TP Synthèse d'images: Skinning
CPE

durée - 4h

1 Géométrie

Notez le chargement des données tels que *mesh cat* initialisés dans la fonction *load model* de la classe *scene*.

Question 1

Affichez le maillage du chat dans la fonction draw scene *à l'intérieur de la condition* show bind pose *en suivant les indications dans le code.*

Il s'agit du maillage du chat dans la position de repos (bind pose).

Lors de ce TP nous allons déformer ce chat grace à un squelette d'animation et à des poids de dépendances entre les sommets du chat et les os du squelette. On appelle cette approche *skinning*, ou plus précisément : Linear Blend Skinning (LBS).

2 Animation du squelette

La première étape d'un *skinning* consiste à définir un squelette ainsi qu'une animation associée. Un squelette est formé d'une hierarchie de repères (position et orientation) appellés joints. On appelle *os* le segment liant un repère donné à son repère parent.

Pour définir de manière aisé une déformation sur le squelette, il est commun d'encoder la position et l'orientation d'un repère dans la base locale de son repère parent. L'animation d'un squelette peut donc être encodée en tant qu'une suite de position/orientation exprimées dans le repère local au cours du temps.

On désignera donc par **repère global** la base canonique standard. Et par **repère local** la base du joint parent. Les coordonnées globales et locales seront les coordonnées exprimées respectivement dans ces bases. On posera que le parent du tout premier joint (joint n'ayant pas de père dans la hierarchie) est le repère global de la base canonique.

On appellera *frame* l'une des position clée stockée dans la classe animation. Si cette classe encode N positions et orientations, on aura une animation comportant N frames différentes.

- La classe *skeleton* stocke la structure du squelette (connectivitée). Les positions et orientations définissent les transformations du repère globale vers les repères locaux de chaque os dans la position de repos (inverse bind pose).
- La classe *animation* stocke les positions et orientations de chaque os au cours du temps. Les coordonnées sont exprimées dans le repère local (de l'os parent). La classe *animation* socke autant de positions et orientations qu'il y a de *frame* définie.

2.1 Prise en main

Question 2 *Observez la classe* skeleton *et* animation.
Observez où sont initialisées ces classes, à partir de quels fichiers ? Observez ces fichiers.

2.2 Construction d'un squelette

Dans la classe *scene*, nous allons chercher à construire une fonction qui, pour une frame donnée, viendra remplir les deux structures

— bone positions

— bone orientations

par les positions et orientations de chaque joint exprimées dans le repère **global**.

Notez que si l'on appelle G_j la matrice exprimant la base d'un joint j dans le repère global, L_j la matrice exprimant la base d'un joint j dans le repère locale, et $p(j)$ l'indice du joint parent à j, on a la relation suivante :

$$\mathrm{G}_j = \mathrm{G}_{p(j)}\, \mathrm{L}_j \ .$$

On peut représenter ces matrices par blocs :

$$\mathrm{G}_j = \left(\begin{array}{c|c} q_G^j & t_G^j \\ \hline 0 & 1 \end{array}\right) \quad \mathrm{L}_j = \left(\begin{array}{c|c} q_L^j & t_L^j \\ \hline 0 & 1 \end{array}\right) \ ,$$

avec t et q respectivement les parties de translations et de rotations associées à ces matrices. t_G^j représente la position du joint j exprimée dans le repère global. t_L^j la position du joint j exprimée dans le repère local. q_G^j représente l'orientation (quaternion ou matrice) du joint j dans le repère global, et respectivement q_L^j pour le repère local.

Question 3 *Donnez l'expression de t_G^j et q_L^j en fonction de $t_L^j, t_G^{p(j)}, q_L^j, q_G^{p(j)}$.*

Notez que l'on exprimera les rotations q sous forme de quaternion. On s'assurera donc d'avoir à tout moment $\|q_G^j\| = 1$.

Pour implémenter ces relations, nous avons à disposition les appels suivants :

— `cat_skeleton->number_of_bones();` : permet d'accéder au nombre total d'os du squelette d'animation.

— `cat_animation->position(frame,index);` : permet d'accéder à la position (locale) de l'os *index* à la *frame* donnée (t_L^{index}).

— `cat_animation->orientation(frame,index);` permet d'accéder à l'orientation (locale) de l'os *index* à la *frame* donnée (q_G^{index}).

— `cat_skeleton->parent(index);` permet d'accéder à l'indice de l'os/joint parent de celui désigné à l'*index* courant ($p(index)$). Notez que dans les données fournies, nous avons toujours `parent(index)<index`.

Question 4 *Implémentez la fonction qui viens compléter les positions et orientations de chaque joints dans le repère global. (À réaliser dans la fonction* create skeleton *en suivant les indications dans le code).*

Remarque : Pour appliquer une rotation issue d'un quaternion q sur une classe `vec3` p, on utilisera l'appel `q.rotate(p)`.

