



Master Informatique, Synthèse d'Images et Conception Graphique

Rapport de Projet Moteur 3D

KADRI MAROUEN

31 MAI 2022

Table des figures

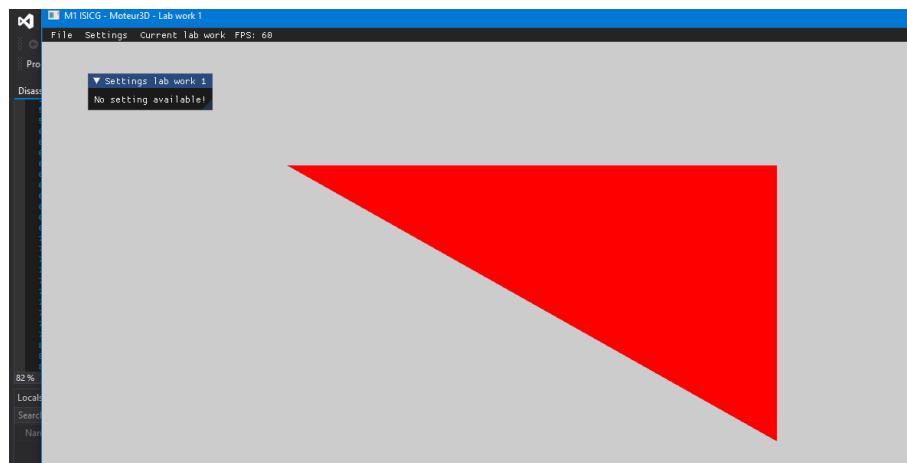
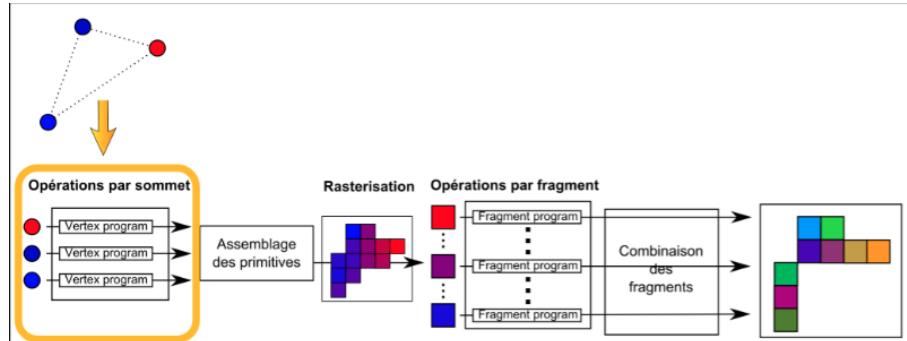
1	Quad	2
2	Cube	3
3	Bunny (a) Phong lighting model (b) Blinn-Phong lighting model (c) Gaussian lighting model	4
4	Conférence (a) Phong lighting model (b) Blinn-Phong lighting model (c) Gaussian lighting model	4
5	src learnOpengl.com	4
6	(a) avec Normal-Map (b) sans Normal-Map	5
7	(a) Transparance (b) Mipmaps & filtrage bilinéaire	5
8	Sponza (a) Phong lighting model (b) Blinn-Phong lighting model (c) Gaussian lighting model	5

Table des matières

1	TP1	2
2	TP2	2
3	TP3	3
4	TP4	3
5	TP5	4
6	Conclusion	6
7	Bibliographie	6

1 TP1

le but de ce tp est de se familiariser avec OpenGL en utilisant les buffers et shaders . Nous avons entré une liste de trois coordonnées 3D qui sont traité par le pipeline graphique pour afficher un triangle rouge dont la couleur est fixé dans le fragment shager.



2 TP2

Nous avons affiché un **Quad** à la base de deux triangle normalement basé sur 6 vertex mais avec l'aide des **EBO** nous avons fait le travail avec 4 vertex , on résolue le problème de redondance des données. pour l'animation nous avons utilisé la variable **uTranslationX** pour glisser le quad horizontalement. nous avons modifié la luminosité à travers l'interface **imgui** on utilisant une variable uniforme.

