Matrices 4x4 et transformations

Pour que le code du TP3 fonctionne, nous aurons besoin de certaines des fonctions implémentées lors du TP1 et TP2. Si vous n'avez pas terminé le TP1 et TP2, il est conseillé de le terminer avant de commencer le TP3.

Le code est ici et un simple maillage d'un cube ici.

Exercice 1

1. Importer des fonctions des TP précédents.
2. Réimplémentation : mise à l'échelle, rotation, projection orthographique avec des matrices 4 x 4.
3. Trouver la fonction : translation_matrix et implémenter la traduction.
4. Trouver la fonction : perspective_projection_matrix et implémenter la projection en perspective.
5. Analyser la fonction : apply_transformations_homogeneous et expliquer son implémentation.

Exercice 2

Dans cet exercice, nous allons mettre en œuvre la mise en orbite de cubes autour du cube central qui peut lui-même se déplacer et d'autres cubes avec lui. Veuillez regarder la vidéo.

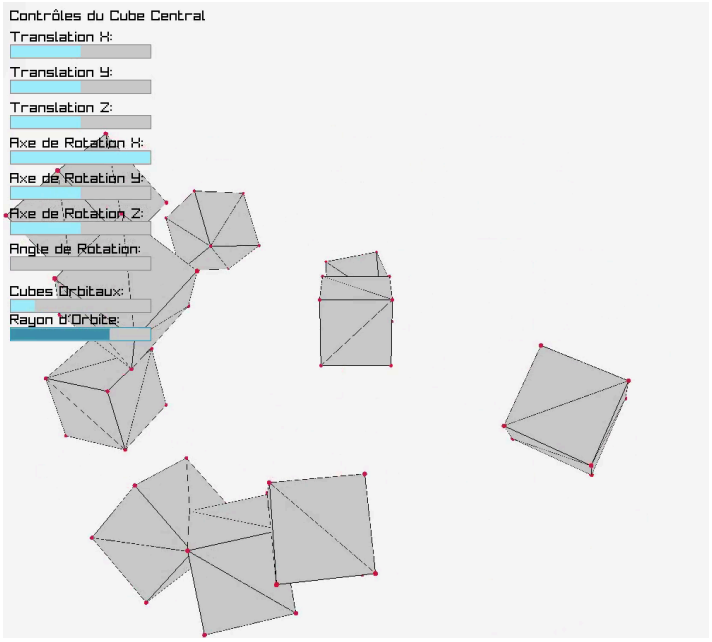

Fig. 1. Extrait de la vidéo, cliquez pour en voir plus.

Le code est ici

1. Implémentez les parties manquantes du code comme indiqué par « TODO : »
2. Expliquer l'ordre des transformations.
3. Modifiez le code de manière que chaque cube en orbite ait une taille différente (aléatoire) et soit à un rayon aléatoire par rapport au cube central.
4. Ajoutez un plan sous le cube central à une distance constante et utilisez la projection orthographique pour projeter tous les cubes sur le plan
5. Utilisez la projection en perspective pour projeter tous les cubes sur un plan

Exercice 3

Le code est ici

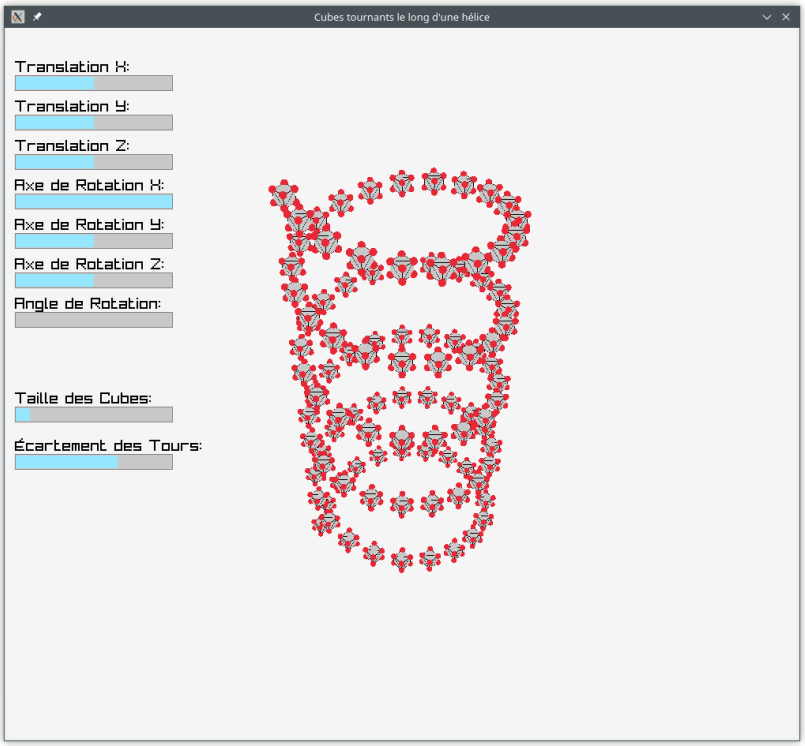

Fig. 2. Visualisation de l'hélice.

1. Trouvez la ligne : num_cubes = int(num_turns * 20) et paramétrez le nombre de cubes par tour.
2. Faites tourner les cubes en synchronisation, chacun doit être tourné un peu plus que le précédent.
3. Faites de même avec l'échelle. Les cubes doivent varier du plus petit au début de l'hélice au plus grand.
4. Implémenter d'autres types de courbes paramétriques, par exemple des nœuds.