Question 5 *Une fois mis en place, affichez le squelette du chat pour une frame donnée. Vous devez obtenir un résultat proche de celui de la figure 1. (Notez les indications dans le code dans la partie* show skeleton*).*

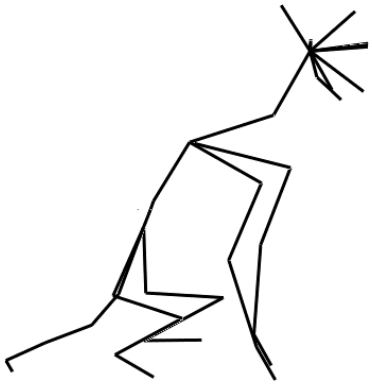

FIGURE 1 – Exemple de squelette du chat.

2.3 Animation du squelette

Dans la fonction *draw scene*, la variable *current frame* indique la frame courante.

Question 6 *Au cours du temps, calculez et affichez le squelette pour des frames évoluants au cours du temps.*

Question 7 *Indiquez ce que vous observez lorsque la vitesse d'évolution est trop lente.*

2.4 Interpolation de squelette (optionnel)

La variable *local time* indique l'écart relatif du temps courant par rapport au temps correspondant à la frame précédente. Par exemple, *local time* de 0.25 indique que l'on est à $3/4$ sur la frame courante, et à $1/4$ sur la frame suivante.

En se servant des variables *current frame* et *local time*, il est possible d'en déduire la position position interpolée du squelette au cours du temps.

Question 8 (optionnel) *Modifiez la fonction de création du squelette pour que la fonction recoive cette fois en paramètre la frame courante, la frame précédente, et le* local time*. Le squelette doit alors être interpolé continuement entre chaque frame. On interpolera les positions linéairements, et les orientation à l'aide d'un SLERP de quaternions. Observez cette fois que le squelette se déplace continuement au cours du temps.*

3 Géométrie

3.1 Rappel de skinning

Soit T_j la matrice 4×4 de la base d'un joint j donné pour la frame courante. Soit B_j la matrice 4×4 de la base d'un joint j donné pour la position de repos (bind pose). Soit $\mathbf{p} = (x, y, z)$ les coordonnées du maillage dans la pose de repos exprimée dans la base globale.

La position $\mathbf{p}_2$ des coordonnées du maillage déformé pour la frame courante est donnée par :

$$\mathbf{p}_2 = T_j \, B_j^{-1} \mathbf{p} \, .$$

Rem. B^{-1} peut être vue comme la matrice de passage des coordonnées globales (dans la pose de repos) vers des coordonnées locales. T est respectivement vue comme la matrice de passage des coordonnées locales vers globales (dans la pose déformée).

Dans le cas d'un skinning lisse (coordonnées dépendant de plusieurs joints), on aura alors

$$\mathbf{p}_2 = \sum_{j \in \mathcal{J}} \omega_j T_j \, B_j^{-1} \mathbf{p} \, ,$$

avec $\mathcal{J}$ contenant l'ensemble des indices des os dont dépend la position $\mathbf{p}$, et ω_j les poids de skinning correspondants.

Les positions et orientations des matrices T_j sont stockés dans les vecteurs `bone position` et `bone orientation`.
Les positions et orientations des matrices B_j^{-1} sont accessibles depuis les appels `cat skeleton->position(i)` et `cat skeleton->orientation(i)`. Notez que l'inverse est déjà calculé.

3.2 Pré-multiplication par la pose de repos inverse

— Soit $t_{B^{-1}}$ la translation associée à la matrice B^{-1}.
— Soit $q_{B^{-1}}$ la rotation associée à la matrice B^{-1}.
— Soit t_T la translation associée à la matrice T.
— Soit q_T la rotation associée à la matrice T.

Question 9 *Quelle est la translation t et rotation q associée au produit matriciel $T\,B^{-1}$? Cette opération est réalisée dans la fonction* apply inv bind pose*.*

3.3 Déformation des sommets

La méthode *draw skinned* du fichier *opengl drawer* appelle la fonction *skin vertices* qui viens déformer les coordonnées du maillage avant l'affichage.

Question 10 *Observez la struct* Skinning *de la classe* mesh*. Que contient ses informations ? Pourquoi a t-on 6 valeurs dans la struct ? Comment peut-on récupèrer les indices des os dont dépend le sommet i ? Et quels sont les poids de skinning associées ?*

Question 11 *Complétez la fonction* skin vertices *de manière à calculer la déformation pour chaque sommets du maillage. Affichez le maillage résultant en décommentant les lignes spécifiées dans le code.*

Question 12 *Observez l'influence des poids de skinning en n'en considérant qu'une sous partie (n'oubliez pas de normaliser la somme des poids à 1 pour chaque sommet).*

Question 13 *Affichez en couleur l'intensité des poids de skinning pour un os donné (ou d'autres informations pertinentes) sur les sommets du maillage.*