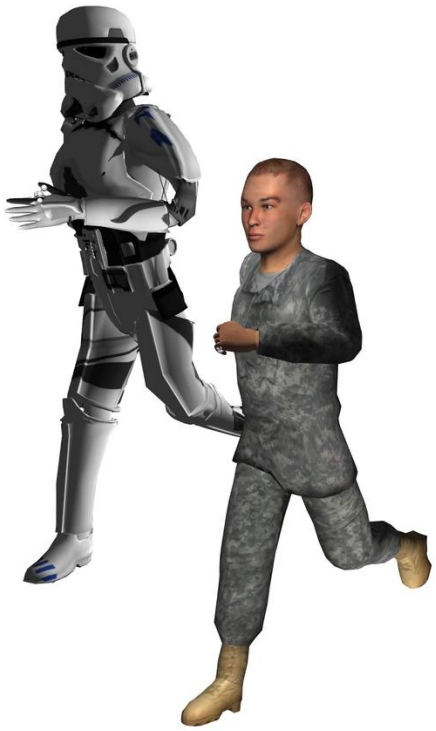
# X-Wing & Stoormtrooper animés

INF585 – Projet de Computer Animation

Cédric JAVAULT – Master AI Year 1

# Introduction

Pour ce projet de fin de trimestre, j'ai voulu travailler sur des notions vues en cours, mais que je n'avais pas eu l'occasion d'implémenter en TD. J'ai ainsi manipulé une **courbe d'interpolation** (type Catmull Rom Spline) pour définir la trajectoire d'un vaisseau spatial, créé du relief au sol avec un **bruit de Perlin**, manipulé des **matrices de rotation** dans tous les sens (pour le X-Wing mais aussi pour tous les joints du squelette), et fait du **skinning de manière procédurale** pour animer un Stormtrooper à partir d'un mesh 3D.

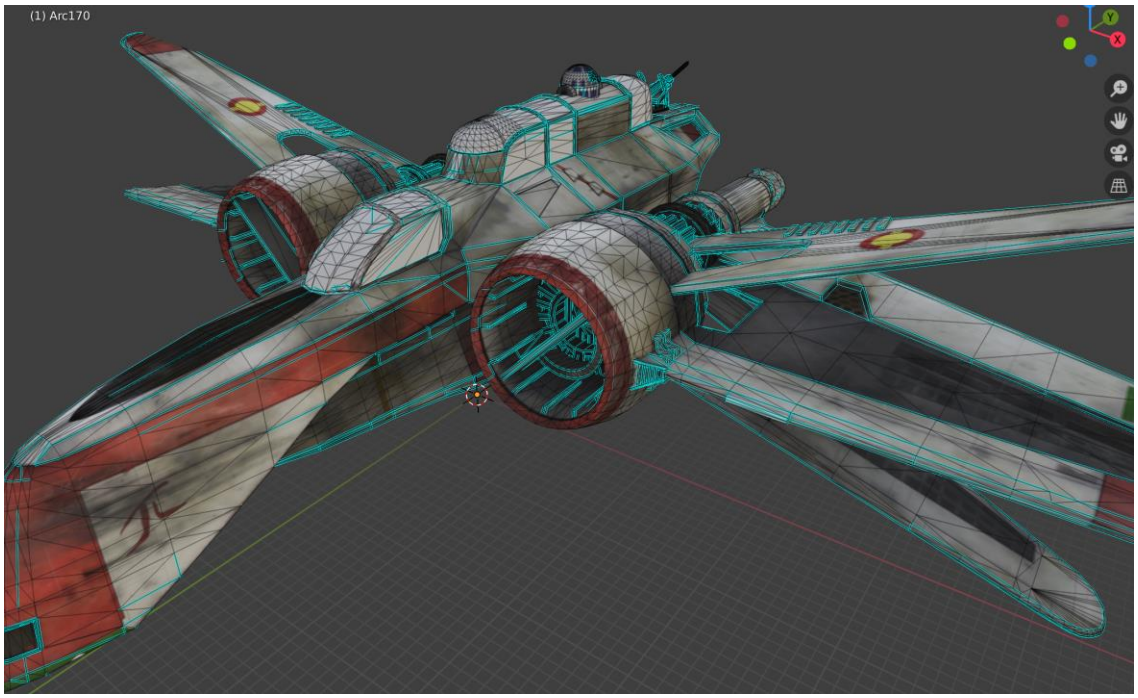


En INF585, et c'est certainement ce donne une touche de magie à ce cours, on est souvent à la limite entre l'informatique et « l'art ». J'ai choisi de **me concentrer sur les aspects informatiques** sans creuser outre mesure certains points qui auraient donné plus de réalisme au rendu final mais tenaient plus du côté artistique (à titre d'exemple, je ne suis pas très satisfait de la texture du sol à gauche)

