TP Synthèse d'images: Deformation de surfaces
- Skinning -
CPE

durée - 6h

06-07 Janvier 2010

FIGURE 1 – Exemple d'animation skinnée.

1 But du TP

Ce TP consiste à mettre en place une deformation par skinning. On interpolera pour cela à la fois le squelette d'animation ainsi que les positions des sommets dans les repères locaux. Dans un premier temps la deformation sera réalisée entièrement sur CPU. Puis dans un second temps, celle-ci sera mis en place sur GPU en GLSL.

2 Prise en main de l'environnement

Dans l'ensemble du TP, on considèrera que chaque sommet du maillage dépend toujours de 4 os.

2.1 Chargement du modèle

Le modèle animé de personnage se décompose en 2 parties :

1. Le maillage de base que l'on trouvera dans le fichier *data skinning/mesh/body.basemesh*. Il s'agit d'un format similaire au format OFF, mais complexifié avec des informations de skinning et de textures. Les caractéristiques sont illustré en fig. 2.

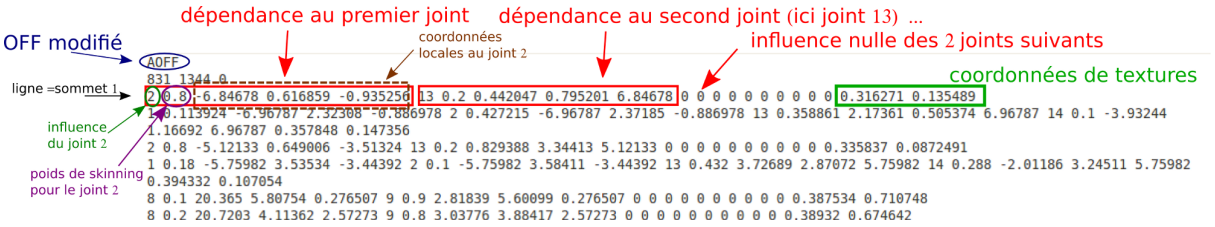

FIGURE 2 – Exemple du format AOFF : Format off classique + informations de skinning (indice du joint, poids de skinning, coordonnées xyz locales) pour 4 joints suivit des coordonnées de textures.

2. Les animations des joints au cours du temps. Plusieurs animations sont disponibles dans le repertoire *data skinning/anim/* sous forme de fichiers lisibles. Le modèle possède 62 joints (un joint par ligne). La position clé de chaque joint est enregistrée sous forme (position xyz, quaternion xyzw). Chaque frame de chaque position clé est ainsi lisible toute les 62 lignes. Le nombre total de frames étant indiqué en début de fichier.

Question 1 *Observez la structure de ces deux types de fichiers et retrouvez les éléments mentionnés.*

Le chargement de ces fichiers se déroule une unique fois au début du programme dans le fichier *main* dans la fonction *load model* (voir fig. 3).

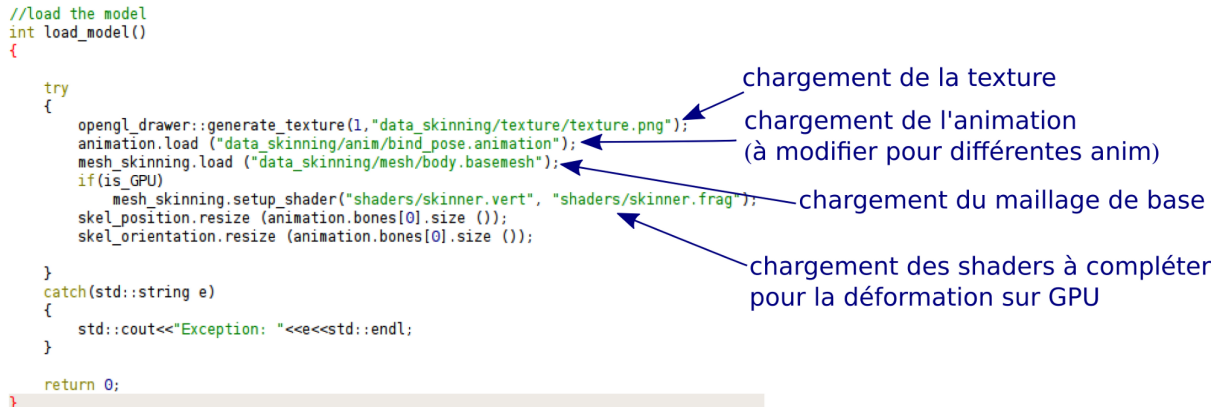

FIGURE 3 – Fonction de chargement des modèles exécutée une unique fois au lancement.