Fig. 3. Noeud 3 2

Nœud 3,2 (Nœud de trèfle)

- $x(t) = \sin(3t)$
- $y(t) = \cos(2t)$
- $z(t) = \sin(2t)$

Nœud 4,1 (Nœud en huit)

- $x(t) = (2 + \cos(2t)) \cdot \cos(3t)$
- $y(t) = (2 + \cos(2t)) \cdot \sin(3t)$
- $z(t) = \sin(4t)$

Nœud 4,3

- $x(t) = \cos(2t) \cdot (2 + \cos(3t))$
- $y(t) = \sin(2t) \cdot (2 + \cos(3t))$
- $z(t) = \sin(5t)$

Nœud 5,1 (Nœud de Solomon ou Nœud de Cinquefoil)

- $x(t) = (2 + \cos(5t)) \cdot \cos(2t)$
- $y(t) = (2 + \cos(5t)) \cdot \sin(2t)$
- $z(t) = \sin(5t)$

Exercice 4

Dans cet exercice, nous allons implémenter un bras robotique qui peut se déplacer à différents niveaux comme démontré dans le film. Le code est ici.

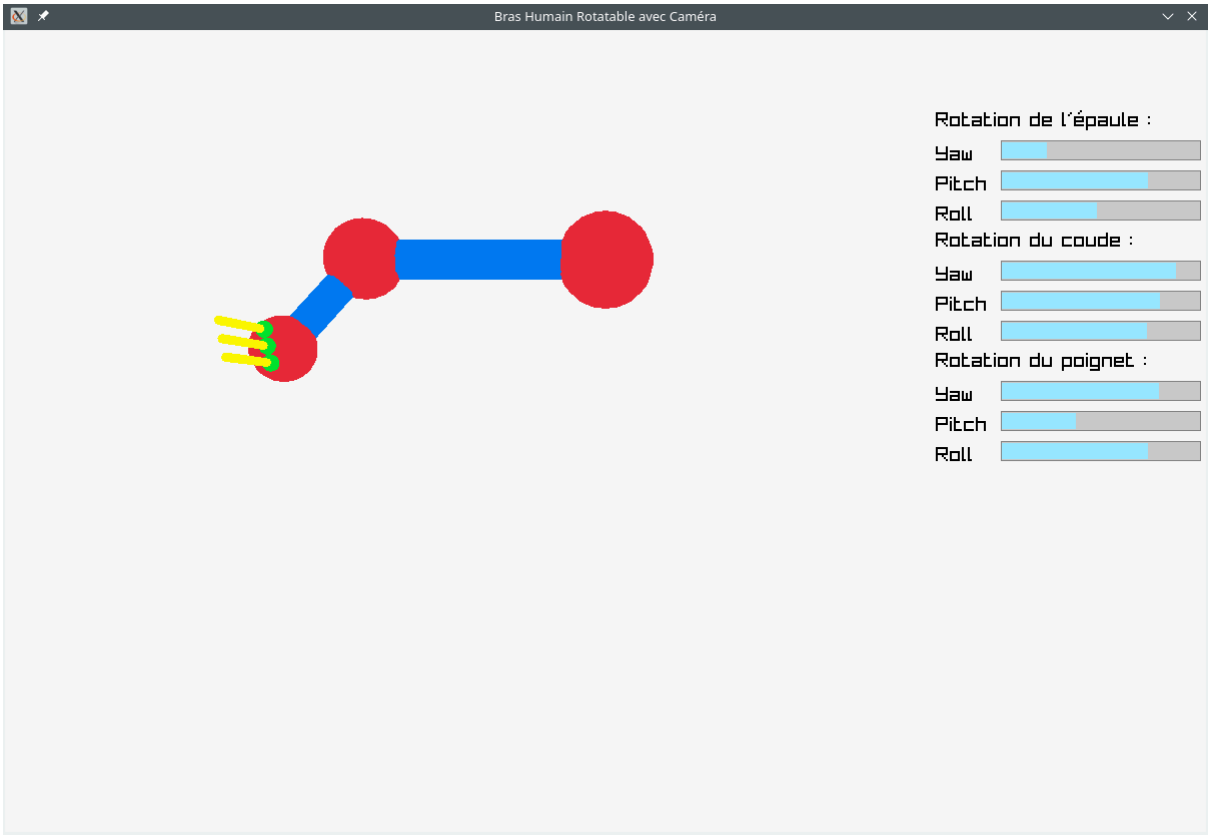

Fig. 4. Bras rotatable

1. Trouvez la fonction : rotation_matrix_yaw_pitch_roll et implémentez les rotations d'angle d'Euler.
2. Trouvez la fonction : apply_transformation_to_segment et implémentez les rotations correctes autour d'un point arbitraire en déplaçant le point depuis le reference_point vers l'origine, puis en revenant.
3. Dans la fonction apply_transformation_to_segment, composez correctement les transformations qui doivent être exprimées dans la matrice transformation_matrix.

4. Trouvez la fonction : `calculate_fingers_positions`, et utilisez la fonction `apply_rotation` pour faire pivoter correctement les points.
5. Trouvez dans la fonction `main` : « TODO : déverrouiller la vérification une fois le code implémenté. » et déverrouillez les assertions. Si tout s'est bien passé, vous devriez maintenant être en mesure de faire pivoter le bras.
6. Pouvez-vous améliorer la façon dont les doigts sont implémentés ?