Je souhaitais initialement faire un seul mini dessin animé avec à la fois un X-Wing et un Stormtrooper ; j'ai donc mené de front les deux développements dans l'idée de les fusionner ultérieurement. Mais j'ai ensuite réalisé qu'il serait difficile de les animer dans la même séquence (les tailles sont assez différentes) et qu'il fallait probablement faire se succéder une séquence 'X-Wing, puis une autre 'Stormtrooper'… Comme cela n'avait pas grand intérêt et que le travail était fait, j'ai finalement décidé de laisser mes développements en l'état et de présenter **deux demi projets séparés**.

# I. Le mini projet X-Wing

Je voulais initialement animer le Faucon Millenium de Star Wars. Je n'ai toutefois pas réussi à trouver un Mesh 3D qui me convenait (trop lourd, trop léger, mauvais format, pas de texture…). Je me suis finalement rabattu sur un X-Wing[1] avec 30 000 vertex et 51 000 faces. Le voici sous Blender :




**Etape 0**. Je suis reparti du code des TD pour gagner du temps et j'ai utilisé les procédures déjà implémentées pour charger le mesh et sa texture. J'ai remis le X-Wing dans le bon axe avec une petite matrice de rotation.

**Etape 1**. Je voulais faire tourner le vaisseau au-dessus d'un point d'atterrissage, puis le poser, et enfin le faire redécoller. J'aurai bien sûr pu programmer une courbe p(t)=(x(t),y(t),z(t)) avec un x(t)=Rcos(t) et y(t) = Rsin(t) pour la rotation au début, et d'autres paramètres pour la suite. mais j'ai préféré tester une **interpolation par courbe cubique** sur une suite de points (type Catmull Rom Spline).

- C'est à la fois très simple (on gère juste une suite de 3 coordonnées : x, y, z + le temps) et assez compliqué à paramétrer : quand on a défini les points dans l'espace-temps, la trajectoire est entièrement contrôlée… sauf qu'il n'est pas si simple de trouver à la main les points même si on connait le mouvement souhaité.
- Cela donne des résultats un peu surprenants : de petits virages contre intuitifs, ou un effet de retour vers x négatif (soit x'(t)<0) … alors que tous les x sont croissants.
- Mais au final, après quelques tâtonnements, j'ai fini par avoir un effet assez réaliste et amusant !

**Etape 2.** Pour le sol, je ne voulais pas un sol plat mais quelque chose d'un peu plus chahuté. J'ai donc introduit une grille 300 x 300 dont la position en z est fonction d'un un bruit de Perlin :

```
// Mise en place du sol avec un bruit de Perlin pour modéliser le relief
const size_t N = 300;
shape = mesh_primitive_grid(N, N,vec3(-30, -30, 0), vec3(60, 0, 0), vec3(0, 60, 0));
for (size_t x = 0; x < N; x++)
    for (size_t y = 0; y < N; y++) {
        shape.position[y * N + x].z = -4+perlin(((float)x) /N, ((float)y) / N, 8, .7f, 2.0f)*2;
    }
```

[1] Pour les puristes de Star Wars, il semble que ce n'est exactement un X-Wing, mais un ARC170, un modèle proche.

Pour que le caractère non plat du sol soit bien perçu par le « spectateur », on ne peut pas se contenter d'une couleur uniforme au sol, il faut une texture. J'ai eu du mal à trouver quelque chose de satisfaisant car les textures de sol que je trouvais sur Internet étaient en base résolution (ou étaient payantes). J'ai fini par en mettre 3 différentes (de la lave, un bout du sol lunaire que j'ai trouvé sur le site de la Nasa[2], et un bruit procédural que j'avais généré pour INF584[3]). Je propose aussi à l'utilisateur de mettre le simple wireframe.


