

Vous allez réaliser en assembleur x86-64 sous Linux un mini-moteur 3D.

Il se composera de plusieurs étapes.

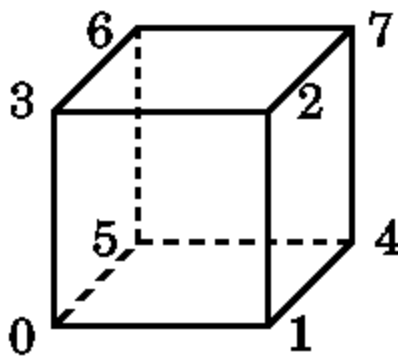
I – Définition d'un objet 3D :

Il se compose de sommets qui sont des points à trois coordonnées x, y et z.

Ces points sont reliés entre eux par des arêtes. Un ensemble d'arêtes reliées entre elles forme une face.

Exemple :

un cube est composé de 8 sommets reliés entre eux par 12 arêtes formant 6 faces :



Les sommets de ce cube, centré sur le repère, auront pour coordonnées :

Numéro de point	x	y	z
0	-100	-100	-100
1	100	-100	-100
2	100	100	-100
3	-100	100	-100
4	100	-100	100
5	-100	-100	100
6	-100	100	100
7	100	100	100

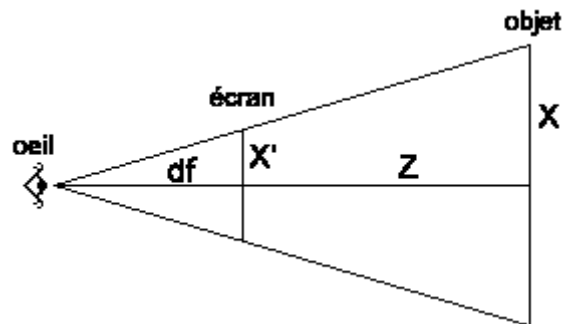
Les faces seront composées des points suivants :

Numéro de la face	Sommets
0	0,1,2,3
1	1,4,7,2
2	4,5,6,7
3	5,0,3,6
4	5,4,1,0
5	3,2,7,6

Les numéros de faces n'ont pas d'importance, à la différence des numéros des points et de leur ordre dans les faces comme on le verra plus tard.

II – Affichage d'un objet 3D sur un écran 2D :

Pour afficher votre objet en coordonnées 3D sur votre écran en coordonnées 2D, vous allez utiliser la méthode de la projection perspective qui se base sur le schéma suivant :



Soit df (distance focale) la distance de l'œil à l'écran,
 X une distance sur de l'objet 3D,
 X' la projection de cette distance sur l'écran et
 Z la distance de l'objet à l'écran (=sa coordonnée en z).

Le théorème de Thalès nous donne la relation suivante :

$$X'/df = X/Z,$$

$$\text{soit } X' = (df * X) / Z.$$

On peut généraliser cela sur la coordonnée en Y :

$$\text{soit } Y' = (df * Y) / Z.$$

Comme dans notre fenêtre, le point en haut à gauche a pour coordonnées (0,0), il faut décaler le point projeté pour simuler un repère centré dans la fenêtre, donc ajouter les coordonnées (X_{off}, Y_{off}) du point central de la fenêtre aux coordonnées du point projeté :

$$\text{soit } X' = (df * X) / Z + X_{off}.$$

et

$$\text{soit } Y' = (df * Y) / Z + Y_{off}.$$

De même, on définit un offset en Z , Z_{off} , qui nous permet de zoomer/dézoomer sur l'objet.

Les formules finales de projection d'un point 3D (X, Y, Z) en point 2D (X', Y') seront donc :

$$X' = (df * X) / (Z + Z_{off}) + X_{off}.$$

et

$$Y' = (df * Y) / (Z + Z_{off}) + Y_{off}.$$

Pour afficher les sommets de votre objet, vous appliquerez ces formules à l'ensemble de vos sommets.

Pour afficher la totalité de votre objet, vous devrez transformer les sommets en point 2D et tracer des lignes entre les sommets selon le tableau définissant les faces.

Exemple :

Pour le cube donné précédemment :

Vous transformez les 8 sommets en points 2D avec les formules de projection.

Vous tracez les faces :

Face 0 : Ligne du point 0 au point 1
 Ligne du point 1 au point 2
 Ligne du point 2 au point 3
 Ligne du point 3 au point 0

Face 1 : Ligne du point 1 au point 4
 Ligne du point 4 au point 7
 Ligne du point 7 au point 2
 Ligne du point 2 au point 1

Etc...

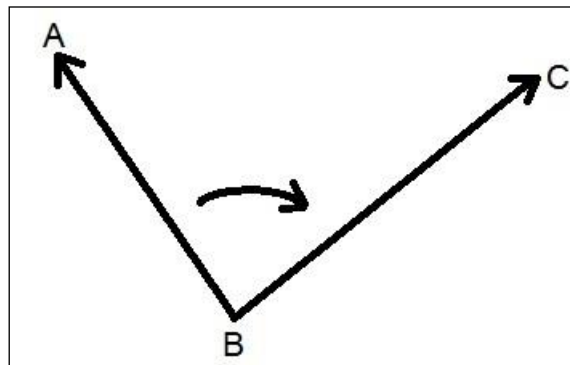
III – Gestion des faces cachées :

Votre objet est pour l'instant affiché en "fil de fer" et on voit à l'écran des faces qui sont normalement cachées par les autres.

