ORAN Mikail
PHAM Vincent

Rapport SI3D

TP1

Partie 1

Quelles remarques peut-on faire sur le temps cpu et le temps gpu en fonction du nombre d'objets affichés (faites plusieurs essais en modifiant le nombre d'objets affichés, ainsi que le nombre de lignes et de colonnes de la grille) ?

Le temps CPU augmente exponentiellement par rapport au GPU. Ceci doit être dû à toutes les transformations à calculer pour dessiner les cubes côte à côte. Voici un petit tableau récapitulatif.

Nombre d'éléments	CPU (ms)	GPU (ms)
1 x 1	0.05	0.1
10 x 10	1.2	0.6
100 x 100	40	2.5
1000 x 100	4000	230

Est ce que le temps d'affichage varie selon la position de la caméra ? et le temps cpu ? gpu ?

Le temps d'affichage ne varie pas selon la position de la caméra car les objets qui sont en dehors du champ de vision sont quand même calculés et dessinés, même si on ne les voit pas.

Partie 2

Pour cette partie on cherche à dessiner des objets en utilisant uniquement des fonctions OpenGL.
Afin d'y arriver nous devons configurer les attributs de sommets (VAO) avec les buffers puis dessiner le tout grâce à glDrawArrays() ou glDrawElements().

Partie 3

Pour passer un vecteur de translation au shader, on peut simplement utiliser un uniform.
Pour ce faire, nous avons besoin de le préciser dans l'application grâce à la fonction glUniform4fv() tout en ayant trouvé sa position au préalable avec la fonction glGetUniformLocation().

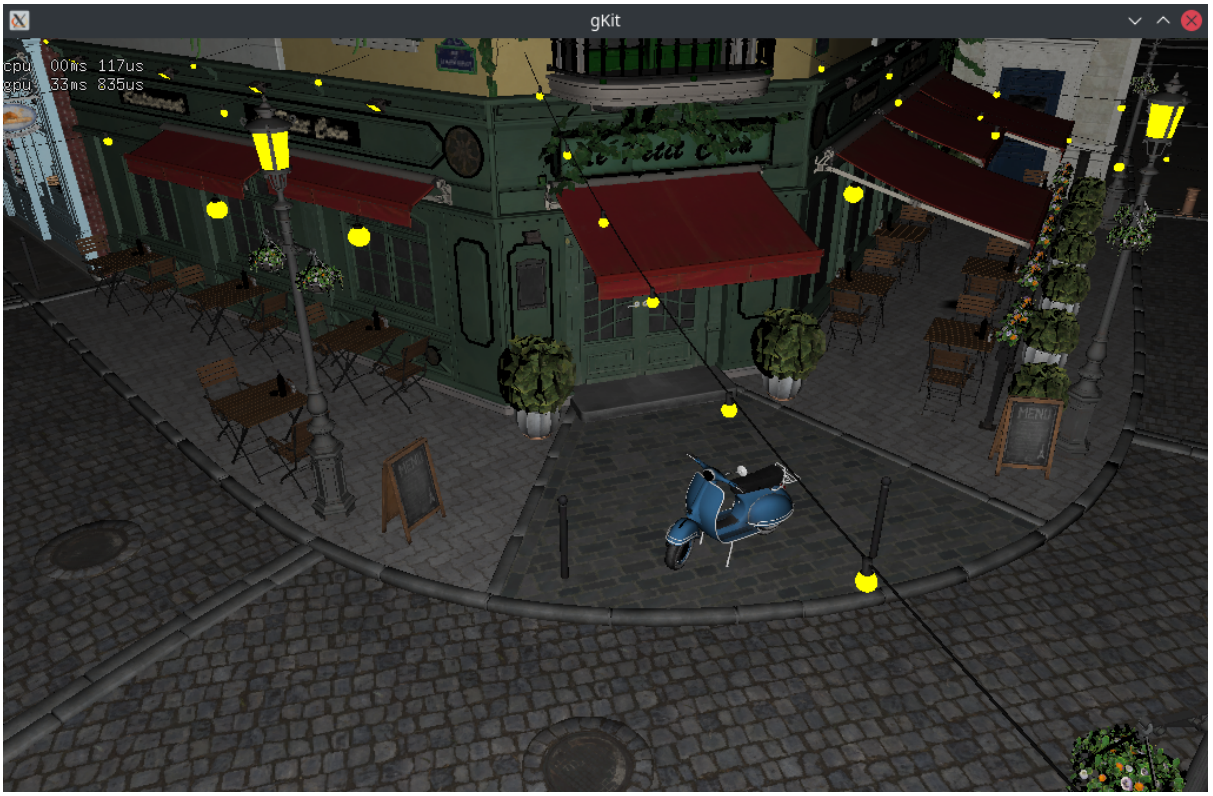
Partie 4

Dans cette partie on cherche à afficher des objets avec leur matière mais sans faire de draw superflux. L'objectif ici est de faire en sorte que le fragment shader puisse décider de la couleur d'un fragment d'un triangle selon sa matière.
Pour cela nous devons passer au shader un tableau uniform qui contient toutes les matières nécessaires à dessiner l'objet.
L'étape suivante est de pouvoir savoir quelle est l'indice de la matière d'un triangle dans l'uniform de matières. Il faut donc pour cela que nous triions les triangles de l'objet à afficher selon leur matière avant de créer et configurer le VAO. Comme attribut de sommet supplémentaire nous passons un tableau qui, pour chaque triangle, indique l'indice de sa matière dans le tableau de matière défini précédemment.
Il ne nous reste plus qu'à combiner ces informations dans le shader pour avoir la matière correcte. Comme l'attribut d'indice n'est disponible uniquement dans le vertex shader, on utilise un varying pour passer l'information au fragment shader. On n'oublie pas de préciser le mot-clé *flat* pour indiquer que la valeur ne doit pas être interpolée.
Dans le fragment shader on utilise l'indice reçu du vertex shader pour accéder à la valeur correcte dans l'uniform des matières.