**Etape 3**. Il faut que le vaisseau s'oriente dans la direction du mouvement. Je voulais le laisser « à plat » par rapport au sol et il n'y a alors qu'un degré de liberté (angle de rotation autour de Oz). Cet angle est guidé par la direction de (x'(t), y'(t)). Rien de bien compliqué une fois qu'on pose correctement les données… à condition toutefois de gérer les situations où le vaisseau s'arrête (x'=y'=0 -> éviter la division par zéro et imposer la direction précédent l'arrêt). Cela se gère par un petit bout de code comme celui-ci :

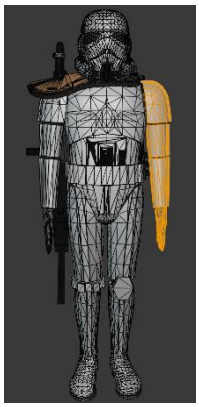
```
p = locatisation_xwing(t);
vec3 direction = p - lastp;
lastp = p;
float a = direction[0];
float b = direction[1];
if (a != 0) {     // Pour éviter les discontinuités
    angle = atan(b / a);
    anglederotation = 3.14159 / 2 - angle;
    if ((a < 0) && (abs(b) + abs(a)) > 0.0003)
        anglederotation += 3.14159; // atan ne donne une réponse qu'en -pi/2 et +pi/2
}
mat3 rotationpourremettredanslaxe = mat3(vec3(cos(anglederotation), -sin(anglederotation), 0.),
    vec3(sin(anglederotation), cos(anglederotation), 0),
    vec3(0, 0, 1));
mesh_xwing.uniform.transform.rotation = (rotationpourremettredanslaxe * remisedanslebonsens);
```

[2] La NASA a publié une carte de la Lune en très haute résolution qui est libre de droit… Seul « problème, ils ont indiqué les noms des différents lieux sur la carte, et mis une grille.

[3] INF 584 = Raytracing. J'ai implémenté un bruit de Wormey pour gérer un fond derrière le visage ; je me suis resservi du code qui génère ce fond pour fabriquer une texture pour le sol.

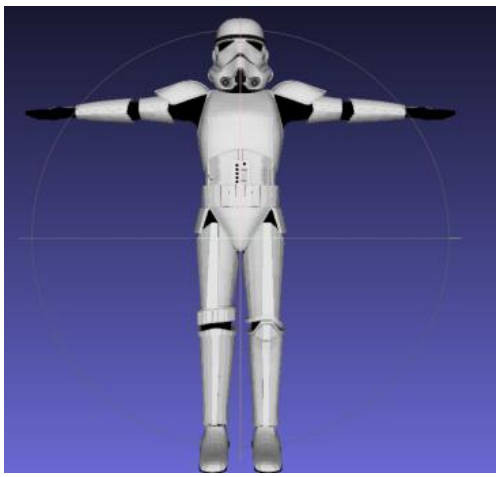
# II. Le mini projet Stormtrooper

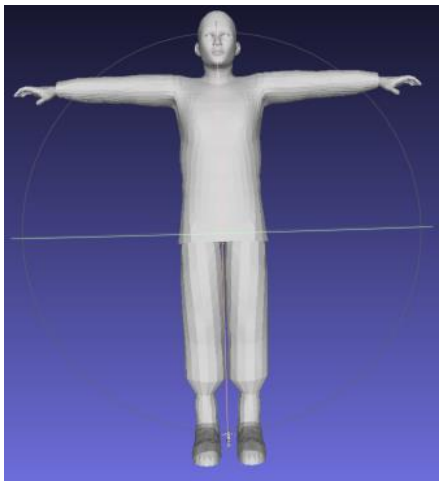
Pour ce second mini projet, changement de décor : je voulais remplacer le personnage qui court / marche dans le TD de Skinning par un Stormtrooper. Finalement, j'ai adapté le code pour les mettre **côte à côte, dans le même mouvement**. Les enseignants m'avaient bien indiqué en TD que le skinning n'était pas simple et que je risquais de souffrir pour rattacher correctement l'ensemble des vertex ; je confirme : ce fut un **travail compliqué avec un débogage difficile** !!! Entrons dans le détail.



**Etape 0**. Comme pour le premier mini projet, je suis parti du code fourni pour les TD et j'ai commencé par chercher un Stormtrooper. J'ai perdu du temps sur un premier modèle qui était trop loin de la position de repos du personnage du TD pour pouvoir faire un bon recollement : il aurait fallu commencer par remettre les bras à l'horizontal. C'est certainement possible avec Blender en sélectionnant tous les vertex du bras, et en les faisant pivoter (j'avais commencé - voir illustration à droite). Problème : j'aurai dû passer plusieurs heures sur Blender et apprendre à mieux le maîtriser ; ce n'était pas mon objectif.