Il existe de nombreuses méthodes pour supprimer ces faces cachées, nous allons utiliser la méthode des normales.

Elle se décompose en plusieurs étapes :

- On prend 2 vecteurs de la face concernée orientés dans le sens des aiguilles d'une montre (sens trigonométrique) et avec une origine commune (les points doivent d'abord avoir été transformés en 2D) :



Ici, les vecteurs BA et BC répondent à ces obligations.

Pour le cube, on peut citer les vecteurs 10 et 12 pour la face 0, 12 et 14 pour la face 1, 45 et 47 pour la face 2, etc...

Pour rappel, on calcule les coordonnées d'un vecteur AB avec la formule :

$$\begin{aligned}X_{AB} &= X_B - X_A \\ Y_{AB} &= Y_B - Y_A\end{aligned}$$

- On calcule alors la coordonnée en z du produit vectoriel de ces 2 vecteurs dans un ordre précis (le produit vectoriel n'est pas commutatif) :

Pour les deux vecteurs BA et BC vus précédemment, on calcule :

$$(X_{BA} * Y_{BC}) - (Y_{BA} * X_{BC})$$

Ceci est la coordonnée en Z du vecteur perpendiculaire à la face (ou vecteur normal). Si cette valeur est nulle ou négative, alors cela signifie que la face ne pointe pas vers l'œil et qu'elle ne doit pas être affichée.

Exemple :

Pour notre cube :

- on calcule les coordonnées des vecteurs 10 et 12 :

$$X_{10} = X_0 - X_1$$

$$Y_{10} = Y_0 - Y_1$$

$$X_{12} = X_2 - X_1$$

$$Y_{12} = Y_2 - Y_1$$

- On calcule la normale :

$$(X_{10} * Y_{12}) - (Y_{10} * X_{12})$$

- Si la valeur obtenue est strictement positive, on affiche la face, sinon on passe à la face suivante.

On réitère ce procédé pour toutes les faces de l'objet.

IV – Rotations :

Comme nous avons 3 axes, il y aura 3 rotations distinctes avec 3 angles distincts.

Soient x' , y' et z' , les coordonnées du point x,y,x après rotation.

Selon l'axe X, les transformations seront :

$$\begin{aligned}x' &= x \\y' &= y*\cos(\text{angX}) - z*\sin(\text{angX}) \\z' &= y*\sin(\text{angX}) + z*\cos(\text{angX})\end{aligned}$$

Selon l'axe Y, les transformations seront :

$$\begin{aligned}x' &= z*\sin(\text{angY}) + x*\cos(\text{angY}) \\y' &= y \\z' &= z*\cos(\text{angY}) - x*\sin(\text{angY})\end{aligned}$$

Selon l'axe Z, les transformations seront :

$$\begin{aligned}x' &= x*\cos(\text{angZ}) - y*\sin(\text{angZ}) \\y' &= x*\sin(\text{angZ}) + y*\cos(\text{angZ}) \\z' &= z\end{aligned}$$

On peut enchaîner les rotations sur les différents axes dans n'importe quel ordre.

V – Indications :

Votre programme devra fonctionner pour un **dodécaèdre**.

Pour un programme complet, les opérations se feront dans l'ordre suivant :

- 1 – Rotations
- 2 - Projection des sommets en 2D
- 3 - Calcul des normales
- 4 - Affichage de l'objet

Le minimum demandé, ce sont les étapes 2 et 4.

Ensuite, vous pouvez effectuer soit l'étape 1, soit l'étape 3, soit les 2.

(Voir fichiers exemples à télécharger sur la page du projet)

Pour l'objet, vous définirez

- un tableau nommé *objet* de type DWORD contenant les coordonnées des sommets (en flottant).
- un tableau nommé *faces* de type DWORD contenant les points de chaque face énumérés à la suite (en entier).

Pour le dodécaèdre demandé, vous avez un fichier "dodecaedre.asm" où se trouvent les coordonnées de ses points ainsi que les faces qui le composent.

Exemple :

Pour un tétraèdre :

```
objet:      dd    -100,100,100
            dd    100,100,100
            dd    0,-100,0
            dd    0,100,-100
```

```
faces:      dd    0,1,2,0
            dd    1,3,2,1
            dd    3,0,2,3
            dd    0,3,1,0
```

VI – Requis :

- Vous programmerez **uniquement** en assembleur NASM x86-64 sur Linux.
- Pour les calculs en flottant, vous n'utiliserez que les instructions SSE, à l'exception des calculs trigonométriques (sinus et cosinus) que vous effectuerez avec les instructions X87. Il est d'ailleurs conseillé de les précalculer comme vu dans le TP5.

VII - Fichiers fournis :

compileX11.sh : assemble un programme assembleur contenant du code X11 et génère l'exécutable.

dodecaedre.asm : contient les coordonnées et faces d'un dodécaèdre.

code_pour_dessiner.asm : contient le code permettant de créer une fenêtre et de dessiner dedans.

exemples_de_rendus.pdf : contient des images montrant les résultats attendus.