2.2 Boucle d'affichage

L'interpolation du squelette d'animation, la déformation par skinning ainsi que l'affichage du personnage se réalise dans la boucle d'affichage principale. Il s'agit de la fonction *draw scene* dans le fichier *main*.

Son déroulement est illustré en fig. 4

3 Interpolation du squelette

Le squelette d'affichage est géré par la classe *animation*. La position et l'orientation de chaque joint pour chaque frame est stocké dans la variable *bones*. Le joint numéro k_j de la frame k_t est accessible par la syntaxe : animation.bones[k_t][k_j].

Pour chaque joint, on traitera séparément l'orientation (quaternion) et sa position (vecteur de coordonnées xyz).

Pour interpoler le mouvement du squelette linéairement il faut définir la frame courant et la suivante.

```
//Draw the scene
int draw_scene()
{
    glEnable(GL_POLYGON_OFFSET_FILL);
    glPolygonOffset(5,5);


    static double current_time = 0;
    static double last_time = 0;
    static double info_last_time = 0;

    current_time = glutGet (GLUT_ELAPSED_TIME) / 1000.0;
    info_last_time += current_time - last_time;
    last_time = current_time;
    if (info_last_time >= 1.0 / FRAMERATE)
    {
        info_last_time -= 1.0 / FRAMERATE;
    }

    interpolate_skeletons (animation.bones[0],animation.bones[1],0);

    glEnable(GL_LIGHTING);
    glColor3d(0.7,0.7,0.7);
    //glEnable(GL_TEXTURE_2D);

    if(is_GPU)
    {
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        mesh_skinning.draw (skel_position, skel_orientation, true);
    }
    else
        mesh_skinning.draw_cpu (skel_position, skel_orientation, true);

    glDisable(GL_TEXTURE_2D);
    glDisable(GL_LIGHTING);


    return 0;
}
```

fonction appelée en permanence

condition executée toute les 1/FRAMERATE s

Interpolation du squelette d'animation entre 2 positions clés (à compléter)

Variable globale: utilise-on le GPU ?

skinning sur GPU et affichage par appels des shaders (à completer)

skinning sur CPU puis affichage du résultat (à compléter)

FIGURE 4 – Fonction d'affichage.

Question 2 *Définissez 2 variables :* frame current *et* frame next *contenant les numéros de frames pour lesquels on souhaite intepoler le squelette. Afin d'avoir une animation identique indépendamment de la vitesse de raffraichissement, on incrémentera ces compteurs au maximum 25 fois par secondes. On pourra notamment se servir de la condition sur la variable* FRAMERATE.

Question 3 *Testez l'animation du squelette en modifiant l'appel à* interpolate skeletons. *Envoyez cette fois les joints à interpoler ainsi que le ratio en temps entre les deux instants clés (entre 0 et 1). Note : On prendra soin de considérer une animation ayant plus de 1 frame.*
Le maillage doit alors s'animer de manière saccadé (la fonction d'interpolation sera à compléter par la suite) et comparable à ce que l'on peut voir en fig. 5.

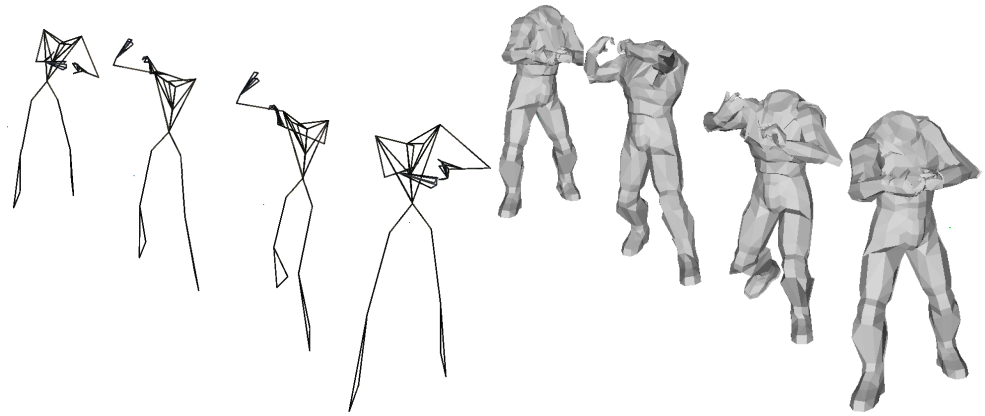

FIGURE 5 – Animation du squelette ainsi que du personnage par skinning rigide.

