Les matrices sous 3D Studio MAX

par Antoche

1 Introduction

Dans tous les moteurs 3D standards et classiques, un objet est généralement constitué de deux données : une matrice, représentant la position et l'orientation de l'objet dans l'espace, et des données permettant de l'afficher. Ici je ne vais parler que des meshes, pour que la compréhension soit plus simple, mais ce que je vais dire s'applique à tous les objets (`INode*`) 3DSMAX, meshes ou non. Pour dessiner le mesh, il suffit donc de transformer ses vertices par la matrice de l'objet (ou de charger cette matrice dans son API préférée qui se chargera du boulot) et c'est terminé.

2 Allons-y

Dans 3DSMAX, cette matrice est représentée par le "Pivot", qui est visible et aisément manipulable. L'utilisateur a notamment le moyen de modifier ce pivot sans "toucher" à l'objet lui-même, en utilisant la commande Hierarchy>Affect Pivot Only. C'est-à-dire que l'objet sera affiché exactement au même endroit, mais sa matrice sera différente, ce qui permet par exemple d'avoir un axe de rotation différent de celui de base. Mais la loi mathématique de la bijection appliquée à la transformation matricielle pour les matrices non singulières (ainsi que le bon sens) nous fais remarquer que si on modifie la matrice, il faut forcément aussi modifier les vertices eux-mêmes si l'on veut qu'à la fin de l'opération l'objet soit dessiné au même endroit qu'avant. Cela veut donc dire qu'avec le modèle décrit précédemment, il faudrait modifier un à un les vertices de l'object à chaque changement de pivot... Pas super pratique (et imaginez la galère si par exemple il s'agit non plus de meshes mais d'objets procéduraux).

3 La solution

Elle est toute conne, est c'est celle qu'utilise 3DSMAX pour éviter ces opérations coûteuses. Il utilise en fait deux matrices par objet : la première, la NodeMatrix, définit la position du pivot. La seconde, l'ObjectMatrix, définit la position des vertices. Elles sont initialement égales, et quand on bouge l'objet avec son pivot (l'opération standard), elles sont toutes les deux modifiées "de la même façon" (cf. plus bas). Par contre, lorsqu'on déplace uniquement le pivot (sans l'objet), seule la NodeMatrix bouge, et quand on déplace uniquement l'objet (sans le pivot), seule l'ObjectMatrix bouge. Ainsi, plus besoin de toucher aux vertices.

4 Un peu de précisions

Cependant, ces matrices ne sont pas du tout indépendantes. En fait, l'ObjectMatrix doit être déduite de la NodeMatrix. En effet, imaginez que vous fassiez tourner normalement (par modification normale, j'entend une modification qui s'applique à l'objet et qui utilise la NodeMatrix pour pivot) un objet ayant des matrices Node et Object différentes (c'est-à-dire un pivot modifié) autour de son pivot : appliquer la rotation aux deux matrices indépendamment est une mauvaise idée, car cela donnera le mauvais résultat : le pivot sera bien comme il faut, mais l'objet aura tourné autour du pivot défini par ObjectMatrix (i.e. autour de lui-même) et non autour du pivot définit par NodeMatrix. La bonne méthode est d'appliquer la rotation à la NodeMatrix et de déduire l'ObjectMatrix en utilisant l'OffsetMatrix.

5 Késako ? Une troisième matrice ??

L'OffsetMatrix est en fait tout simplement la matrice de passage définissant comment passer de la NodeMatrix à l'ObjectMatrix. Elle se calcule donc tout simplement en faisant $\frac{ObjectMatrix}{NodeMatrix}$. Cette OffsetMatrix est égale à l'identité au départ, et change si l'une des 2 matrices de l'objet (Object ou Node) est modifiée indépendatamment de l'autre. Donc, lorsque l'objet subit une modification normale, c'est à la NodeMatrix d'être modifiée, et l'ObjectMatrix doit en être déduite en faisant tout simplement $NodeMatrix * OffsetMatrix$.

6 Mais tout ca, ca me sert à quoi ?

Et bien, en fait, à rien. Ca fait 2 paragraphes que je vous explique comment fonctionne 3DSMAX en interne, et en fait vous en avez limite rien à battre, parce qu'à moins que vous ne vouliez réutiliser ce système de 2 matrices (plus une) dans votre propre moteur 3D (ce qui est une mauvaise idée), ca ne vous est pas extrêmement utile. Pourquoi est-ce une mauvaise idée ? Et bien parce que le seul avantage est celui précité de ne pas avoir à modifier les vertex un à un lorsqu'on modifie le pivot. Et comme le pivot n'est modifié qu'au moment du dessin des objets et plus du tout dans le rendu, ca ne vous servira finalement pas à grand chose.

7 Mais alors, je me suis fait mystifier là ?

Pas tant que ça. D'abord, si un jour vous faites votre propre modeleur, vous serez à coup sûr amené à utiliser une technique dans ce genre. Vous en présenter une ici et mettre un doigt sur ses particularités peut vous être utile. Et puis même si vous n'en ferez jamais, on ne perd jamais à en savoir un peu plus (en plus ca m'évitera à répondre aux mails des types qui désirent en savoir un peu plus ;)). Et enfin, même si savoir comment 3DSMAX gère ces deux matrices en interne

ne vous est pas ncessaire directement, *savoir à quoi correspondent ces deux matrices est d'une importance vitale pour un développeur de plug-in 3DSMAX, quel qu'il soit*(et là, je pense que vous devez être un peu plus concerné). En effet, imaginez que vous exportiez des modèles 3DSMAX pour une moteur 3D (activité ô combien originale et rare) : viendra bien le moment où il vous faudra choisir quelle matrice prendre (je suppose évidemment que le moteur 3D en question n'utilise qu'une seul matrice par objet). Sachant tout ce que je viens de vous dire plus haut, les deux solutions possibles vous viendront immédiatement à l'esprit :

- Soit utiliser l'ObjectMatrix et les vertices tels quels, auquel cas les vertex seront bien placés mais le pivot dans le moteur 3D sera différent de celui de 3DSMAX s'il a été modifié par l'artiste, ce qui peut être gênant si vous envisagez ensuite d'animer l'objet.
- Soit utiliser la NodeMatrix, et dans ce cas-là vous devrez prétransformer vos vertices par l'OffsetMatrix au moment de l'export afin qu'ils soient bien définis dans le repère de la NodeMatrix et non plus dans celui de l'ObjectMatrix. Dans ce cas-là, le pivot dans le moteur 3D sera le même que celui de 3DSMAX, et les vertices seront au bon endroit.

8 Génial ! Que demander de plus ?

Peut-être voulez-vous savoir comment récupérer toutes ces matrices depuis votre plug-in ? Pour les Node et Object Matrices, c'est assez simple, il suffit d'appeler `INode::GetNodeMatrix()` et `INode::GetObjectMatrix()`. Mais pour l'offset matrix, c'est un peu plus chiant : la doc du SDK préconise l'utilisation de fonctions permettant de récupérer l'offset scale, l'offset position et l'offset rotation pour ensuite reconstruire la matrice, mais ces fonctions ne fonctionnent qu'à certaines conditions (c'est aussi écrit dans la doc, mais en tout petit), alors ne vous faites pas chier : prenez la NodeMatrix et l'ObjectMatrix à un moment donné et calculez la matrice de passage, c'est beaucoup plus simple et ca marche partout !

Voilà, amusez-vous bien,
Antoche
http://www.antoche.fr.st