heures à coincer et à ne pas avancer. Les autres de la classe semblaient aussi perdus que moi. Etant donné les autres cours que nous avons en parallèle, et le gros projet de raytracing en Progammation 3D, j'ai préféré me focaliser sur autre chose, ayant peur de perdre des journées entières sur un seul TP (comme ça a été le cas pour certains de mes camarades). Je sais que ce n'est pas le choix le plus intelligent, étant donné que c'est de la matière utile pour la suite, mais voilà je n'avais plus le coeur à m'y mettre après avoir perdu autant de temps sur des tâches qui n'ont pas abouties.

5 Ce que j'ai appris

OpenGL requiert une compréhension bien plus profonde des concepts et méthodes clés, et que la théorie ne suffit pas pour manipuler le code. J'ai aussi appris à utiliser github pour rendre un devoir, je trouve ça très utile et je n'avais jamais du faire ça auparavant.



6 Feedback

Je pense sincèrement que ce travail devrait être raccourci (ou valoir une note finale d'environ 20%). Il est vrai que dans mon cas, c'est le manque de motivation qui a fait que j'ai obtenu si peu de résultats, mais je pense que beaucoup d'autres élèves ont eu énormément de mal à réaliser tous ces niveaux, et pris du retard sur le reste de leur étude pour cela. Néanmoins, j'ai apprécié les cours que vous nous avez donnés et l'aide fournie durant les TPs.