J'ai heureusement trouvé un autre modèle de Stormtrooper qui avait le bon goût d'être beaucoup plus proche (à une rotation et un scaling près) du personnage du TD : 46 200 vertex pour 88 000 faces que j'ai immédiatement simplifié avec MeshLab pour ne garder que 4 fois moins de vertex et de faces[4] :



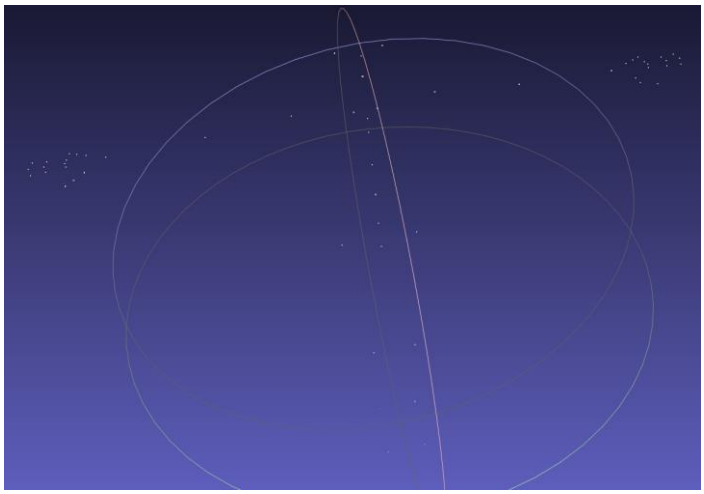


**Etape 1**. Le squelette du personnage fourni en TD a 57 joints. J'ai pris leur position 3D dans la position de repos et j'ai **« matché » un par un les 57 points pour le Stormtrooper**. Cette étape de positionnement précis en 3D dans le Stormtrooper s'est avérée clé pour faire ensuite un bon skinning. Je me suis servi de MeshLab et de ses fonctions de sélection conditionnelle des vertex pour bien repérer mes points. J'ai aussi beaucoup utilisé Excel pour vérifier et ajuster les positions 3D et m'assurer **d'une cohérence pour les points qui s'enchaînent** (colonne vertébrale, jambes, bras) et pour gérer les **symétries**.

La sensibilité est forte. A titre d'exemple, une très légère erreur (épaules légèrement vers l'arrière sur le Stormtrooper alors qu'elles sont dans l'axe pour le personnage initial) se traduisait par une forte déformation du dos dans le mouvement. Le travail a fait l'objet de plusieurs itérations : j'ai corrigé des problèmes visuels dans le rendu en modifiant l'emplacement de certains points. Le nom du fichier portant cette géométrie (« ST2_geometry_ local_**v5** ») montre d'ailleurs bien que ce fut un peu laborieux…

[4] Pour deux raisons : accélérer l'exécution en phase de débogage et éviter de saturer les capacités de mon petit ordinateur dans le rendu final !

Voici le résultat final avec la position en 3D des 57 points du squelette (tête en haut ; on devine les mains en haut à droite et à gauche, etc.) :



**Etape 2**. Le skinning doit maintenant être fait de **manière procédurale** (on ne le charge pas comme pour le personnage du TD). J'ai commencé par utiliser la formule vue en cours :

$$\text{Using cartesian distances: } \alpha_1 = d_1^{-1}/(d_1^{-1} + d_2^{-1})$$

Mais il a fallu ajuster :

- J'ai dû **simplifier et oublier les doigts** ! D'une part, le personnage du TD a les doigts légèrement repliés vers le bas mais pas le Stormtrooper. D'autre part, évidemment, des erreurs de position sur les phalanges sont très vite faites et ont un gros impact. Pour éviter des horreurs visuelles, j'ai donc forcé le rattachement à la main de toutes les phalanges des 5 doigts (passage de 57 à 19 joints !).
- J'ai géré le **problème des jambes** : pour certains vertex, les deux plus proches voisins ne sont pas sur la même jambe ! J'ai limité le problème en rapprochant volontairement les joints des jambes de l'axe (du coup, ils ne correspondent pas exactement au personnage du TD à cet endroit). Plus important : j'ai imposé que chaque vertex **ne soit rattaché qu'à des joints de la même jambe**.
- Enfin, j'ai pris en compte **les 3 plus proches voisins** et pas seulement les deux plus proches, ce qui a été assez miraculeux sur les derniers problèmes que je rencontrais.

