Département d'Informatique et de Recherche Opérationnelle
Université de Montréal

Mikhail Bessmeltsev
Bureaux : 3357
bmpix@iro.umontreal.ca

IFT3355 : Infographie
Travail Pratique 2 : Amusez-vous un peu avec les transformations

Disponible :	Mercredi 29 septembre 2021	23 : 59
Remise :	Mercredi 20 octobre 2021	23 : 59

1 Introduction

L'objectif principal de ce TP est de pratiquer les rotations, translations, changements d'échelle ainsi que d'appliquer les transformations sur un maillaige à l'aide du *Linear Blend Skinning*. Vous serez en charge de créer, d'appliquer et de mettre à jour des matrices de transformation en *JavaScript* sans utiliser les fonctions de *Three.js*. Les matrices de transformation résultantes devront rester relativement simples. Trois tâches principales devront être réalisées : créer un modèle de robot, l'animer et appliquer les transformations au maillage.

2 Introduction

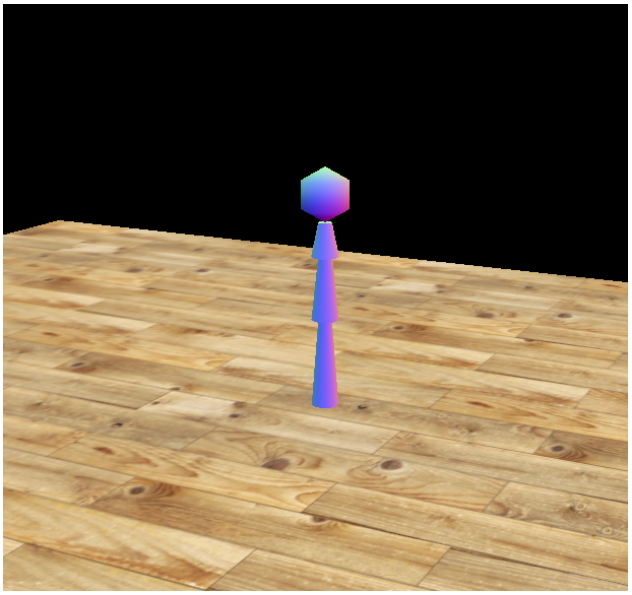

FIGURE 1 – Scène initiale : la partie supérieure du robot.

Le template fourni est très similaire à celui utilisé dans le TP précedent. On notera toutefois quelques différences :

— Le modèle d'armadillo n'est plus chargé dans la scène. A la place, quatre primitives 3D représentant la partie supérieure d'un robot sont générées dans la scène.

— Une nouvelle fonction à compléter nommée `updateBody` est fournie. Cette fonction est appelée à chaque frame et devra être complétée pour animer la géométrie.

3 Règles importantes

Three.js fournit plusieurs outils pour hiérachiser et transformer les objets 3D. Toutefois et comme l'objectif principal du TP est de comprendre comment ces méthodes fonctionnent, **vous ne devrez pas les utiliser**. En particulier, vous devrez :

1. Déclarer explicitement les nouvelles matrices en utilisant la méthode `Matrix4().set`. La création de matrices à l'aide d'opérations sur des matrices existantes est également permise (ex : multiplication). La création de matrices à l'aide de fonctions telles que `Matrix4().makeRotationX` n'est pas autorisée.
2. Les objets doivent bouger en changeant leurs repères à l'aide de la méthode `object.setMatrix`. Les opérations sur leurs attributs de position, de rotation et d'échelle ne sont pas autorisées.
3. La création de hiérarchie à l'aide de l'attribut `parent` n'est pas autorisée.

Vous pourrez trouver un exemple d'initialisation et de transformation d'objet dans le template fourni.

4 Travail à réaliser (100 pts)

Dans un premier temps, assurez-vous que vous puissiez exécuter le code dans votre naviguateur web. Pour ce faire, n'hésitez pas à regarder de nouveau les instructions du TP précédent. Souvenez-vous également de suivre les exigences décrites dans la section précédente. Étudiez la structure du template de manière à avoir une compréhension globale de la façon dont le programme fonctionne.