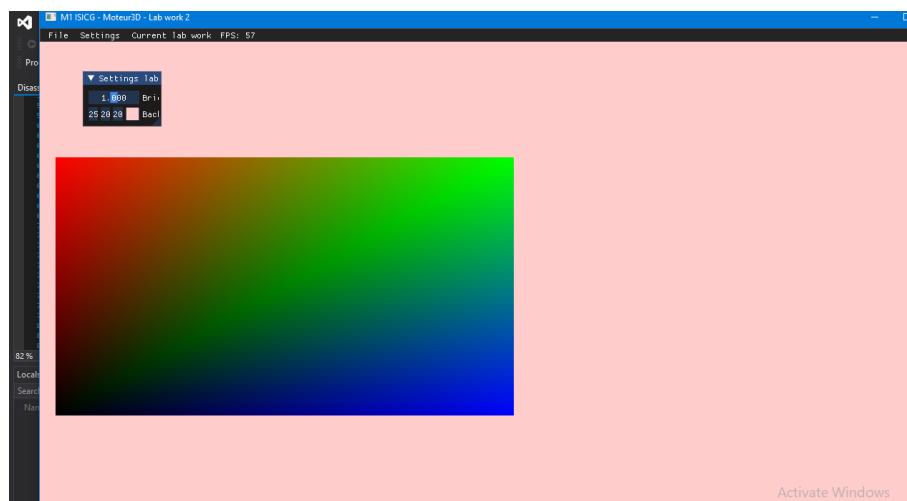


FIGURE 1 – Quad

3 TP3

Dans ce TP nous avons ajouté un repère pour la caméra et à partir de laquelle on calcule les matrices de projection.

- Le calcul de la matrice MVP se fait sur CPU et est envoyé au shaders.
- la réalisation de la rotation de cube se fait à travers les fonctions trigonométriques.
- **Toggles Button** sont ajouté à l'interface graphique pour appliquer une animation sur notre cube : par défaut la rotation de cube est sélectionnée et on a le choix de faire la rotation du caméra .

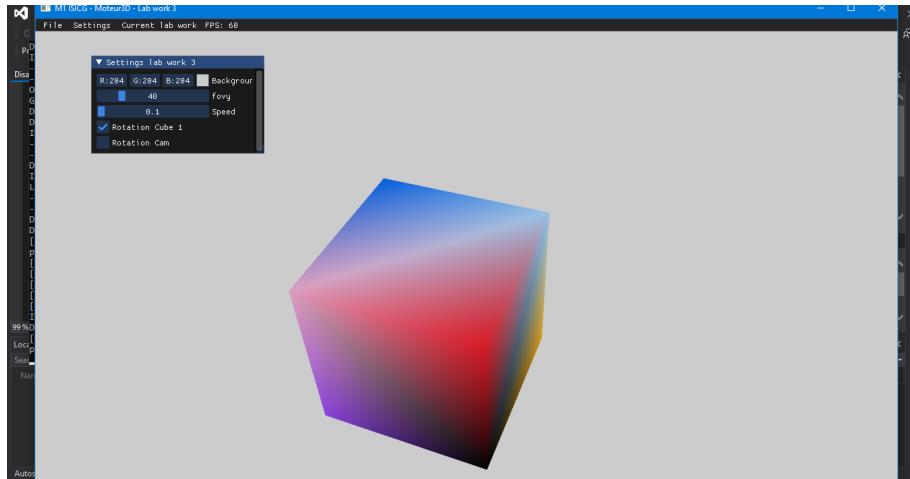


FIGURE 2 – Cube

4 TP4

Dans ce Tp nous avons abordé les différentes modèles d'éclairage :phong ,blinn-phong , Gaussian

- Nous avons passé la couleur de la lumière et la position de la caméra directement du CPU vers les shaders.
- le calcul du modèle d'éclairage se fait dans la fragment shader
- on peut passer directement d'un modèle d'éclairage vers un autre à l'aide des Toggles Button : par défaut le modèle blinn-phong est appliqué

