Cinématique inverse

L'objectif de ce TP est de vous faire mettre en œuvre l'algorithme de CCD (Cyclic Coordinate Descent) pour calculer une cinématique inverse sur une chaine poly-articulée.

Contexte. Dans les différentes classes qui vous sont fournies, la classe *Animation::KinematicChain* vous permet de décrire une chaine cinématique et d'effectuer des calculs sur cette dernière.

Cette classe met à votre disposition deux classes internes importantes :

- *Animation::KinematicChain::DegreeOfFreedom*. Il s'agit une classe de gestion d'un degré de liberté. Cette classe vous permet d'accéder à un degré de liberté de la chaine cinématique comme s'il s'agissait d'un flottant standard (les opérateurs de conversion et affectation ont été redéfinis). Lorsque la valeur d'un degré de liberté est modifiée, l'instance de *KinematicChain* à laquelle il est lié est automatiquement mise à jour. Bien sûr, les degrés de liberté connaissent leur intervalle de validité et ce dernier est automatiquement géré.
- *Animation::KinematicChain::Node*. La classe de chaine cinématique possède en interne, une représentation similaire à celle d'un graphe de scène. Cette représentation permet de décrire la structure d'un squelette sur lequel des calculs doivent être effectués. Cette classe est la classe mère de tous les nœuds du squelette géré par la chaine cinématique et possède toutes les méthodes vous permettant d'interagir le squelette : une méthode permettant de récupérer les degrés de liberté associé au nœud (*getDOF*), une méthode permettant de collecter tous les degrés de liberté depuis la racine jusqu'à ce nœud (*collectDegreesOfFreedom*), une méthode permettant de récupérer la transformation locale (*getLocalTransformation*) et une méthode permettant de récupérer la transformation globale (*getGlobalTransformation*). Vous pouvez consulter la documentation pour avoir plus d'information sur ces méthodes.

La classe *Animation::KinematicChain* vous permet de décrire la structure d'un squelette via des méthodes d'ajout de transformations. Ces méthodes se décomposent en deux groupes : l'ajout de transformations dynamiques (possédant des degrés de liberté) et l'ajout de transformations statiques (ne possédant pas de degré de liberté).

- Les transformations statiques permettant de décrire des transformations ne pouvant pas être modifiées par le moteur d'animation : les méthodes addStaticTranslation et addStaticEulerRotation vous permettent d'ajouter ce type de transformation.
- Les transformations dynamiques permettant de décrire des transformations pouvant être modifiées par le moteur d'animation: les méthodes addDynamicTranslation, addDynamicEulerRotation et addDynamicRotation permettent d'ajouter ce type de transformation.

La classe *Animation::KinematicChain* met aussi à votre disposition la méthode *derivate* permettant de calculer la dérivée de la position d'un nœud par rapport à un degré de liberté fourni. Cette méthode vous sera utile pour la suite du TP (tout comme ce qui est décrit ci-dessus).

Question 1 : Dans la plateforme fournie, créez une nouvelle classe d'application TP2_siaa (intégrez la gestion de la caméra et des touches) et enregistrez cette application dans le programme principal.

Question 2 : Dans la classe d'application, créez une méthode permettant d'ajouter une chaine poly-articulée au graphe de scène. La chaine poly-articulée sera représentée par une suite de cylindres de 0.1 de rayon et 0.5 de hauteur reliés par des articulations symbolisées par des sphères de rayon 0.2. Chaque articulation comportera une rotation suivant l'axe X et une rotation suivant l'axe Z. Cette méthode devra prendre en paramètre le nombre de segments constituant la chaine. Une fois cette méthode implémentée, testez l'affichage de votre chaine.

Question 3 : Modifiez le code de la fonction précédente pour initialiser, en parallèle de votre construction de graphe de scène une instance de *Animation::KinematicChain.* Vous veillerez aussi à conserver, dans une structure de données, la correspondance entre le nœud du graphe de scène et le degré de liberté (*Animation::KinematicChain::DregreeOfFreedom*) lui correspondant. Identifiez bien les éléments dynamiques et statiques du squelette à animer. Les angles de rotation devront pouvoir varier dans l'intervalle $[-\pi/2; \pi/2]$.

Question 4 : Soit $P \in R^3$ la position de l'extrémité de la chaine poly-articulée. *P* est donc fonction des angles $(\theta_1, \dots, \theta_n)$ telle que $P = f(\theta_1, \dots, \theta_n)$ en considérant que la chaine comporte *n* rotations. Exprimez dP en fonction de $d\theta_i$. Déduisez en l'expression de $d\theta_i$en fonction de dP.

Question 5 : Implémentez l'algorithme du CCD sous la forme d'une classe. Cet algorithme devra être paramétré par un pointeur sur l'instance de *Animation::KinematicChain* et par le nœud extrémité sur lequel le calcul de CCD devra être effectué. Dans un premier temps, implémentez la méthode *convergeToward* permettant d'effectuer une itération de convergence du CCD sur tous les degrés de liberté de la chaine concernée. Cet algorithme devra prendre en paramètre un vecteur correspondant à un décalage par rapport à l'extrémité de la chaine articulée ainsi qu'un flottant correspondant à la variation angulaire maximale applicable lors d'une itération.

Question 6 : Visualisez le résultat de vos calculs en générant une cible aléatoire (symbolisée par une sphère rouge dans l'application) et en affichant à chaque image le résultat du calcul d'une itération de l'algorithme de CCD. Observez la convergence. A des fins de test, vous pourrez configurer la touche 'n' afin que l'appui sur cette dernière génère une nouvelle cible aléatoire.

Question 7 : Dans la classe de CCD, ajoutez une méthode *solve* calculant la solution au problème de cinématique inverse. Cette méthode reverra un booléen signalant la réussite ou l'échec de la convergence et prendra les mêmes paramètres que la méthode *convergeToward*.

Question 7 : Proposez une solution utilisant l'interpolateur du TP précédent pour décrire la trajectoire de l'extrémité de la chaine poly-articulée et ainsi mieux contrôler le mouvement du bras poly articulé.

Question 8 (subsidiaire): Proposez une solution permettant de gérer l'animation de la chaine poly-articulée en interpolant entre la posture courante et la posture calculée par CCD (réfléchissez bien, cette solution pourrait vous conduire à légèrement modifier votre graphe de scène).