HOANG Valentin 5ETI
ROBERT Antoine Majeur IMI

Module Animation – Simulation:
Compte Rendu du TP Skinning

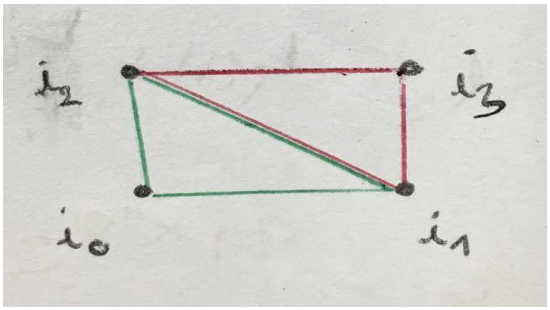
Introduction:

Au cours des différentes séances de TP, nous avons pu mettre en applications les notions que nous avons vues en cours concernant l'animation et la simulation. Nous avons dans un premier temps, implémenter l'animation d'un tissus, dans un second temps, appliquer le skinning à un cylindre et à un chat et finalement étudier l'algorithme de Convex Hull. Notre rapport traitera de la partie skinning, que nous avons approfondis et amélioré.
L'objectif de ce TP est d'appliquer une déformation 3D aux objets que nous avons. Pour appliquer cette déformation, nous allons utiliser la méthode du Skinning. On va pour cela, déformer la surface voulue grâce à son squelette articulé. Nous avons appliqué cette méthode pour un cylindre (composé de deux os), puis sur une forme un peu plus complexe : un chat.

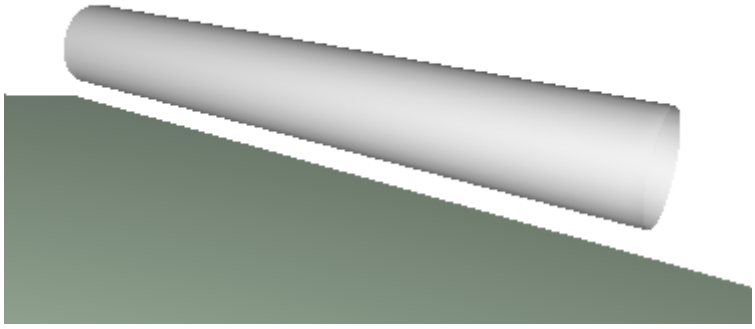
Skinning du cylindre:

Plusieurs étapes sont nécessaires pour déformer notre cylindre. La première étape est de créer le maillage du cylindre. Il faudra ensuite créer son squelette et l'animer. Grâce à l'animation du squelette, nous finirons par déformer le cylindre. Nous allons, dans cette partie, détailler ces différentes étapes.

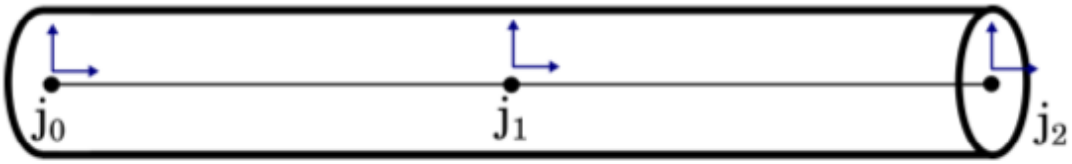
Le maillage du cylindre se fait en deux étapes. Nous calculons la position des vertex pour former un disque de centre r à la hauteur h, puis nous faisons varier cette hauteur entre 0 et la longueur voulu du cylindre. Une fois la position vertex obtenus, il suffit alors d'établir la connectivité pour avoir la forme que nous voulons. Pour faire cela, nous créons à chaque fois, deux triangles. Voici comment la connectivité est établie:



Il ne reste plus qu'à afficher notre mesh afin d'avoir notre cylindre fixe :



Passons à la partie concernant le squelette. Nous allons maintenant créer le squelette du cylindre correspondant à la géométrie actuelle de la surface que l'on appelle position au repos ou encore *bind pose* car c'est dans cette pose que nous allons attribuer les poids de skinning par la suite. Pour réaliser le squelette nous considérons que le cylindre est formé d'une hiérarchie simple de deux os mis bout à bout. Nous définissons donc 3 positions permettant de définir ces deux os et chaque position est associé à un repère propre. Voici la géométrie du squelette voulue :