Question 4 *Complétez la fonction* interpolate skeletons *pour réaliser cette fois l'interpolation linéaire des positions, et l'interpolation des orientation par SLERP. Vous devez cette fois obtenir un mouvement fluide.*

4 Déformation par skinning sur CPU

Pour l'instant, le maillage est déformé par skinning rigide sur le CPU par rapport à l'os d'influence prépondérante. Cela explique la position extrème de certains sommets.

Question 5 *Observez la déformation importante de certains sommets au cours du mouvement. Notez le déplacement rigide de certains groupes de sommets.*

La fonction de déformation est liée à la méthode d'affichage appelée par *mesh skinning.drawcpu()*. Cette méthode est défini dans la classe *AnimatedMesh*. Son fonctionnement global est le suivant :

1. Création d'une classe de maillage standard
2. Déformation par skinning et ajout des sommets déformés dans le maillage
   (a) Récupération de la positions relative du sommet dans le référentiel des 4 os.
   (b) Récupération des poids de skinning liés aux 4 os.
   (c) Récupération de l'orientation et de la position du joint courant
   (d) Calcul de la nouvelle position relative à l'instant courant pour chaqun des 4 os. Par exemple, pour l'os 1, on aura $x_{r1} = q_1 x$, avec q_1 quaternion lié au joint 1, x position relative initiale du sommet dans le repère du joint 1, et x_{r1} position relative finale lié à cet os.
   (e) Calcul de la position finale interpolée d'après les poids de skinning pour les 4 os. Si on note les positions relatives finales des sommets pour les 4 os par $(x_{r1}, x_{r2}, x_{r3}, x_{r4})$, les positions absolues des 4 joints par $(x_{j1}, x_{j2}, x_{j3}, x_{j4})$, et les 4 poids de skinning correspondant par (w_1, w_2, w_3, w_4), alors la position finale du sommet est donnée par

$$y = \sum_{k=1}^{4} w_k (x_{jk} + x_{rk})$$

3. Ajout du nouveau sommet dans la classe de maillage.

Pour l'instant, seul le premier joint est pris en considération de manière rigide.

Question 6 *Complétez la boucle de déformation pour réaliser l'interpolation par skinning lisse suivant les 4 joints. On notera que les informations de coordonnées relatives, poids de skinning, numéro de joint sont concaténées dans la variable* vertex attrib0123.
Au final vous obtiendez un résultat comparable à la fig. 1

5 Deformation sur GPU

On va maintenant réaliser la déformation par skinning directement sur GPU.

Question 7 *Passez la variable globale* isGPU *à true. Observez l'appel à* AnimatedMesh : :draw *ainsi que le shader associé. Que pouvez dire de l'animation observée sur l'écran.*

On rappelle que la multiplication matricielle entre une matrice de rotation et un sommet de l'espace x peut se réaliser dans l'espace des quaternions par l'opération $y = qxq^\star$, où $q = (\mathbf{v}, w)$ est un quaternion, $q^\star = (-\mathbf{v}, +w)$ est son conjugué. x étant alors considéré comme un quaternion dont la dimension $w = 0$. De plus, on rappelle que la multiplication entre deux quaternions $q_1 = (\mathbf{v}_1, w_1)$ et $q_2 = (\mathbf{v}_2, w_2)$ peut s'exprimer par

$$q_1 q_2 = (w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2, w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2).$$

Question 8 *Implémentez dans le vertex-shader la rotation par quaternion (stocké dans un vec4) sur un vecteur de l'espace (stocké dans un vec3). On considèrera la signature de fonction suivante :*
vec3 applyRotation(vec4 q,vec3 p)
(autre solution : passez directement par l'expression matricielle)

Note : Les produits scalaires et vectoriels pourront être appelés par les commandes : *dot(vec3,vec3)* et *cross(vec3,vec3)*.

Question 9 *Complétez le shader afin de calculer le skinning complet similairement à ce qui avait été fait sur le CPU. Comparez le temps d'execution (ex. sur carte graphique actuelle en fig. 6).*

FIGURE 6 – Vitesse de raffraichissement lors de l'implémentation sur GPU.

Question 10 (Supplément si il reste du temps) *Implémentez le support de la déformation des normales du maillage sur GPU pour obtenir un rendu avec shading.*