Master 2 Informatique filière AVR

TD de préparation aux TP d'Artoolkit

Exercice 1. OpenGL : la matrice modelview

OpenGL considère deux matrices :

- la matrice **modelview** matrix qui définit comment les objets sont transformés dans le repère monde (aligné avec le marqueur 3D) : c'est la matrice extrinsèque 3x4 [**R t**] vue en cours, complétée d'une quatrième ligne [0 0 0 1] (nous verrons plus loin pourquoi cette quatrième ligne),

- la matrice de **projection** qui définit les propriétés de la caméra relatives à la projection perspective (zoom, rapport d'aspect, point principal). C'est la matrice intrinsèque 3x3 **K** vue en cours, complétée d'une quatrième ligne et d'une quatrième colonne, utilisées en particulier de gérer le clipping. Cette matrice est définie une fois pour toute dans Arttolkit. Nous ne la considérerons plus dans la suite de ce TD.

La fonction **glMatrixMode**(GLenum *mode*) permet de dire sur laquelle de ces deux matrices (*mode* = GL_MODELVIEW ou *mode* = GL_PROJECTION) seront appliquées les opérations matricielles apparaissant dans les lignes suivantes du code.

1. À quel produit de trois matrices correspondra la matrice modelview à l'issue des commandes suivantes ? Quel est le résultat de ce produit matriciel ?

```
argConvGlpara(trans, gl_para);
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd( gl_para );
glRotatef(45.0f, 0.0f, 0.0f, 1.0f);
glTranslatef( 20.0f, 0.0f, 0.0f );
glutSolidCube( 40.0f);
```

2. À quoi ressemblera le cube sur son marqueur ?

3. Mêmes questions en inversant l'ordre des appels aux fonctions `glRotatef` *et* `glTranslatef`.

Cet exemple montre également que la matrice de changement de repère [R t] peut être vue comme une matrice de transformation rigide appliquée aux points de la scène (exprimés dans le repère monde).

Exercice 2. OpenGL : la pile de matrices

On souhaite dans cet exercice écrire le code C qui permettra d'afficher deux cubes posés sur le marqueur, l'un translaté de 10 cm dans la direction de l'axe x, l'autre translaté de 10 cm dans la direction opposée.

1. Le code ci-dessous résout-il ce problème ? Quel résultat obtient-on ?

```
argConvGlpara(trans, gl_para);
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd( gl_para );
// premier cube
glTranslatef( 10.0f, 0.0f, 0.0f );
glTranslatef( 0.0f, 0.0f, 20.0f );
glutSolidCube( 40.0f);
// deuxième cube
glTranslatef( -10.0f, 0.0f, 0.0f );
glTranslatef( 0.0f, 0.0f, 20.0f );
glutSolidCube( 40.0f);
```

Pour pallier ce problème, OpenGL utilise une pile de matrices. Toutes les opérations matricielles effectuées au sein du code s'adressent à la matrice active, c'est-à-dire à la matrice au sommet de la pile. La fonction **`glPushMatrix()`** permet de recopier la matrice active en haut de la pile tandis que la fonction **`glPopMatrix()`** permet de détruire la matrice située au sommet de la pile.

Utilisez ces deux fonctions pour résoudre notre problème **en utilisant le moins de lignes de code possible.**

Exercice 3. Changements de repère

On suppse que deux marqueurs ont été détectés par Artoolkit dans une image vidéo. Soit [**R**1 **t**1] et [**R**2 **t**2] les matrices extrinsèques obtenues respectivement pour chacun des deux marqueurs. Donnez deux calculs différents permettant de calculer la distance entre les centres des marqueurs :

- l'un ramenant les coordonnées du centre du deuxième marqueur dans le repère du premier marqueur,
- l'autre ramenant les coordonnées des centres des deux marqueurs dans le repère de la caméra.

Obtient-t-on bien la même distance ?