Partie 5

Pour le modèle de Lambert, ainsi que Blinn-Phong, il est nécessaire de calculer certaines informations. Nous disposons d'emblée de certaines informations telles que la normale à la surface ainsi que la position de l'observateur (l'opposé du vecteur de position dans le repère caméra). Pour Blinn-Phong, nous avons besoin de connaître le vecteur intermédiaire entre l'observateur et la direction de la lumière.

Résultat du TP1



TP2

Partie 1

Comment déterminer qu'un point est dans le frustum ? ou à l'extérieur ?

Dans le repère projectif, on vérifie que les composantes x, y et z du point ne sont pas inférieures à -w ou supérieures a w chacunes.

Quelles sont les coordonnées des sommets du frustum dans le repère projectif ? et dans le repère de la scène ? faut-il utiliser des transformations particulières ?

Dans le repère projectif, le frustum est une boîte ayant pour extrémités (-1,-1,-1) et (1,1,1). Pour trouver ses coordonnées dans le repère de la scène il faut appliquer les transformations inverses.

Partie 2

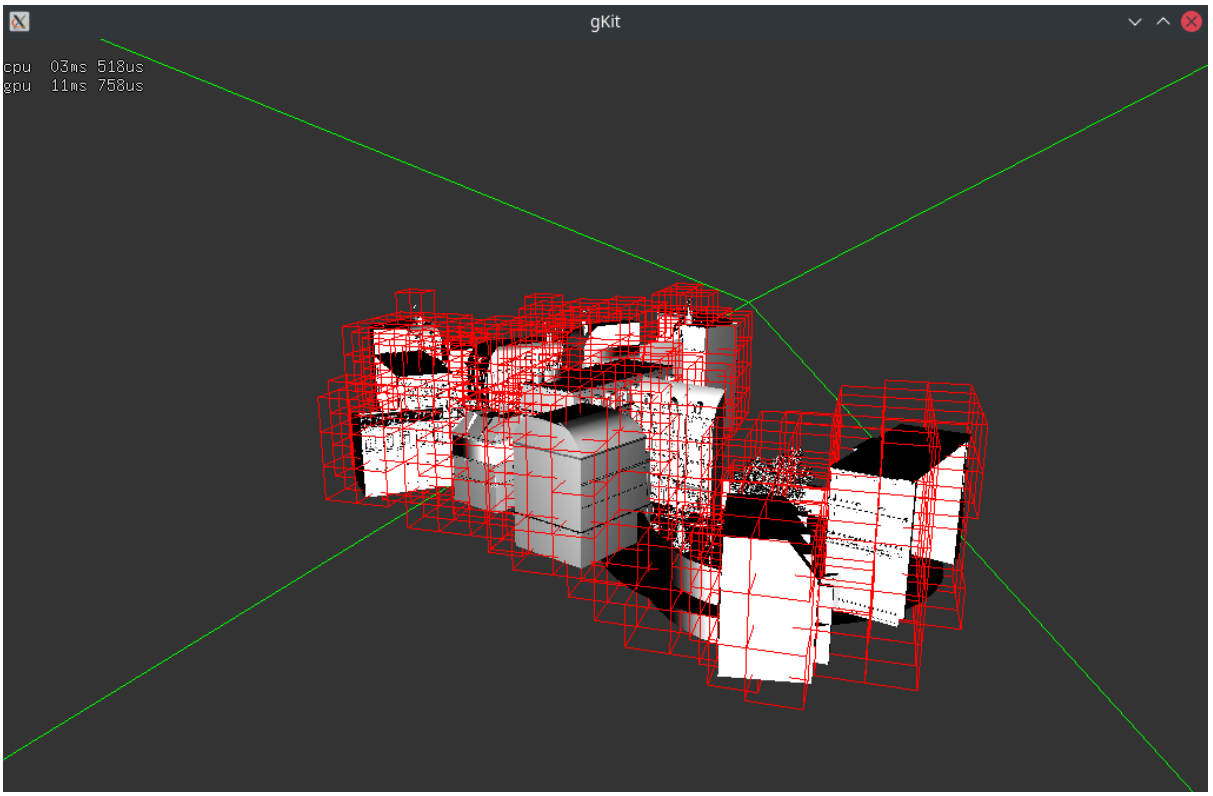
Connaissant pmin et pmax, les points extrêmes d'une boîte alignée sur les axes, comment calculer les coordonnées des 8 sommets de la boîte ?

On peut facilement trouver les points d'une boite alignée sur les axes. Ce sont les 8 combinaisons différentes des coordonnées des points pmin et pmax.

Comment faire pour découper la scène en plusieurs blocs ?

Il suffit de subdiviser l'objet en blocs puis d'itérer sur tous les triangles de l'objet pour tester leur appartenance à une boîte.

La scène découpée en boîtes



Vue caméra frustum	Vue externe