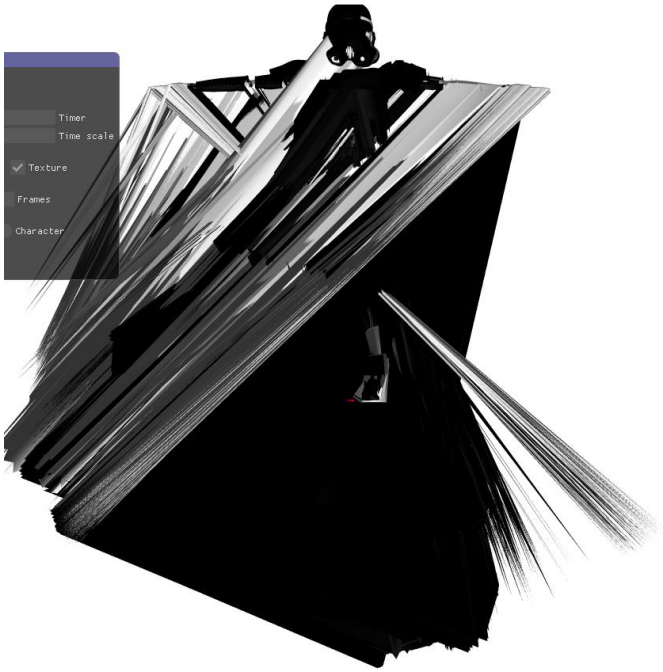
**Etape 3.** Il faut maintenant transférer l'animation du squelette du personnage du TD au Stormtrooper. C'est assez facile, un fois qu'on a bien compris les structures de donnée ; encore faut-il bien poser les « équations » et être rigoureux. Le point clé, c'est de garder exactement la même orientation dans la géométrie, mais **d'ajuster la taille des os** par un facteur ratio = (taille de l'os au repos pour Stormtrooper) / (taille de l'os au repos pour le perso). Voici le bout de code qui fait cela :

```
// Maintenant, il faut passer skeleton_geometry_local au Stoormtrooper dans sa géométrie à lui !
const size_t N = skeleton.connectivity.size();
buffer<joint_geometry> st2_current;
st2_current.resize(N);
st2_current[0].r = skeleton_geometry_local[0].r;
st2_current[0].p = skeleton_ST2.rest_pose[0].p;

for (size_t k = 1; k < N; ++k)
{
    const int parent = skeleton.connectivity[k].parent;
    float taille_bone_st2 = norm(skeleton_ST2.rest_pose[k].p);
    float taille_bone_pTD = norm(skeleton.rest_pose[k].p);
    float ratio = taille_bone_st2 / taille_bone_pTD;
    st2_current[k].r = st2_current[parent].r * skeleton_geometry_local[k].r;
    st2_current[k].p = st2_current[parent].r.apply(ratio * skeleton_geometry_local[k].p) + st2_current[parent].p;

    // On en profite pour copier l'orientation des quaternions des rest_postions
    skeleton_rest_pose_ST2.data[k].r = skeleton_rest_pose.data[k].r;
}
```

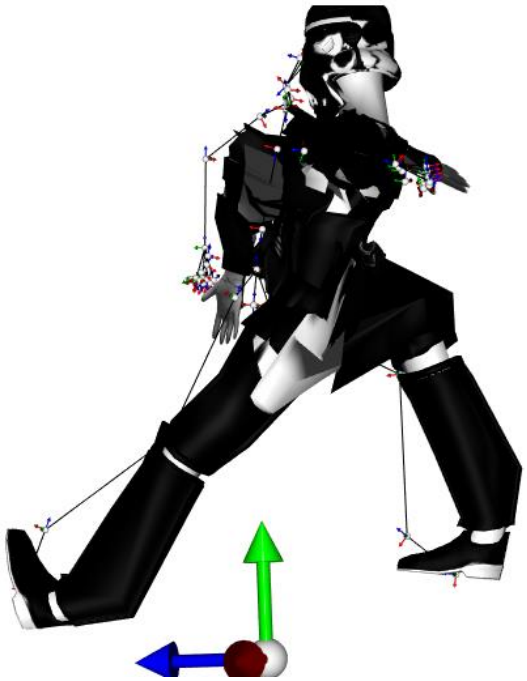
**<u>Etape 4</u>**. Le débogage est assez compliqué !

