Session 15 : Cinématique Inverse (IK)

> **FK vs IK**
>
> Jusqu'à présent, nous avons pensé en **FK (Forward Kinematics)** : "Je tourne l'épaule de 30°, donc le coude bouge, donc la main avance." C'est facile à calculer ($P_{\text{enfant}} = M_{\text{parent}} \times P_{\text{local}}$) mais difficile à contrôler.
>
> L'**IK (Inverse Kinematics)** pose le problème inverse : "Je veux que la main soit à la position (x, y, z). Comment dois-je tourner l'épaule et le coude ?"

1. Le Problème mathématique

Une chaîne cinématique est composée d'os (segments rigides) et d'articulations (joints). Pour un bras humain, on veut atteindre une cible (Target) avec l'effecteur final (End Effector).

- **Cas simple :** Un bras à 2 segments en 2D. On peut le résoudre par trigonométrie (Théorème d'Al-Kashi / Loi des cosinus).
- **Cas complexe :** Un tentacule à 10 segments en 3D. La trigonométrie devient infernale. Il existe une infinité de solutions possibles.

Comme pour le solveur physique de la Session 14, nous allons utiliser une **approche itérative**.

2. Algorithme 1 : CCD (Cyclic Coordinate Descent)

C'est l'algorithme "robotique". Il est très simple à comprendre et à coder.

Principe : On part du bout de la chaîne (le poignet) et on remonte vers la racine (l'épaule). Pour chaque articulation, on la fait tourner pour aligner l'effecteur final vers la cible.

1. On regarde le dernier os. On le tourne pour qu'il pointe vers la cible.
2. On regarde l'avant-dernier os. On le tourne pour que le bout de la chaîne pointe vers la cible.
3. On continue jusqu'à la racine.
4. On répète la boucle jusqu'à ce que l'erreur soit faible.

Avantage : Très stable, permet de mettre des limites d'angles (constraints) facilement. **Défaut :** Donne un mouvement un peu raide, robotique.

3. Algorithme 2 : FABRIK

Forward And Backward Reaching Inverse Kinematics. C'est l'algorithme utilisé dans la plupart des jeux modernes (Unity, Unreal) car il est rapide et visuellement naturel. Il ne travaille pas avec des angles, mais avec des **positions**, exactement comme notre moteur Verlet (Session 10) !

L'algorithme en 2 passes :

1. **Backward (De la cible vers la base) :**
   - On place l'effecteur final **sur** la cible.
   - On tire le point précédent pour qu'il soit à la bonne distance (longueur de l'os).
   - On remonte jusqu'à la racine.
   - (Problème : la racine s'est détachée du corps !)
2. **Forward (De la base vers la cible) :**
   - On remet la racine à sa position d'origine (sur l'épaule).
   - On tire le point suivant pour qu'il soit à la bonne distance.
   - On redescend jusqu'à l'effecteur.

En répétant ces deux passes, la chaîne se tend naturellement vers la cible.

4. IK et Physique : L'Animation Procédurale

C'est ici que la magie opère. Comment mélanger Rapier (Physique) et IK (Animation) ?

Le placement des pieds (Foot IK). *Dans un jeu, un personnage court sur un terrain accidenté.*

1. ***Raycast :*** *On lance un rayon physique depuis le genou vers le bas pour trouver le point d'impact exact du sol (via le `QueryPipeline` de la session 11).*
2. ***Cible IK :*** *On définit ce point d'impact (+ décalage semelle) comme la cible IK du pied.*
3. ***Résolution :*** *On utilise FABRIK pour plier le genou et la cheville afin que le pied touche ce point.*

> **Active Ragdolls**
>
> Pour aller plus loin (ex: **Gang Beasts**, **Human Fall Flat**), on n'utilise pas l'IK pour bouger le maillage visuel. On utilise l'IK pour calculer la **pose cible**, puis on utilise des Moteurs Physiques (`JointMotors`) sur le Ragdoll pour qu'il essaie d'atteindre cette pose en utilisant de la force physique.
>
> Cela permet au personnage de trébucher, d'être poussé, tout en essayant de se relever.

5. Exemple : La jambe d'araignée

Vous pouvez regarder cette vidéo:

https://www.youtube.com/watch?v=Ihp6tOCYHug&t=393s

L'objectif est de coder une patte à 3 segments.

- Utiliser un **Raycast** Rapier pour trouver le sol.
- Coder l'algorithme **FABRIK** (ou analytique 2 os) pour calculer la position des jointures.
- Mettre à jour les positions des Meshes Three.js (Kinematic) pour visualiser la patte.