Il faut fixer les paramètres suivantes de la caméra pour :

- le rendu de la scène

```
_camera.setPosition( Vec3f( 12, 12, 12 ) );
_camera.setLookAt( Vec3f( 7, 4, 10 ) );
//-----
```

- affichage du Bunny

```
' _camera.setPosition( Vec3f( 0, 0, 3 ) );
'_camera.setLookAt( Vec3f( 0, 0, 1 ) );
```

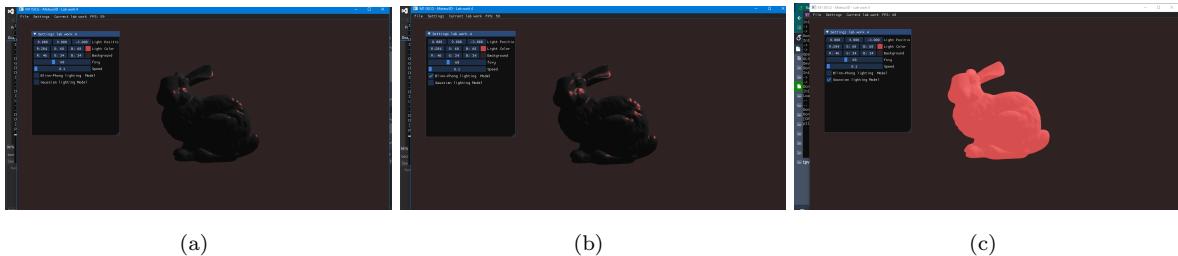


FIGURE 3 – Bunny (a) Phong lighting model (b) Blinn-Phong lighting model (c) Gaussian lighting model

Après le remplacement du lapin par la modélisé Conférence , nous avons obtenu ce résultat :

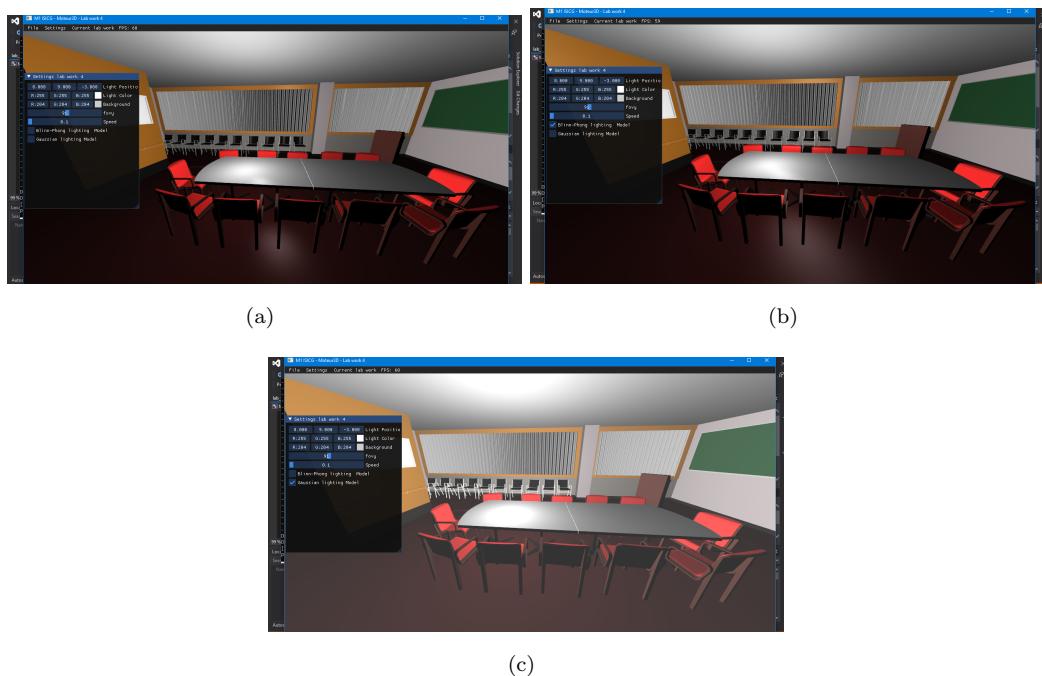


FIGURE 4 – Conférence (a) Phong lighting model (b) Blinn-Phong lighting model (c) Gaussian lighting model