- Les différents objets n'ont pas toujours la même échelle et la même orientation initiale, la même position initiale, ce qui est source d'erreur.
- On gère des milliers de vertex et de faces sans savoir a priori lesquels tester !
- Tout est en 7D avec des positions 3D (déjà pas si facile à « lire » facilement) et des quaternions (franchement illisible, en tout cas pour moi)
- Quand il y a un bug, on ne se sait pas bien si le problème vient de la position calculée pour les joints dans le squelette, d'une approximation dans le skinning, ou d'encore autre chose…

Mais soyons visuels et ludiques : voici quelques-uns des bugs que j'ai recontrés !

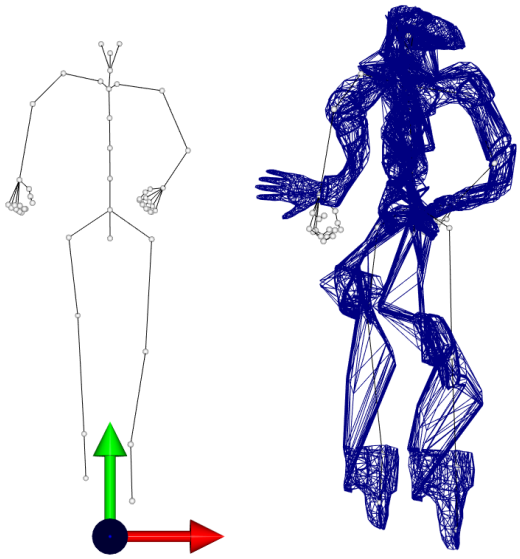



J'ai gagné une cape de Dark Vador !



Je crois que je marche à l'envers…

J'ai le bras long !



Euh, mes quaternions sont mal orientés !

# Aspects pratiques

Comme indiqué plus haut, je suis reparti de la base de code donnée pour les TD. Pour chacun des deux mini projets, je n'ai modifié qu'une poignée de fichiers que j'annexe à ce rapport :

- **skinning et skinning_loader** (soit 4 fichiers avec les .h et .cpp) : **le cœur de mon travail**
- **current_scene.hpp, sources_scenes.hpp et main.cpp** : changements très mineurs.
- **camera.hpp et .cpp** : j'ai ajouté une ou deux routines au code préexistant pour positionner la caméra facilement depuis main.
- **quaternion.cpp et .h** : mon PC est formel, j'ai modifié ces fichiers ; la modification ne m'a pas frappée et elle doit être mineure ; je fournis néanmoins les versions qui m'ont servi au moment de la compilation.

J'ai bien sûr également joint les données qui permettent de faire tourner les mini-projets :

- **Pour le X-Wing** : le sous dossier ARC170 qui contient le mesh et la texture, ainsi que les trois fichiers PNG pour la texture du sol : lave, nasa et noise.
- **Pour le Stormtrooper** : le sous dossier ST2 contient le mesh et la texture mais aussi le fichier ST2_geometry_local_v5, construit de haute lutte, avec la géométrie du squelette en position de repos. Je n'ai pas remis le personnage du TD.

Bien évidemment, si vous vouliez recompiler et rencontriez un problème, n'hésitez pas à me contacter (cedric.javault@polytechnique.edu) ; je ne suis pas à l'abris d'une erreur de fichier.

# Conclusion

Je pense l'avoir dit et écrit dans le petit questionnaire, je me suis régalé en cours et en TD ; c'est avec le même enthousiasme que je me suis attelé à ce projet de fin de trimestre.

La partie X-Wing ne présentait pas de difficulté particulière ; elle m'aura permis d'implémenter des éléments vus en cours et de me familiariser un peu plus avec la manipulation des matrices de rotation et la géométrie 3D. Pour la partie Stormtrooper, j'avoue que je ne soupçonnais pas les difficultés que j'allais rencontrer, et particulièrement l'impact graphique de la moindre approximation dans le skinning.

Un point m'aura marqué au final, dans les deux mini projets, tout comme il m'avait déjà frappé en TD : quand on a un problème voire un bug, plutôt que de multiplier les tests en espérant trouver la solution « avec de la chance », il faut savoir se poser, **traduire son problème graphique en langage mathématique** : finalement, on aboutit presque toujours à une équation assez simple. Comme quoi, les profs de maths qui ont dû mal à intéresser leurs élèves, les profs de prépa qui ont du mal à trouver des applications concrètes à l'algèbre linéaire… tous devraient venir faire un cours de Computer Animation !