Rapport Moteur3D

Amadou DIARRA

1 Introduction

L'objectif de ce projet était de saisir le fonctionnement d'OpenGL. Pour atteindre cette compréhension, il a été nécessaire d'approfondir la réalisation des travaux pratiques.

2 TP1

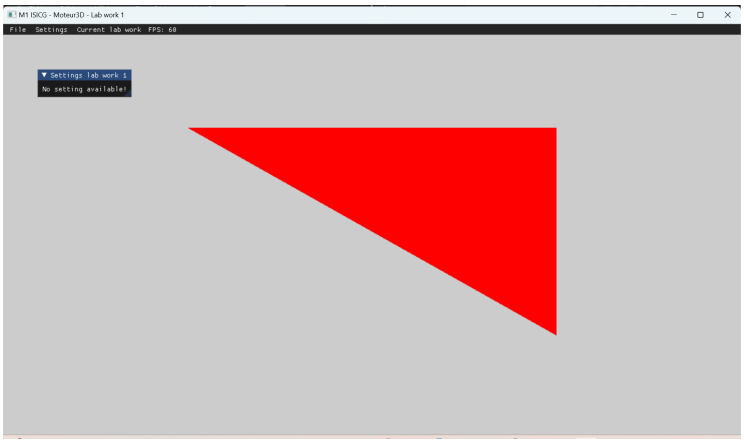
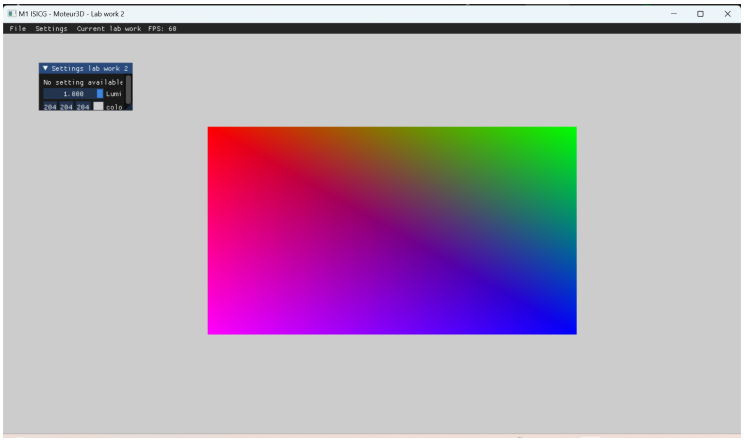


FIGURE 1 – Resultat TP1

L'objectif de ce travail pratique était de s'initier à OpenGL. Il consistait à comprendre comment utiliser un VBO et un VAO, ainsi que la liaison avec le fragment shader et le vertex. Une fois ces concepts compris en suivant les explications, afficher un triangle devint une tâche simple.

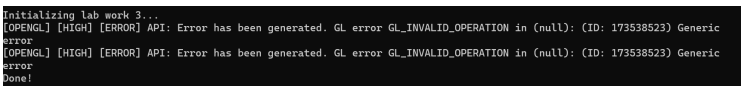
3 TP2



L'objectif de ce travail pratique consistait à améliorer le programme du TP1 en passant d'un simple triangle à un rectangle en mouvement. Pour ce faire, il a été nécessaire d'ajouter davantage de points dans le VBO (Vertex Buffer Object) et d'introduire un EBO (Element Buffer Object). L'utilisation de l'EBO permettait de spécifier les indices des points pour définir les triangles. Par exemple, en utilisant les indices 1, 2, 3, 4, 2, 0, on pouvait tracer un triangle avec les points 1, 2, 3, et un autre avec les points 4, 2, 0.

Une autre étape importante était de comprendre le système d'uniforme, qui consiste à envoyer des informations aux shaders, en l'occurrence aux vertex shaders dans ce contexte. Cela était nécessaire pour effectuer une translation, c'est-à-dire déplacer le rectangle. En résumé, l'objectif était d'enrichir le programme en ajoutant des fonctionnalités telles que la gestion d'un rectangle mobile, l'utilisation d'un EBO pour définir les triangles, et la transmission d'informations via des uniformes pour réaliser des transformations comme la translation.

