TP OpenGL 7
Introduction au format OBJ

Licence Informatique 3ème année

Année 2021-2022

1 Introduction

Les logiciels d'infographie utilisent généralement leur propre format de sauvegarde des données 3D qu'ils créent, ces formats étant la plupart du temps incompatibles entre-eux. La nécessité de pouvoir échanger facilement des données 3D entre logiciels a donné naissance à différents formats simplifiés, pris en charge par de nombreuses applications. Dans ce TP nous allons étudier l'un de ces formats, le OBJ, qui est un format texte assez simple (du moins pour ce qui sera vu) et peut être appréhendé assez rapidement dans ses fonctionnalités de base.

2 Les sommets

Dans le format OBJ, un sommet est décrit par la syntaxe suivante :

```
v  x  y  z
```

où le caractère `v` introduit un sommet (*vertex*) et `x y z` représentent les coordonnées de ce sommet. Généralement, la première partie d'un fichier OBJ consiste à énumérer les sommets qui composent la scène et/ou un objet, mais ils peuvent apparaître n'importe où dans le fichier.

Exemple Le fichier `cube01.obj` fourni avec cet énoncé contient 8 sommets, qui permettront par la suite de représenter les 6 faces carrées d'un cube. Son contenu est le suivant :

```
# les sommets d'un cube
v 1.000000 1.000000 -1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 1.000000 -1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 1.000000
```

Notez que toute ligne qui commence par un caractère `#` est considérée comme un commentaire.

Application Complétez l'application fournie avec cet énoncé, de manière à :
- créer une structure de données permettant de représenter un point 3D par ses coordonnées (structure ou classe) ;
- créer un vecteur C++ permettant de stocker des points 3D. Ce vecteur sera défini comme une variable globale ;
- écrire une fonction permettant de relire le contenu d'un fichier OBJ et de ranger les points 3D lus dans ce vecteur.

3 Les faces

Une face d'un objet est décrite par l'une des deux syntaxes suivantes (version simplifiée) :
- `f is1 is2 is3` pour un triangle ;

— `f is1 is2 is3 is4` pour un quadrilatère.

Ici, les is_i représentent les indices des sommets dans le tableau global des sommets. Notez cependant que les indices des sommets, dans le format OBJ, commencent à 1, alors que dans le vecteur, ils commenceront à 0.

Exemple Le fichier `cube02.obj` fourni avec cet énoncé complète le fichier précédent, en introduisant les 6 faces du cube à partir des 8 sommets qui y figurent :

```
# les sommets d'un cube
...
# les faces du cube
f 1 5 7 3
f 4 3 7 8
f 8 7 5 6
f 6 2 4 8
f 2 1 3 4
f 6 5 1 2
```

Application Complétez votre application de manière à :

- créer une structure de données/ une classe permettant de représenter un triangle ;
- créer un vecteur C++ permettant de stocker des triangles, ce vecteur étant également défini comme une variable globale ;
- compléter votre fonction de lecture du format OBJ de telle sorte qu'elle puisse relire les faces qu'il contient et construise les triangles corrrespondant. On précise qu'un quadrilatère formé des points A, B, C et D devra donner lieu à la création des deux triangles ABC et ACD ;
- compléter votre méthode de dessin de telle sorte qu'elle dessine les faces qui se trouvent dans le vecteur de triangles à chaque appel.

4 Les objets

Une scène est généralement composée de plusieurs objets. Ceux-ci peuvent être différenciés de diverses façons, l'une d'entre-elles étant de définir un nom d'objet par l'intermédiaire de la syntaxe suivante :

```
o nom_de_l'objet
```

Tout ce qui suit cette définition est alors considéré comme faisant partie du même objet. Le fichier `cube03.obj` illustre ce procédé, en définissant deux objets différents dans la scène :

```
# les sommets d'un cube
...
# les faces du cube
o cube01
f 1 5 7 3
...
# les sommets d'une pyramide
...
# les faces d'une pyramide
o pyramide
f 10 11 12 13
...
```

Application Complétez votre application de manière à :

- créer une structure de données/une classe permettant de représenter un objet 3D. Ce dernier aura à minima un nom et une liste de faces. La liste de faces globales pourra alors être supprimée et sera remplacée par un vecteur d'objets 3D ;
- compléter votre fonction de lecture du format OBJ de telle sorte qu'elle puisse identifier et créer les objets composant une scène ;
- complétez votre structure/classe d'objet 3D de manière à ce qu'elle dispose de sa propre fonction d'affichage. Cette fonction sera appelée par la fonction `dessiner` pour les objets composant la scène.

5 Les matériaux

Les objets définis dans le fichier OBJ peuvent être associés à un matériau, permettant de préciser la manière dont cet objet interagit avec la lumière. Les matériaux sont définis dans un second fichier, d'extension `.mtl` (*Material Template Library*), avec de nombreux paramètres. Nous présentons ci-dessous une version très simplifiée des matériaux, qui est suffisante pour un premier aperçu ;

- `Ka r v b` : spécifie le coefficient de réflexion ambiant du matériau pour chaque longueur d'onde `r`, `v` et `b` (valeur comprise entre 0 et 1) ;
- `Kd r v b` : spécifie le coefficient de réflexion diffus du matériau pour chaque longueur d'onde `r`, `v` et `b` (valeur comprise entre 0 et 1) ;
- `Ks r v b` : spécifie le coefficient de réflexion spéculaire du matériau pour chaque longueur d'onde `r`, `v` et `b` (valeur comprise entre 0 et 1) ;
- `Ns s` : spécifie la valeur du coefficient de brillance, généralement comprise entre 0 et 100 ;
- `map_Kd filename` : spécifie le nom d'un fichier image qui correspond à une texture présente dans le matériau.

