UJF - M2 IICAO 2005-2006 : Infographie
TP7 : Textures 2D

28 novembre 2005

Aujourd'hui nous allons voir comment utiliser des textures en openGL.

1 Textures à partir d'image

1.1 Chargement de la texture

En openGL, il est possible de charger plusieurs textures en mémoire, mais une seule est active à la fois. Voici comment, à partir d'une image PPM, charger la texture associée sur la carte graphique (cf initdonnees) :

```
// Charge l'image à partir du fichier
image1 = new PPMImage( "mon_image.ppm" ) ;
// Alloue 1 texture, l'identificateur de cette texture est stocké dans le tableau te
glGenTextures(1, texName) ;
// Une seule texture à la fois en openGL : active celle qui vient d'être allouée
glBindTexture(GL_TEXTURE_2D, texName[0]) ;
// Maintenant charge l'image en mipmap (plus simple)
gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, image1->sizeX, image1->sizeY, GL_RGB, GL_UN
```

1.2 Coordonnées de texture

En cours, vous avez vu qu'à chaque vertex d'un maillage texturé est associé des coordonnées de textures (que l'on notera s et t). Etant donné un triangle où les trois sommets ont des coordonnées de texture :au moment du remplissage de ce triangle, pour chaque pixel, openGL va interpoler les coordonnés de texture des sommets en fonction de la position du pixel. Ainsi il pourra aller chercher la position du pixel correspondant dans la texture.

Pour associer à un vertex une coordonnée de texture, il suffit avant le glVertex3f, d'utiliser la procédure glTexCoord2f :

```
glTexCoord2f(s, t) ;
glVertex3f(x, y, z) ;
```

En openGL, une texture est stockée en mémoire dans un domaine de taille [0 :0]x[1 :1] (la texture-image originale est donc transformée en carré sur la

carte graphique). OpenGL vous permet de définir deux comportements lorsque, pour un sommet, vous lui donnez des coordonnées de texture en dehors de ce domaine (par exemple : *(s,t)=(-0.4,+2)*). Soit il coupe à l'intervale [0 :0]x[1 :1] (GL_CLAMP), soit il la répête (GL_REPEAT) (pratique pour les textures répétables). Il est possible de définir un comportement différent dans chaque direction (s et t).

1. Regargez la procédure dessinerFaceTex et, pour les valeurs initiales de xmin et xmax, faites un schéma du "mapping" entre le carré et la texture.
2. Avec les touches 'x/X' et 'c/C' qui modifient xmin et xmax, obtenez l'image initiale (celle du fichier PPM). Sortez du domaine [0 :0]x[1 :1], qu'observez vous et déduisez en l'option utilisée.
3. Appliquez une rotation à la texture (et non aux sommets du maillage), pour cela vous disposez de deux méthodes :
   (a) Utiliser des valeurs de cosinus et sinus pour les coordonnées de texture
   (b) Appliquer une transformation à la matrice des textures (il est possible d'appliquer toutes sortes de transformations à cette matrice : rotations, changement d'échelle et même projection !) :

```
glMatrixMode( GL_TEXTURE ) ;
glRotate( ... ) ;
// Pour revenir dans la matrice modelview :
glMatrixMode( GL_MODELVIEW ) ;
```

1. (maintenant vous pouvez changer les valeurs initiales de xmin et xmax pour visualiser toute l'image dès le début)

1.3 Mipmapping

La texture chargée en mémoire l'est sous forme de mipmap (reportez vous à la procédure initGLTextures) :

1. Rapprochez vous du plan, le résultat se dégrade. Pour quelle raison ? Si vous étiez la machine openGL, quelle solution proposeriez vous pour améliorer le rendu ?
2. Référez vous à la manpage de glTexParameterf pour avoir plus d'information concernant les choix des pixels en fonction des coordonnées de texture. Pourquoi n'y a t-il que deux paramêtres possibles pour l'option GL_TEXTURE_MAG_FILTER ?
3. Agrandissez le carré 3D, utilisez la répétition de texture en s et t et utilisez la seconde texture ("texName[1]"). Eloignez vous de l'objet et utilisez la touche 'm' pour changer le paramètre de GL_TEXTURE_MIN_FILTER. Expliquez vos observations ainsi que le fonctionnement de chacun des modes, ses avantages et ses inconvénients.

1.4 Mélange des couleurs

Il existe différentes méthodes d'applications d'une texture : par exemple que chaque couleur du pixel rendu est celle de la texture ou que c'est un mélange de la couleur provenant de la texture et de la couleur donnée par le glColor3f (ou le glMaterial).

1. Référez vous à la manpage de glTexEnvf.
2. Utilisez la touche 'e' pour changer entre ces différents modes et expliquez leur fonctionnement.
3. A partir des 2 images chargées (brick et damier), créez une troisième texture en RGBA avec pour RGB les mêmes valeurs que l'image "brick" et possédant un canal alpha (l'opacité) en fonction du damier noir et blanc (par exemple : noir = opaque). Essayez les différents modes avec cette nouvelle texture. Pour cela faites bien attention d'utiliser le mode RGBA pour la nouvelle texture (il y a donc 4 octets par pixel).

2 Animer une texture

2.1 Texture statique

Vous pouvez animer une texture plaquée en modifiant avec le temps les coordonnées de texture associées aux sommets.

Basez vous sur la texture de brick répétée **une vingtaine de fois**. Nous souhaitons l'animer en perturbant légèrement les coordonnées de texture (par exemple avec une fonction sinus).

1. Que faut-il faire pour obtenir une animation fine (détaillée, avec des hautes fréquences spatiales) ? (ie. qu'est-ce qui vous en empêche avec le maillage précédent)
2. Réalisez cette modification et perturbez la texture en fonction du temps.

2.2 Texture dynamique et textures procédurales

Une texture est une suite de pixels. Vous n'êtes pas obligé de vous baser sur une image PPM pour en créer une, une fonction en C peut s'en charger.

1. Ecrivez une fonction qui remplisse procéduralement une texture **répétable** et plaquez la sur le maillage précédent.