UJF - M1 Informatique 2005-2006 : Infographie
TP2 : Maillages et couleurs

7 octobre 2005

Dans ce TP nous verrons comment manipuler un maillage 3D, le visualiser à l'écran grâce à `openGL` ainsi que le lissage de Gouraud pour les couleurs.

1 Maillage "simple"

1. Ouvrez le fichier `Maillage.h` et regardez les différents types définis ainsi que les fonctions allouer et liberer qui vous permettrons de manipuler le type `Maillage`.
2. Ouvrez le fichier `main.cpp`.
   (a) Regardez la procédure `initRosen` qui remplit le maillage `mRosen`. Que pouvez vous dores et déjà dire sur le maillage (regardez les valeurs affectées aux coordonnées) ?
   (b) Regardez la procédure `dessiner` : c'est elle qui est exécutée à chaque affichage de la fenêtre `openGL`. Pour l'instant nous nous intéressons au cas "`SIMPLE`" qui appele la procédure `dessinerMaillageMat`.

1.1 Dessiner un triangle

Compilez le projet en tapant "`make`". Exécutez le programme en tapant "`./run`". Dans la fenêtre openGL rien ne s'affiche pour l'instant, c'est normal.

Comment dessine t-on un triangle en openGL ?

```
glBegin(GL_TRIANGLES); // Dit à openGL que l'on va dessiner des triangles
glVertex3f( 0.0, 0.0, 0.0 ); // Le premier point (0,0,0)
glVertex3f( 1.0, 0.0, 0.0 ); // Le deuxième point (1,0,0)
glVertex3f( 1.0, 1.0, 0.0 ); // Le troisième point (1,1,0)
glEnd(); // Termine le tracé de triangles (OBLIGATOIRE!)
```

OpenGL va dessiner un triangle pour chaque paquet de 3 "`glVertex3f`" , pour dessiner plusieurs N triangles à la suite, il n'est pas nécessaire d'utiliser N

"glBegin->glEnd", un seul suffira.

Comment affecte t-on une couleur RGB à un triangle ? Il suffit d'ajouter la commande "`glColor3f`" avant le premier vertex (les couleurs vont de "0.0" à "1.0", ainsi le blanc s'écrit "1.0,1.0,1.0") :

```
glBegin(GL_TRIANGLES); // Dit à openGL que l'on va dessiner des triangles
glColor3f( 1.0, 0.0, 0.0 ); // Rouge
glVertex3f( 0.0, 0.0, 0.0 ); // Le premier point (0,0,0)
glVertex3f( 1.0, 0.0, 0.0 ); // Le deuxième point (1,0,0)
glVertex3f( 1.0, 1.0, 0.0 ); // Le troisième point (1,1,0)
glEnd(); // Termine le tracé de triangles (OBLIGATOIRE!)
```

1. Manipulez la fenêtre openGL avec la souris pour tourner autour du triangle.
2. Qu'observez vous ?

1.2 Dessiner un maillage

Exécutez maintenant le programme en tapant "`./run -t 1`" (pour obtenir la liste des paramêtres utilisables, tapez "`./run -h`").

Ouvrez les fichiers `dessiner.h` et `dessiner.cpp`. Ces fichiers définissent les procédures (au départ incomplètes) qui sont utilisées pour le rendu de maillages. La première procédure `dessinerMaillageMat` est écrite pour permette de dessiner un maillage simple (de type "matricielle" comme `mRosen`).

1. Ecrivez les commandes openGL permettant de dessiner les triangles associés au maillage.
2. Que pouvez-vous dire sur l'orientation des triangles ?
3. Que pouvez-vous dire sur les couleurs ?

1.3 Dessiner un maillage par bandes de triangles

Exécutez maintenant le programme en tapant "`./run -t 2`".