Un matériau est créé par le mot clé `newmtl`, suivi de son nom et de ses caractéristiques :

```
# materiel gris clair
newmtl materiel01
Ns 96.078431
Ka 1.000000 1.000000 1.000000
Kd 0.640000 0.640000 0.640000
Ks 0.500000 0.500000 0.500000
map_Kd  opengl.bmp
```

Comme pour les fichiers OBJ, une ligne commençant par un caractère `#` correspond à un commentaire. De même, un fichier MTL peut contenir un nombre quelconque de matériaux, qui seront différenciés par leur nom.

Le lien entre les matériaux et les objets contenus dans un fichier OBJ se fait en deux étapes :

1. définir dans le fichier OBJ le nom du ou des fichiers contenant les matériaux, via le mot-clé `mtllib` : `mtllib nom.mtl` ;
2. préciser, pour chaque objet, le matériel à utiliser, via le mot-clé `usemtl` : `usemtl nomMateriel`.

Le fichier `cube04.obj` illustre ce procédé, en associant deux matériaux définis dans le fichier `cube04.mtl` aux deux objets présents dans la scène :

```
mtllib cube04.mtl

# les sommets d'un cube
...
# les faces du cube
o cube01
usemtl materiel01
f 1 5 7 3
...
# les sommets d'une pyramide
...
# les faces d'une pyramide
o pyramide
usemtl materiel02
f 10 11 12 13
...
```

Application Complétez votre application de manière à :

- créer une structure/classe permettant de représenter un matériel ;
- créer un vecteur C++ permettant de stocker les matériaux, ce vecteur étant défini comme une variable globale ;
- créer une fonction permettant de relire un fichier au format MTL, de créer les matériaux correspondants et de les ranger dans le tableau de matériaux ;
- modifier la fonction de lecture d'un fichier OBJ de telle sorte qu'elle puisse relire un fichier de matériaux et associer un matériel aux objets lorsque cela est requis. On précise qu'il faudra ajouter une information de matériel à chaque objet (par exemple un indice vers le matériel dans le tableau global de matériaux) et qu'il faudra affecter un matériel par défaut aux objets si aucun materiel n'est précisé dans le fichier OBJ ;

— modifier votre application de manière à visualiser l'effet du matériel : dans un premier temps, vous vous contenterez d'utiliser la valeur du coefficient `Kd` comme valeur de couleur associée à l'objet.

6 Textures et normales

Un fichier OBJ peut contenir des informations liés aux corodonnées de textures et aux noramles à associer à chaque sommet de la scène. Le format de ces informations est donné ci-après.

6.1 Coordonnées de texture

Une coordonnée de texture est introduite par la syntaxe suivante :

```
vt  s t
```

où `s` et `t` représentent une coordonnée dans une texture. Comme pour les sommets, ces coordonnées peuvent apparaître n'importe où dans le fichier et doivent être stockées, après lecture, dans un vecteur global. Les coordonnées de texture seront ensuite associées à chaque point via leur indice dans ce vecteur, en se rappelant que les indices dans un fichier OBJ commencent à 1.

6.2 Normales

Une normale est introduite par la syntaxe suivante :

```
vn  nx ny nz
```

où `nx`, `ny` et `nz` représentent les coordonnées de la normale. Comme pour les sommets et les coordonnées de texte, ces normales peuvent apparaître n'importe où dans le fichier et doivent être stockées, après lecture, dans un vecteur global. Les normales seront ensuite associées à chaque point via leur indice dans ce vecteur, en se rappelant que les indices dans un fichier OBJ commencent à 1.

Application : Modifier votre application pour qu'elle puisse relire et stocker toutes les coordonnées de texture et toutes les normales présentes dans un fichier OBJ. Vous pourrez utiliser le fichier `cube05.obj` pour effectuer vos tests.

6.3 Application des textures et des normales

Les coordonnées de texture et les normales doivent être associées aux sommets qui composent chaque face des objets. Pour ce faire, la syntaxe de définition des faces est modifiée comme suit :

```
f  v/vt/vn  v/vt/vn  v/vt/vn  v/vt/vn
```

où `v` représente un indice de sommet, `vt` un indice de coordonnée de texture et `vn` un indice de normale. A noter que ces deux derniers indices ne sont pas toujours présents. On peut donc avoir des syntaxes intermédiaires de la forme :

```
f  v v v v
f v//vn  v//vn v//vn
f v/vt v/vt v/vt
```

ce qui complique un peu leur relecture ...

Application : Modifier votre application pour qu'elle puisse décoder ces syntaxes et stocker les indices relus dans les objets correspondants. Vous utiliserez ensuite toutes ces informations pour obtenir un rendu qui intègre les textures et l'éclairage. Vous pourrez utiliser le fichier `cube06.obj` pour vos tests.