5 TP5

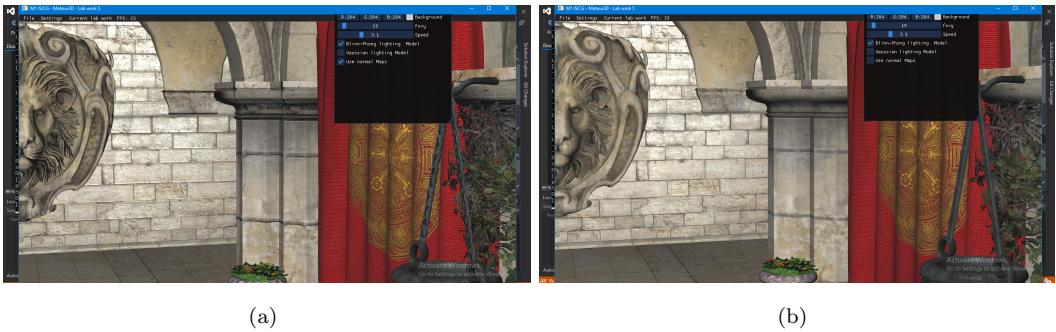
Dans ce Tp nous avons abordé les textures en gardant la même interface graphique (ImGui)

- Application des normales (normal mapping) est de donner aux normales des variations similaires ce qui rendre notre objet plus réel. selon Learnopengl.com nous avons appliqué la premier méthode puisque j'ai la trouvé assez simple .

So now that we have a TBN matrix, how are we going to use it? There are two ways we can use a TBN matrix for normal mapping, and we'll demonstrate both of them:

1. We take the TBN matrix that transforms any vector from tangent to world space, give it to the fragment shader, and transform the sampled normal from tangent space to world space using the TBN matrix; the normal is then in the same space as the other lighting variables.
2. We take the inverse of the TBN matrix that transforms any vector from world space to tangent space, and use this matrix to transform not the normal, but the other relevant lighting variables to tangent space; the normal is then again in the same space as the other lighting variables.

FIGURE 5 – src learnOpengl.com

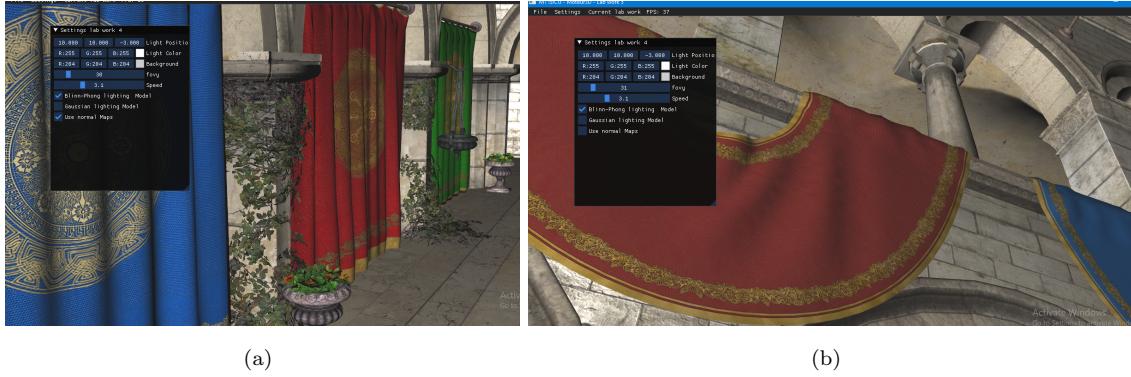


(a)

