Projet d'Architecture Matérielle et Assembleur

ESGI – 2i

Soutenance Novembre 2025

Ce projet devra être réalisé en groupe de trois.

Mail de contact : **nneveu@gmail.com**

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

Il se composera de plusieurs étapes.

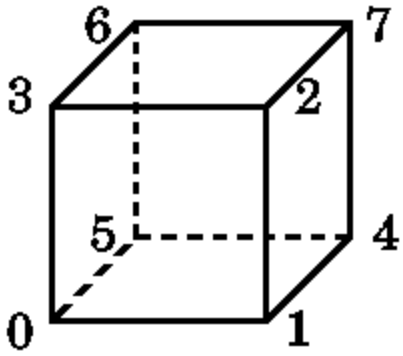
NB : Si vous n'utilisez pas la VM donnée en cours, il vous faudra installer les paquets suivants pour réaliser ce projet : nasm, edb-debugger, build-essential et xorg-dev. Il vous faudra aussi le script qui permet de compiler 'compile64'.

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, pourront avoir 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

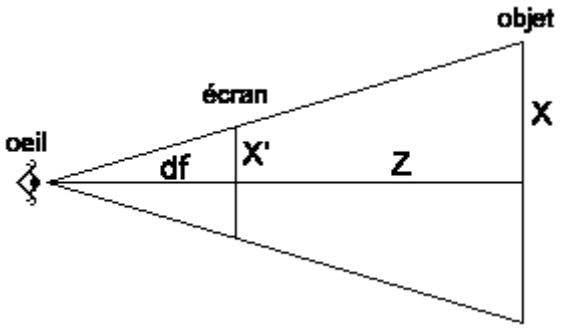
Les faces seront composées des points suivants :

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,

soit X'=(df*X)/Z.

On peut généraliser cela sur la coordonnée en Y :

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 (Xoff,Yoff) du point central de la fenêtre aux coordonnées du point projeté :

soit X'=(df*X)/Z+Xoff.
et
soit Y'=(df*Y)/Z+Yoff.

De même, on définit un offset en Z, Zoff, qui nous permettre 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+Zoff)+Xoff.
et
Y'=(df*Y)/(Z+Zoff)+Yoff.

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 entres ces sommets en deux dimensions 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

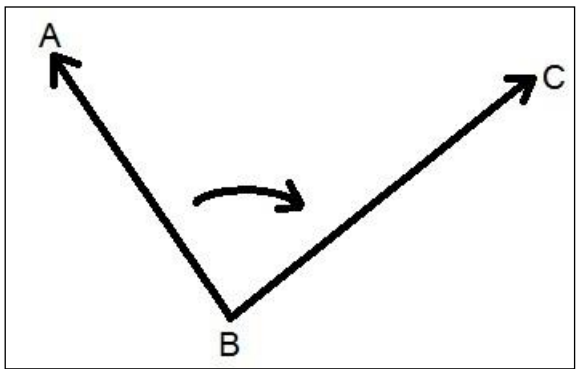
Etc…

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 de l'exemple, on peut citer les vecteurs 1-0 et 1-2 pour la face 0, 1-2 et 1-4 pour la face 1, 4-5 et 4-7 pour la face 2, etc…

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

$$X_{AB}=X_B-X_A$$
$$Y_{AB}=Y_B-Y_A$$

- 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 1-0 et 1-2 :

$X_{1-0}=X_0-X_1$
$Y_{1-0}=Y_0-Y_1$

$X_{1-2}=X_2-X_1$
$Y_{1-2}=Y_2-X_1$

- On calcule la normale :

$(X_{1-0}*Y_{1-2})-(Y_{1-0}*X_{1-2})$

- 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, z après rotation.

Selon l'axe X, les transformations seront :

x' = x
y' = y*cos(angX) - z*sin(angX)
z' = y*sin(angX) + z*cos(angX)

Selon l'axe Y, les transformations seront :

x' = z*sin(angY) + x*cos(angY)
y' = y
z' = z*cos(angY) - x*sin(angY)

Selon l'axe Z, les transformations seront :

x' = x*cos(angZ) - y*sin(angZ)
y' = x*sin(angZ) + y*cos(angZ)
z' = z

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

V – Indications :

Votre programme devra fonctionner au moins pour un **dodécaèdre** dont vous seront fournies les coordonnées.
Mais vous aurez aussi d'autres objets qui vous seront fournis et votre programme devrait fonctionner avec eux au prix de modifications minimes de votre code (pensez à utiliser des constantes avec %define notamment)

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.

Pour les rotations, l'idéal est de passer les valeurs des angles sur la ligne de commande. Mais il sera toléré d'entrer les angles au clavier, ou directement dans le code.

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" où se trouvent les coordonnées de ses points ainsi que les faces qui le composent. Vous aurez aussi un fichier "cone" organisé de la même façon.

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 vues en cours**. Pour les calculs de cosinus et sinus, le programme les précalcule et les range dans deux tableaux de 360 cases de DWORD appelés cosinus et sinus (Exemple : [cosinus+5*DWORD] contient la valeur du cosinus d'un angle de 5°).