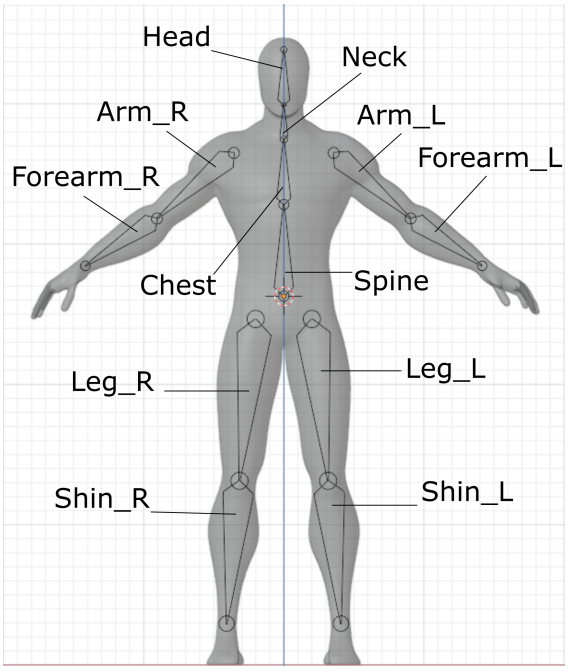

FIGURE 2 – Disposition et nomenclature des os

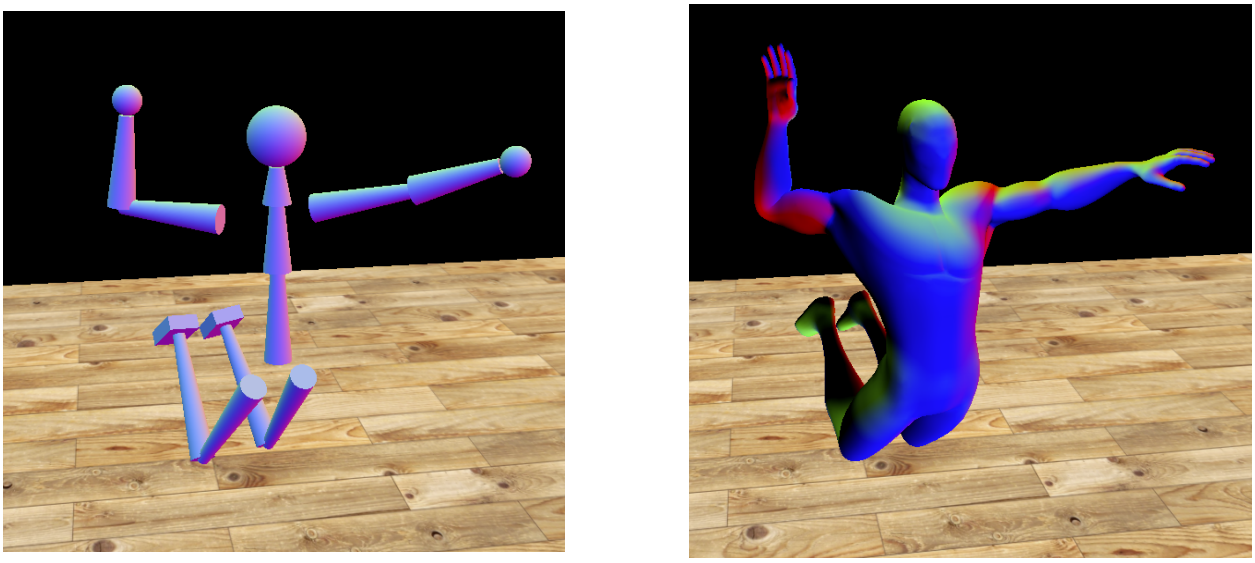

FIGURE 3 – Exemple de pose de haut niveau

1. **Exercices** (90 pts)

   (a) **Compléter les fonctions de transformations** (5 pts).
   Les fonctions `translation(x,y,z)`, `rotX(theta)`, `rotY(theta)`, `rotZ(theta)` et `scale(x,y,z)` sont déclarées et mises à votre disposition. Vous devez les définir.

