TD Animation 3D

Master I Informatique M. AMMI

Université Paris-11 2008-2009

L'objectif de ce TD est s'initier aux techniques de cinématique inverse et à l'animation par Keyframing.

L'approche étudiée dans le volet cinématique inverse est une solution approche analytique. Nous développerons une approche à base de Jacobienne dans le prochain TD.

Avant de commencer le traitement des exercices, il est demandé d'examiner et de commenter le code source fourni.

Exercice 1 : cinématique directe

- Construire, avec les composants hiérarchiques d'OpenGL, un bras de robot comprenant 2 segments : UpArm & LowArm. (dans la fonction DrawScene).
    - Le bras comprend 2 DLL en rotation suivant l'axe Z
    - Les transformations des différents segments doivent être gérées par l'intermédiaire de la structure t_Bone (rot & trans). Il faut donc compléter cette structure avant de l'utiliser.
    - Les dimensions des segments sont définies par les variables UpArm_Length, LowArm_Length, Arm_Width, Arm_Depht.
- Compléter la fonction InitBonesystem() pour l'initialisation de la configuration géométrique du robot.
- Compléter la fonction de gestion de la souris « PorcessMouseActive() »pour pouvoir manipuler directement les orientations des deux segments
    - Bouton droit pour l'orientation
- Améliorez les fonctions précédentes de gestion des angles (dans « PorcessMouseActive() ») pour avoir un comportement d'interaction cohérent : lorsqu'on clique de nouveau sur l'écran, la nouvelle orientation du segment est calculée à partir de la configuration précédente, sans avoir de sauts.
- Placer un axe (voir liste AXIS_DLIST) sur l'extrémité de chaque segment.

Exercice 2 : cinématique inverse

Sur la base du système réalisé précédemment, nous allons mettre en place une fonction « ComputeIK(~) » permettant de manipuler le robot à partir de son point terminal.

- Décrire la solution analytique
    - Extraire les fonctions exprimant la valeur des angles en fonction de la position de l'effecteur final.
- Proposez un algorithme puis une implantation C pour gérer la cinématique inverse.

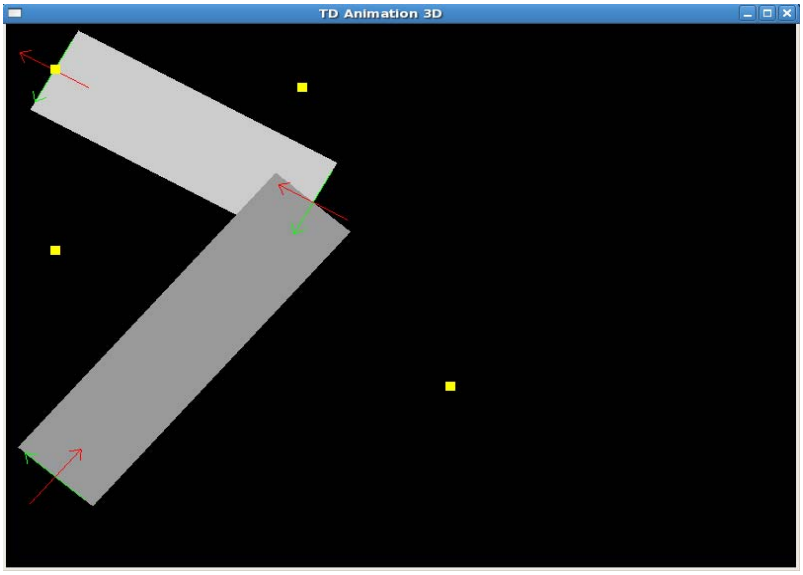
Exercice 3 : Animation par Keyframing

Interpolation linéaire

- Proposez une structure de données pour gérer les instants clés « Key ».
- Initialisez 5 moments comme suit **(t,x,y)**: (0.0,50,50), (100.0,300,70), (200.0,450,400), (400,50,250), (500,50,50).
    - Afficher les positions des points clés avec des points de taille 5.0. Il faut compléter et utiliser la fonction « Draw_animation_sequence() ».
- Proposez un algorithme pour générer une interpolation linéaire entre les points clés.
- Proposez une implantation pour l'algorithme proposé dans la fonction « SolveLinear(float t, int x, int y, int z) ».
- Exploitez la fonction « Idle() » pour jouez l'animation.
- Quelles observations pouvez-vous faire sur le mouvement du robot ?

Interpolation non linaire (Hermite) :

- Sur la base des explications données en annexe, sur l'interpolation de type Hermite, proposez un algorithme pour réaliser l'animation.
- Introduire les paramètres supplémentaires de l'interpolation Hermite dans la structure « Key » pour gérer a continuité, la tension la pente de la courbe au niveau du point.
- Proposez une implantation de cet algorithme dans la fonction « SolveTCB(float t, int x, int y, int z) ».
- Implanter toutes les méthodes requises par cette fonction (H0, H1, etc.).



Annexe :

L'interpolation de type Hermite cubique se fait à l'aide de 4 polynômes de degré 3 notés H0, H1, H2, H3. Nous interpolons toujours entre 2 clés, appelons Pn la valeur de la clé de départ, et Pn+1 celle de la clé d'arrivée. Tn et Tn +1 représentent respectivement la tangente à la clé Pn et celle sur la clé Pn+1.

La formule d'interpolation d'Hermite est la suivante :

Les polynômes (appelés « Blending Hermite functions ») :

H0 = 2t3 - 3t2 + 1

H1 = -2t3 + 3t2

H2 = t3 - 2t2 +t

H3 = t3 - t2

Les tangentes :

$$T_n = \frac{(1-T)(1+B)(1-C)}{2} * (P_n - P_{n-1}) + \frac{(1-T)(1-B)(1+C)}{2} * (P_{n-1} - P_n)$$

et

$$T_{n+1} = \frac{(1-T)(1+B)(1+C)}{2} * (P_{n-1} - P_n) + \frac{(1-T)(1-B)(1-C)}{2} * (P_{n+2} - P_{n+1})$$

Ce qui nous permet de calculer P(t) :

P(t) = H0(t)Pn + H1(t)Pn+1 + H2(t)Tn + H3(t)Tn+1.

Explication de l'influence des paramètres TCB :

- T : tension, sert comme son nom l'indique à régler la tension de la courbe, plus T sera proche de 1 plus on se rapprochera d'une ligne droite entre les 2 clés, plus T sera proche de –1 plus la courbe sera lâche. On peut voir cela comme quelqu'un qui tire sur une corde (valeur 1) ou la relâche (valeur –1).
- C : continuity, sert pour définir comment le changement de vitesse et de direction interviendra entre le moment ou l'on entre sur une clé et le moment ou l'on part de cette clé.
- B : bias, définit la direction de la courbe dès qu'elle passe la clé.

Il y a des cas particuliers des splines TCB lorsque :

- T = C = B = 0, on appelle cette interpolation « spline de Catmull-Rom ».
- B = C = 0, on l'appelle « spline Cardinal ».
- T = C = 0, on l'appelle « Bias controlled spline ».
- T = B = 0, on l'appelle « Continuity controlled spline ».