4 TP3

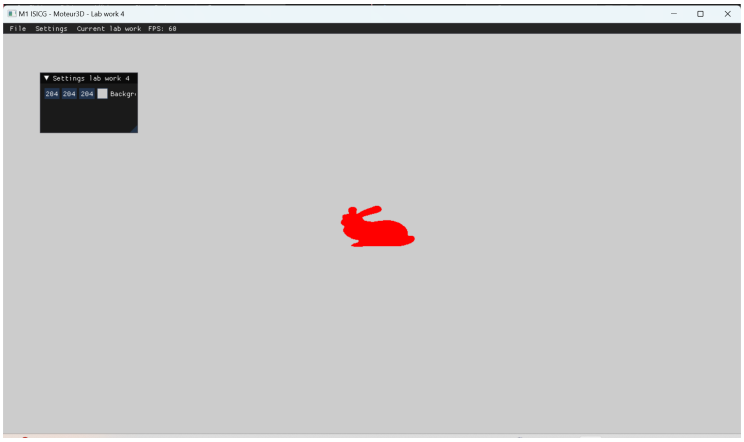


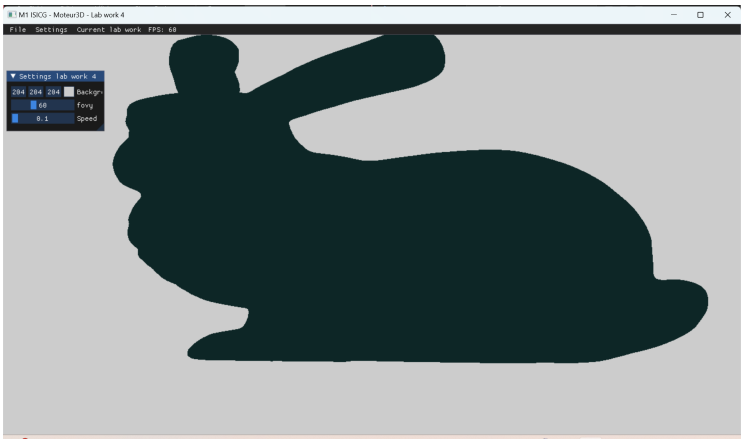
Le TP3 constituait une progression naturelle à partir du TP2, cette fois-ci introduisant la dimension 3D. L'objectif était d'afficher un cube effectuant une rotation en fonction du temps. Avec le passage à la 3D, une tâche cruciale consistait à finaliser l'implémentation d'une classe caméra, en assurant la complétion pour permettre la récupération des différentes matrices nécessaires, telles que la View Matrix et la Projection Matrix.

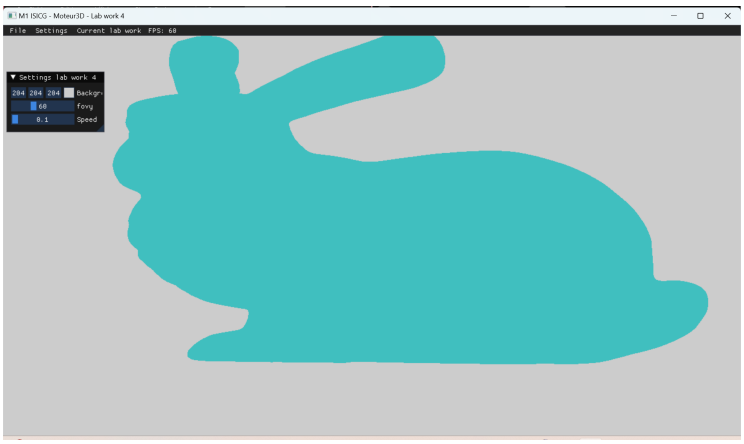
La phase suivante impliquait la représentation du cube à l'aide de l'EBO, VBO, et VAO. Pour mieux appréhender le problème et identifier les triangles du cube à afficher en se basant sur les indices de sommets. Une fois le cube intégré dans le

code, l'étape cruciale était d'envoyer les trois matrices (Modèle/Projection/Vue) à nos shaders.

5 TP4







Dans un premier temps, il était nécessaire de charger le modèle, ce qui impliquait de compléter une classe avec les VBO, VAO, et EBO correspondants. Cependant, cette fois, nous disposions d'un seul VBO pour 5 attributs. Une fois les offsets correctement configurés, nous parvenions à afficher un lapin. La prochaine étape consistait à ajouter progressivement des lumières pour calculer la couleur.