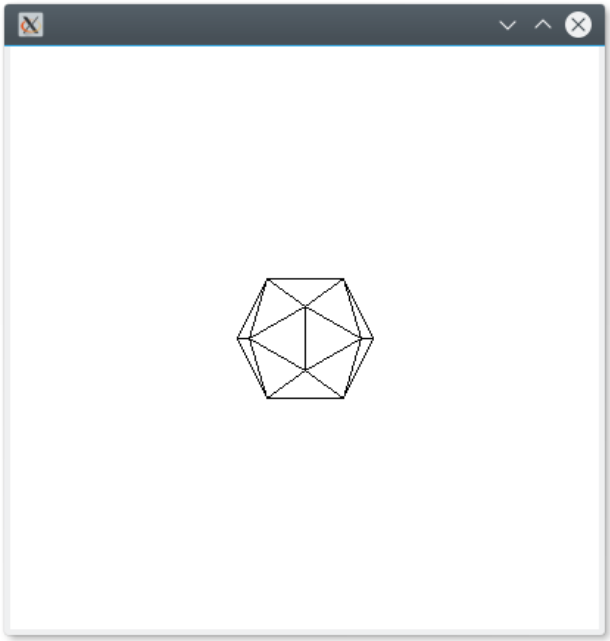
VII - Fichiers fournis :

dodecaedre : contient les coordonnées et faces d'un dodécaèdre.
cone : contient les coordonnées et faces d'un cône.
code_pour_dessiner.asm : contient le code permettant de créer une fenêtre et de dessiner dedans ainsi que le pré-calcul des cosinus et sinus.

VIII - Exemples de rendus attendus pour les différentes étapes du projet :

Etapes 2 et 4 : rendu attendu :

Etapes 2, 3 et 4 : rendu attendu :

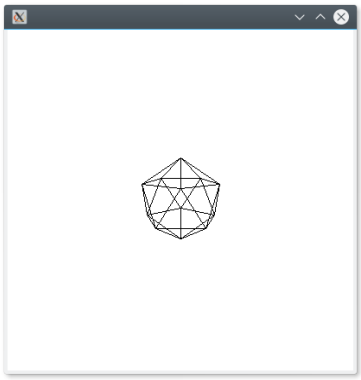


Etapes 1, 2 et 4 : Exemples de rendus attendus :

La ligne de commande suivante (rotation de 45° en X):

```
root@debian7:/asm/projet# ./Projet_etapes1-2-4_dodec 45 0 0
```

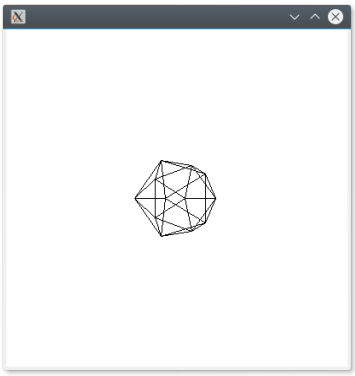
donne le résultat suivant :



La ligne de commande suivante (rotation de 45° en Y):

```
root@debian7:/asm/projet# ./Projet_etapes1-2-4_dodec 0 45 0
```

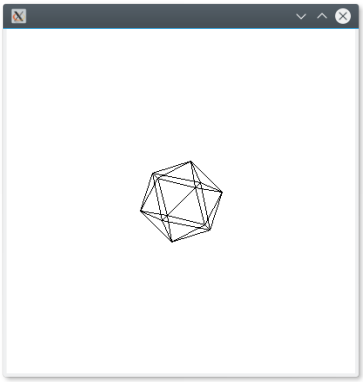
donne le résultat suivant :



La ligne de commande suivante (rotation de 45° en Z):

```
root@debian7:/asm/projet# ./Projet_etapes1-2-4_dodec 0 0 45
```

donne le résultat suivant :

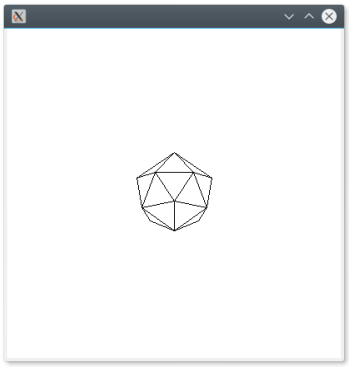


Etapes 1, 2, 3 et 4 : Exemples de rendus attendus :

La ligne de commande suivante (rotation de 45° en X):

```
root@debian7:/asm/projet# ./Projet_final_dodec 45 0 0
```

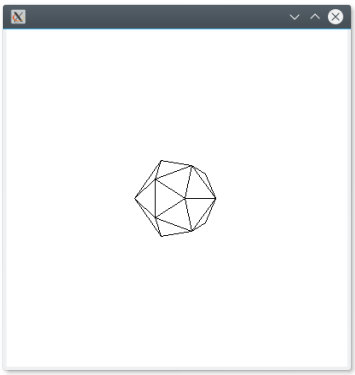
donne le résultat suivant :



La ligne de commande suivante (rotation de 45° en Y):

```
root@debian7:/asm/projet# ./Projet_final_dodec 0 45 0
```

donne le résultat suivant :



La ligne de commande suivante (rotation de 45° en Z):

```
root@debian7:/asm/projet# ./Projet_final_dodec 0 0 45
```

donne le résultat suivant :