(b)

FIGURE 6 – (a) avec Normal-Map (b) sans Normal-Map

— La transparence et les mipmaps :



(a)

(b)

FIGURE 7 – (a) Transparance (b) Mipmaps & filtrage bilinéaire

— Applications des différentes modèles d'éclairage :

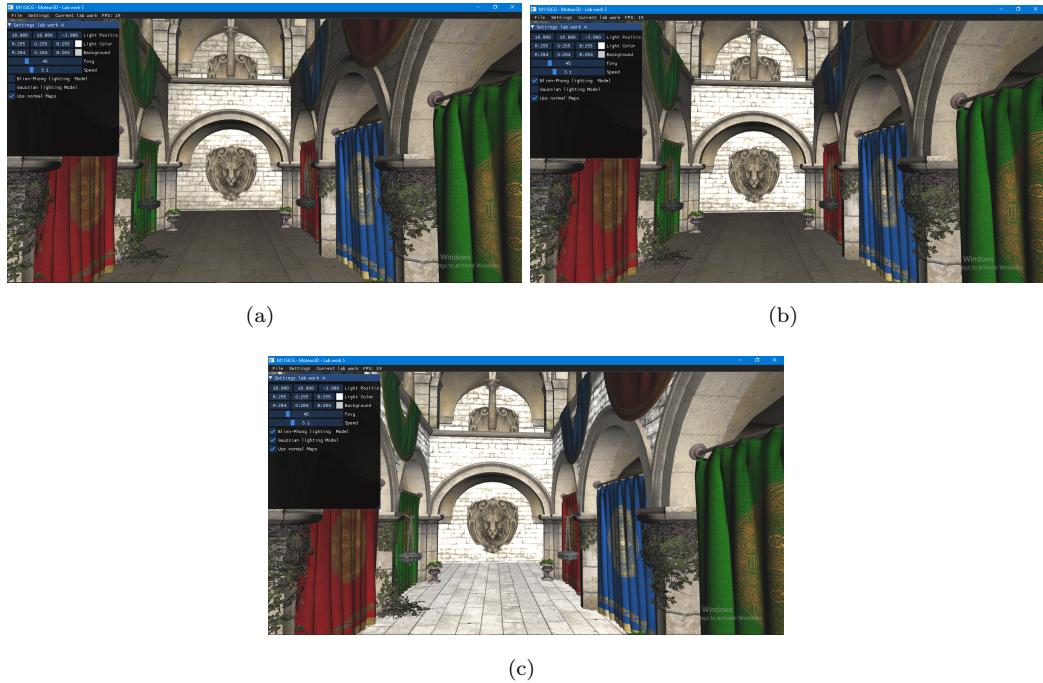


FIGURE 8 – Sponza (a) Phong lighting model (b) Blinn-Phong lighting model (c) Gaussian lighting model

6 Conclusion

Dans l'ensemble, nous pensons que le projet s'est bien passé nous avons fait des améliorations par rapport au semestre 1. Nous avons réussi à implémenter ce que nous avons vu pendant les séances des Tp . Avec plus de temps nous aurions pu l'améliorer, j'ai trouvé des idées sympas sur ce site :<https://lettier.github.io/3d-game-shaders-for-beginners/setup.html> comm FOG Cel Shading, Rim et Lighting Gamma Correction etc..J'essaierai de l'appliquer pendant la vacance .

7 Bibliographie

Ci-dessous une collection de références vers les différentes ressources m'ayant servi lors de la réalisation de ce projet

Références

- [1] *Shader Intermediates - Normal Mapping*
<https://shader-tutorial.dev/intermediates/normal-mapping/>
- [2] *GLSL Tutorial – Uniform Variables*
<https://www.lighthouse3d.com/tutorials/glsl-tutorial/uniform-variables/>
- [3] *Textures*
<https://learnopengl.com/Getting-started/Textures/>
- [4] *blinn-phong-shading*
<https://garykeen27.wixsite.com/portfolio/blinn-phong-shading/>