

*L'ensemble de ces exercices sont destinés à approfondir votre apprentissage
Merci de ne pas vous servir de ChatGpt ou Copilot*

Nous utiliserons encore la SDL2 pour l'ensemble de ce TP

Tracer de lignes

La SDL propose la fonction `SDL_RenderDrawLine` qui trace une ligne et qui prend en argument les coordonnées des deux points et le `Renderer`.

Cette fonction s'appuie sur l'algorithme de Bresenham qui calcule une pente et qui effectue un nombre d'itération définie. A chaque itération l'algorithme dessine un point sur la nouvelle coordonnée calculée

Soit deux points $X1, Y1$ et $X2, Y2$:

étape 1 → calculer la pente $(Y2 - Y1) / (X2 - X1)$

étape 2 → itérer $X1$ jusqu'à $X2$, à chaque itération ajouter la pente à $Y1$

La force de cet algorithme réside dans sa simplicité et ses optimisations possibles, par exemple le traitement des cas particuliers comme les verticales ou les horizontales.

Il a été longtemps utilisé dans l'ensemble des jeux vidéos des années 80 et 90 et notamment dans les premiers moteurs 3D.

Le source `droites.c` dessine une droite qui effectue une rotation sur 360° en utilisant la fonction `SDL_RenderDrawLine`. Après analyse du code, implémentez votre propre fonction de tracer de ligne en vous appuyant Bresenham. Vous remarquerez des imprécisions. Quelles solutions préconisez vous ? Implémentez-les.

Rotation 3D

Une rotation 3D d'un objet s'appuie sur une succession d'opérations qui s'alimentent du résultat de la précédente pour effectuer une rotation complète sur l'axe des X, Y et des Z

Quelques notions de base

Un point en 3D est composé de 3 coordonnées, le X, Y, Z

Ces coordonnées permettent de situer un point dans un espace représenté par 3 axes :

- l'axe des X
- l'axe des Y
- l'axe des Z

Exemple des coordonnées d'un cube en 3D :

sommet 1	-1,1,1
sommet 2	1,1,1
sommet 3	1,-1,1
sommet 4	-1,-1,1
sommet 5	-1,1,-1
sommet 6	1,1,-1
sommet 7	1,-1,-1
sommet 8	-1,-1,-1

La valeur des coordonnées est uniformisée. Elle est toujours comprise entre -1 et 1.

Les rotations 3D s'appliquent sur les 3 angles :

- l'angle des x
- l'angle des y
- l'angle des z

Ils seront exprimés en radian dans cet exercice. On utilisera une rotation basique : $(3.14 * 2)/360$ à chaque itération sur chaque angle.

Le calcul de rotation s'effectue en 3 étapes :

- une première étape qui calcule la « matrice » de rotation de l'espace 3D en tenant compte de l'état des angles X,Y,Z avec les différentes rotations déjà effectuées.
- une seconde étape qui va appliquer à chaque point les transformations nécessaires en tenant de la « matrice » précédemment calculée.
- une dernière étape qui appliquera la projection dans l'espace en divisant les valeurs obtenues en X,Y,Z de chaque point par un point de fuite virtuel

NB : La « matrice » de rotation ne doit pas être comprise dans le sens de la représentation de l'objet mathématique

1. Analyser le code

Le template de code « rotations3d.c » utilise l'ensemble de ces principes pour faire tourner 8 sprites qui forment les sommets d'un cube.

Repérer le calcul de la « matrice » de rotation

Essayer de :

- changer les valeurs des points pour changer la forme
- changer la valeur du point de fuite
- changer l'incrémentation des angles de rotations

Le template utilise 4 tailles différentes de sprite. Il définit la bonne texture pour le sprite selon la valeur du Z calculé pour chaque point.

Vous observerez un problème dans l'affichage des sprites. Expliquez.

Quelle solution possible ? Implémentez-la.

Peut-on faire mieux ?