   (b) **Construire les membres** (15 pts).
   Ajoutez deux bras et deux jambes au robot. Chaque membre sera composée de trois parties : une partie supérieure (cylindre), une partie inférieure (cylindre) et une extrémité (boîte ou sphère). Notez que la partie inférieure de chaque membre devra être positionnée en fonction des coordonnées de sa partie supérieure. De même, chaque extrémité devra être positionné en fonction des coordonnées de la partie inférieure de son membre.
   Vous devrez paramétriser de manière pertinente les memebres de votre robot. Le changement d'un paramètre (ex : hauteur de la cuisse) devra entraîner un repositionnement correct des éléments qui y sont attachés.
   Vous n'avez pas à écrire vos shaders ici : vous pouvez utiliser le matériau par défaut, déjà appliqué au torse dans le template.

   (c) **Animez le robot** (30 pts).
   Il est maintenant l'heure de faire courir notre robot ! Appliquez des rotations sur les membres de manière à faire courir le robot. La partie inférieure d'un membre devra être affectée par le mouvement de sa partie supérieure , mais devra également avoir sa propre composante de rotation. L'extrémité devra subir les mêmes transformations que la partie inférieure du membre. Assurez-vous que l'animation résultante soit cyclique et se déroule à une vitesse exposée comme paramètre. Vos animations devront utiliser la variable `t` mise à jour à l'aide d'un timer déjà déclaré dans le code. L'animation résultante doît s'executer lorsque la touche '0' est pressée.

   (d) **Ajoutez plus de poses** (20 pts).
   Donnez au robot poses statiques différentes, visualisables à l'aide des touches '1' et '2'. Par exemple, appuyer sur la touche '1' pourrait montrer le robot en position de saut, appuyer sur la touche '2' pourrait montrer le robot à genoux, etc.

   (e) **Introduire le maillaige** (20 pts).

Un maillaige et son armature son charger dans la fonction (init). Le maillaige est caché par défaut, mais vous pouvez le faire apparaître à l'aide de la touche '8'. Le maillaige est associé aux shaders `human.vs.glsl` et `human.fs.glsl`. Les indices et les poids permettants le *skinning* sont envoyé au *vertex shader*. La fonction `buildSkeleton` (qui est déjà appelée dans `init`) initialise 12 os vide et initialise leur matrice de système de coordonnées (`matrixWorld`). Votre travail est de définir les matrices de transformation qui seront passé au *shader* pour votre animation et chacune des poses que vous avez concues. La méthode `buildShaderBoneMatrix` prend les matrices contenues dans l'attribut `.matrix` de chaque os du dictionnaire `boneDict` et les envoies au *shader*. Il vous sera aussi nécessaire d'appliquer l'équation de *Linear blend skinning* à chacun des sommets dans le *vertex shader*. Vous devrez réutiliser les matrices de transformations que vous avez déjà défini pour votre robot ainsi que les matrices de modèle de la pose de repos (`matrixWorld`).

2. **License créative** (10 pts)

   Nous voulons maintenant voir de quoi vous êtes capable. Nous vous encourageons à implémenter de nouveaux shaders. Vos idées doivent être d'une complexité similaire à celle des tâches précédentes. Si vous avez des doutes, validez vos idées avec votre professeur ou un TA. Vous pourriez, par exemple, vouloir :

   — Créez des vertex shaders pour certains objets mobiles de la scène et utilisez leurs repères dynamiques (défini par `modelMatrix` en GLSL)

   — Ajoutez des objets supplémentaires à la scène pour embellir vos poses (et peut-être faire intéragir le robot avec ces objets).

   — Ajoutez un ou plusieurs robots à la scène et animez/positionnez-les différemment.

   Des points bonus seront attribués pour récompenser des pistes d'exploration particulièrement remarquables.

5 Remise

Créez un repertoire VOTRENOM_TP2 et placez-y les fichiers sources du projet en respectant la hiérarchie du template. Placez-y également un fichier README.txt contenant votre nom ainsi que toute information que vous aimeriez fournir au correcteur. Compressez ce repertoire avant de le rendre sur StudiUM.

6 Détails

Ce TP utilise les modules javascripts intoduits dans la 6^e édition du standard *ECMAScript* (ES6). Si vous utilisiez Webstorm dans le premier travail de la session, vous ne devriez pas avoir de problème. Par contre, si vous utiliser un serveur local (tel qu'avec la commande `python -m http-server`), vous pourriez avoir des problème. Deux options s'offrent à vous :

1. Utiliser le script python fourni dans le répertoire principal du travail avec la commande `python server.py`
2. Utiliser Webstorm