Rapport d'Algèbre Géométrie

1. L'espace tridimensionnel

1.1 Description de l'espace

Il est décrit par trois composantes x, y et z.

Le point est donc décrit par ce système :
X=abscisse du point
Y=ordonnée du point
Z=profondeur du point

Nous allons travailler dans le repère cartésien (O, X, Y, Z) qui sera notre espace image. Un objet 3D est caractérisé par sa forme et sa position. La forme d'un objet est définie par un ensemble de facettes planes, et une face est logiquement définie par les coordonnées de ses sommets.

Ici l'origine est le centre de la ville. Ce centre est utilisé comme centre de rotation de l'objet.

Ainsi, la position d'un objet dans notre espace image est définie d'une part par les coordonnées de son centre, et d'autre part par son orientation dans l'espace.

Pour déterminer la position d'un objet, nous devons connaître les coordonnées de chacun des sommets qui composent ses faces. En pratique, un même sommet est commun à plusieurs faces.

Cependant, pour pouvoir joindre les points sans joindre ceux qui n'ont pas à l'être (ceci n'est pas un polygone mais une ville), nous choisissons de faire les calculs sur les lignes (avec x, y, z d'origine et x, y, z de destination) plutôt que sur les points. Nous expliquerons les transformations sur les points, la seule modification par rapport au programme est que chaque transformation se fait 2 fois (origine puis destination).

1.2 Les Rotations

Nous avons ici une rotation d'angle a autour de l'origine (donnée en TP).

$x_{A'} = \cos(a)$ et $x_{B'} = -\sin(a)$
$y_{A'} = \sin(a)$ et $y_{B'} = \cos(a)$

Cette opération n'est autre qu'un calcul matriciel, la matrice de la rotation R d'angle a est donc :

$$R = \begin{pmatrix} \text{Cos}(a) & -\sin(a) \\ \text{Sin}(a) & \cos(a) \end{pmatrix} \qquad P = \begin{pmatrix} x \\ y \end{pmatrix}$$

On peut maintenant généraliser le problème à tous les points du plan en utilisant cette matrice R. La rotation d'un point P autour de l'origine devient alors :

$$P' = R \times P = \begin{pmatrix} x.\cos(a) - y.\sin(a) \\ x.\sin(a) + y.\cos(a) \end{pmatrix}$$

Pour passer en 3D, il suffit d'ajouter une dimension à notre matrice, il nous faudra alors 3 axes de rotation pour pouvoir faire tourner la ville dans tous les sens (y compris ceux qui n'ont pas de sens permettant de voir la ville par le bas).

Pour cela, on conserve notre matrice Rotation 2x2 et on l'inscrit dans la matrice identité en laissant la ligne et la colonne de la dimension de l'axe de rotation libre. Ainsi une rotation d'angle 0 selon l'axe voulu n'a pas d'effet (et on va le montrer)

Voici l'allure qu'auront nos matrices pour une rotation autour des axes X, Y et Z :

$$Rx = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(a) & -\sin(a) \\ 0 & \sin(a) & \cos(a) \end{pmatrix}$$

Une rotation d'angle 0 nous permet d'avoir la matrice identité, donc sa multiplication par un point 3D n'aura aucune influence.

$$Ry = \begin{pmatrix} \cos(a) & 0 & -\sin(a) \\ 0 & 1 & 0 \\ \mathrm{Sin}(a) & 0 & \cos(a) \end{pmatrix}$$

$$P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$Rz = \begin{pmatrix} \mathrm{Cos}(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Et voici le point P après une rotation d'axe x, y et z pour un angle ax, ay et az.

$$Px = \begin{pmatrix} x \\ y*\cos(ax) - z*\sin(ax) \\ y*\sin(ax) + z*\cos(ax) \end{pmatrix} \quad Py = \begin{pmatrix} x*\cos(ay) + z*\sin(ay) \\ y \\ -x*\sin(ay) + z*\cos(ay) \end{pmatrix} \quad Pz = \begin{pmatrix} x*\cos(az) + y*\sin(az) \\ -x*\sin(az) + y*\cos(az) \\ z \end{pmatrix}$$

Or, faire une rotation pour le premier axe, puis une pour le second et encore une pour le troisième est assez fastidieux. Ce qui serait agréable à utiliser serait une seule matrice ; Pour cela il suffit de les combiner entre elles. Il s'agit d'un produit matriciel, en effet, pour appliquer une rotation d'axe Ox à P, on calcule P'=Rx*P, donc une rotation a 3 axes est en fait la composée des 3 rotations successives.

$$P' = \begin{pmatrix} (x * \cos(ay) + z * -\sin(ay)) * \sin(ay) + y * -\sin(az) \\ (x * \sin(az) + y * \cos(az)) * \cos(ax) + z * -\sin(ax) \\ (y * \sin(ax) + z * \cos(ax)) * \sin(ay) + z * \cos(ay) \end{pmatrix}$$

1.3 Les Translations

Une translation de vecteur T est une addition vectorielle, P’ est donc la somme de P et de T.

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix} \qquad P' = P + T$$

Il suffit alors d’appliquer la même opération à tous les points de la figure pour appliquer la translation à toute la ville.

1.4 Les homothéties

On peut effectuer aussi facilement un changement d'échelle sur un objet, ce qui permet de le "grossir" ou au contraire de le "rétrécir". Dans ce cas, il faut faire une multiplication de la matrice point P par le facteur r (un réel) de l’homothétie (On utilisera des facteurs 0.9<r<1.1) :

$$P' = r * P = r * \begin{pmatrix} X \\ y \\ Z \end{pmatrix} = \begin{pmatrix} r * x \\ r * y \\ r * z \end{pmatrix}$$

1.5 Conclusion

Ayant un point P, il suffit de composer les trois transformations sur P pour obtenir la position du point P’. il suffit alors de répéter ce processus n fois pour faire bouger notre ville en 3D constituée de n points (ici, le problème est plus complique comme explique dans le chapitre suivant car pour plus de clarté dans le programme, certains points de l’objet sont redondants ; Il y a donc ici plus de n points… Mais le principe reste le même). L’ordre des transformations n’est ici pas important car grossir un objet avant ou après de le faire tourner ou de le translater ne modifie pas le résultat des coordonnées d’arrivée.

2. Représentation de cet espace en bidimensionnel

2.1 Contraintes

Nous avons des coordonnées en 3D, c’est inexploitable pour l’affichage, il faut donc le transformer, pour cela, il faut projeter les points (d’origine et de destination de chaque ligne) sur un plan et tracer les projetés. Comme dit précédemment, les polygone qui constituent la ville contiennent des points communs a plusieurs arêtes, il est inutile de les compter deux fois, cependant comme nous utilisons une structure point d’origine – point de destination pour permettre une meilleure modularité du programme), nous répéterons donc certains calculs 3 fois (pour un polygone de 8 points , sa représentation occupe 24 points).

2.2 Procède de tracé des lignes

Le modèle de trace de ligne est celui de Bresenham (vu en TP) car il nous permet de tracer des lignes sans nous soucier du point haut droite et bas gauche de la fonction « line » qui avait été la source de problèmes d’utilisation (les lignes se croisaient).

2.3 Projection des points

Pour passer d'une dimension 3 à une dimension 2, il faut projeter les coordonnées des sommets qui composent la scène à visualiser. Nous utiliserons la projection perspective pour son rendu réaliste, car elle rend compte des distances et donne véritablement une impression de profondeur à la scène.