Lorsque l'on a un maillage représentant une suite de triangles collés les uns aux autres, il est possible de les dessiner par bande. La commande openGL "`glBegin(GL_TRIANGLE_STRIP)`" permet de dessiner une suite de triangle en minimisant le nombre de `vertex` envoyés. Il va dessiner triangles après triangles en se basant sur paquets des 3 derniers vertex :

```
glBegin(GL_TRIANGLE_STRIP); // Dit à openGL que l'on va dessiner des bandes de triangl
glVertex3f( 0.0, 0.0, 0.0 ); // Le premier point (0,0,0)
glVertex3f( 1.0, 0.0, 0.0 ); // Le deuxième point (1,0,0)
glVertex3f( 0.0, 1.0, 0.0 ); // Le troisième point (0,1,0) => le triangle (0,0,0),(1,0
```

```
glVertex3f( 1.0, 1.0, 0.0 ) ; // Le quatrième point (1,1,0) => le triangle (1,0,0),(0,1
glEnd() ; // Termine le tracé de triangles (OBLIGATOIRE!)
```

1. Complétez la procédure `dessinerMaillageMatStrip` suivant cette idée
2. Vous devriez obtenir le même résultat visuel que précédemment, qu'observez vous entre cette méthode et la précédente si vous utilisez une plus grande finesse d'échantillonage (300x300 par exemple) ?

2 Lissage de Gouraud

Maintenant que vous savez visualiser un maillage simple, nous allons régler ce problème de coloriage des triangles. L'aspect visuel de la surface est très moyen car par défaut, openGL va effectuer le remplissage d'un triangle par une seule et unique couleur, ce type de remplissage est noté par "`GL_FLAT`". Si l'on veut utiliser le modèle de Gouraud, il suffit d'appeler la procédure openGL "`glShadeModel(GL_SMOOTH)`".

La procédure `dessinerMaillageMatGouraud` est appelée en tapant "`./run -t 3`".

1. Vous pouvez maintenant définir une couleur par vertex simplement en plaçant un "`glColor3f`" devant chaque "`glVertex3f`".
2. Pour le même échantillonnage, qu'observez vous entre les deux méthodes de remplissage ?
3. Essayez la commande "`./run -t 4`" qui dessine une surface animée (maillage `mSin`). Etudiez le code associé.

3 Maillage indexé

Nous allons maintenant manipuler des maillages indexés. Reportez vous à la seconde partie du fichier `Maillage.h` pour la définition du nouveau type `MaillageIndexe`.

Vous pouvez voir un exemple de l'utilisation d'une telle structure de données dans la procédure `initTetra` qui remplit le maillage `mTetra`.

1. Complétez la procédure de rendu d'un maillage indexé `dessinerMaillage`.

4 Pour aller plus loin : éclairage

Dans la procédure `main`, décommentez le bout de code marqué `ECLAIRAGE`. Ceci active la fonctionnalité d'éclairage en openGL.

Maintenant qu'un lumière est présente (éclairant du haut vers le bas, dans la direction des Z négatifs), openGL applique un modèle d'illumination plus complet comportant de la diffusion et de la spécularité. Comme il a était vu en cours, de tels modèles sont principalement basés sur la connaissance d'un la

normale de la surface. OpenGL utilise un modèle simplifié d'illumination où la normale est uniquement définie en chaque vertex. L'illumination (ie. la couleur) est calculée en chaque vertex et est ensuite interpolée pour le remplissage du triangle (comme vu précédemment).

Il est donc maintenant nécessaire de disposer d'une normale (unique !) en chaque vertex du maillage.

1. Modifiez la structure MaillageIndexe pour qu'elle puisse aussi stocker les normales.
2. Ecrivez une fonction qui calcule les normales pour chaque vertex d'un maillage.
3. A quelle occasion doit-on calculer les normales ?
4. Pour indiquer à openGL quelle est la normale en un vertex, il suffit, comme pour la couleur, d'ajouter une commande "`glNormal3f`" avant la commande "`glVertex3f`" :

```
glBegin(GL_TRIANGLES);
glColor3f( 1.0, 0.0, 0.0 );
glNormal3f( 0.5, 0.0, 0.5 ); // Le vecteur normal associé au vertex est (0.5,0,0.5
glVertex3f( 0.0, 0.0, 0.0 ); // Le premier point (0,0,0)
[...]
glEnd()
```

5. Réessayez de visualiser l'animation de la fonction sinus en y ajoutant les normales. Qu'observez vous maintenant ?
6. Une normale doit être normalisée (c'est-à-dire que sa norme doit être égale à 1). En openGL, que se passe t-il si la norme du vecteur n'est pas unitaire ?