Chaque repère est défini de manière locale, c'est-à-dire que le repère du joint 1 est définie par rapport au repère 0, et le repère du joint 2 est définie par rapport au repère du joint 1. Chaque repère est défini par sa position et sa rotation exprimé par un quaternion. Dans un premier temps nous créons le squelette correspondant à la *bind pose* dans une structure géométrique qui stocke chaque repère de joint.

```
skeleton_joint joint0(vec3(0,0,0),quaternion(0,0,0,1));
sk_cylinder_bind_pose.push_back(joint0);
skeleton_joint joint1(vec3(0,0,length/2.0f),quaternion(0,0,0,1));
sk_cylinder_bind_pose.push_back(joint1);
skeleton_joint joint2(vec3(0,0,length/2.0f),quaternion(0,0,0,1));
sk_cylinder_bind_pose.push_back(joint2);
```

La position et l'orientation des joints étant exprimées dans un repère local il n'est pas possible d'afficher le squelette immédiatement, il faut d'abord convertir ces coordonnées locales en coordonnées globales. Soit q_{Gj} le quaternion et t_{Gj} la translation du repère j exprimé dans le repère global, q_{Lj} et t_{Lj} leur expression dans le repère locale et p(j) l'indice du parent du joint j. On a alors :

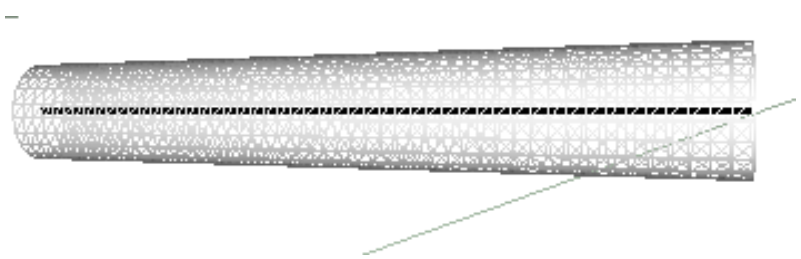
$$q_{Gj} = q_{Gp(j)} * q_{Lj}$$

$$t_{Gj} = q_{Gp(j)} * t_{Lj} + t_{Gp(j)}$$

Une fois les joints du squelette passé dans le repère global il faut tracer des segments joignant deux joints consécutifs dans la hiérarchie, ceci est réalisé dans la fonction *extract_bones()* qui génère un vecteur stockant les os sous la forme de pair de point 3D.

```
std::vector<vec3> extract_bones(skeleton_geometry const& skeleton,skeleton_parent_id const& parent_id)
{
    ASSERT_CPE(skeleton.size()==parent_id.size(),"Incorrect size");

    std::vector<vec3> positions;
    int const N_joint = parent_id.size();
    for(int k=1 ; k<N_joint ; ++k)
    {
        int const parent = parent_id[k];

        //TO DO: completez la structure position avec la position des extermitees des os.
        // question 10
        positions.push_back(skeleton[parent].position);
        positions.push_back(skeleton[k].position);

    }

    return positions;
}
```

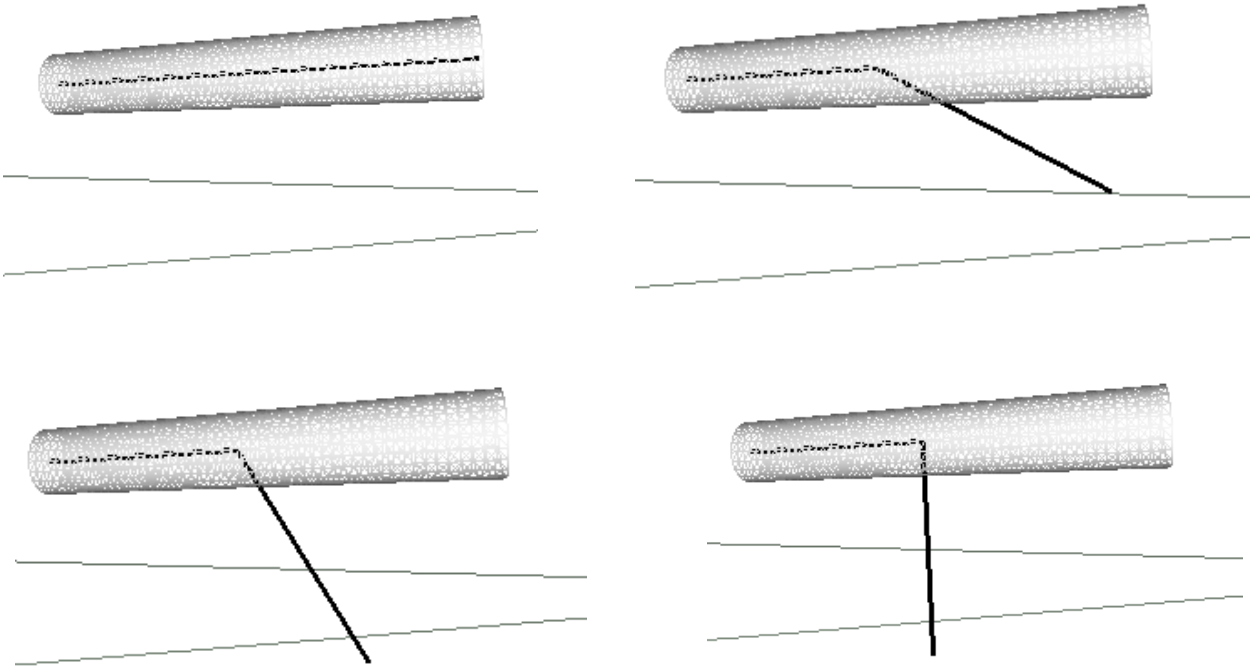
Il ne nous reste maintenant plus qu'à afficher le squelette avec le cylindre :



Maintenant que le squelette a été créé, il faut l'animer. En d'autres termes, cela revient à définir la position et l'orientation de tous les joints à chaque instant considéré. Dans notre cas, on souhaite réaliser notre déformation avec un angle variant de 0° à 90° avec un pas de 30. Cette animation aura lieu, spécifiquement sur le joint 1 (celui au milieu), mais il faut également déclarer la position et l'orientation du joint 0 et du joint 2. La position du joint 1 sera la même cependant l'orientation va changer. En effet, nous allons lui appliquer une rotation suivant l'axe x avec la liste des angles que l'on veut. Attention aux unités qui peuvent entraîner des déformations bizarres. En effet, la méthode des quaternions *set_axis_angle* attend un angle en radian et nous avons des angles en degré. Pour faire notre animation on remplit un squelette d'animation avec la suite des squelettes géométriques que l'on a défini :

```
int const angle[4] = {0,30,60,90};

for(int i=0; i<4; ++i)
{
    skeleton_geometry pose;
    quaternion rotation;
    rotation.set_axis_angle(vec3(1,0,0),angle[i]*M_PI/180.f);
    skeleton_joint joint0(vec3(0,0,0),quaternion(0,0,0,1));
    pose.push_back(joint0);
    skeleton_joint joint1(vec3(0,0,length/2.0f),rotation);
    pose.push_back(joint1);
    skeleton_joint joint2(vec3(0,0,length/2.0f),quaternion(0,0,0,1));
    pose.push_back(joint2);

    sk_cylinder_animation.push_back(pose);
}
```

Pour que l'animation du squelette ait bien lieu, on utilise une variable *frame_cylinder_current*. Elle varie au cours du temps entre 0 et le nombre d'animations que l'on a (c'est-à-dire 4 animations dans le cas du cylindre). Dans la fonction de dessin de la scène nous passons cette variable comme indice au squelette d'animation ce qui a pour résultat de dessiner successivement les squelettes géométriques et ainsi réaliser l'animation. Voici les différentes positions de notre squelette :



Nous pouvons voir que pour le moment le cylindre ne suit pas les mouvements du squelette. Pour résoudre ce problème il faut dans un premier temps attribuer les poids de skinning, c'est-à-dire l'attachement entre chaque sommet de la surface et les repères du squelette. Dans notre cas seul les deux premiers joints sont utiles le troisième n'ayant aucun intérêt visuel. Une première approche pour l'attribution des poids consiste à dire que la première moitié du cylindre suit rigidement le premier os et que la seconde moitié suit le deuxième. Cela revient à attribué pour les sommets de la première partie du cylindre un poids de 1 par rapport au joint 0 et un poids de 0 par rapport aux autres joints. Et pour les sommets de la seconde partie un poids de 1 par rapport au joint 1 et un poids de 0 par rapport aux autres joints.
Une fois les poids attribués il faut appliquer la déformation par skinning, soit un sommet de coordonnées initial p_0 et de coordonnées p(t) après déformation par skinning à l'instant t, on a alors :

$$p(t) = \sum_{joint\ j} w_j S_j(t) p_0$$

$$\text{avec } S_j(t) = T_j(t) B_j^{-1}$$

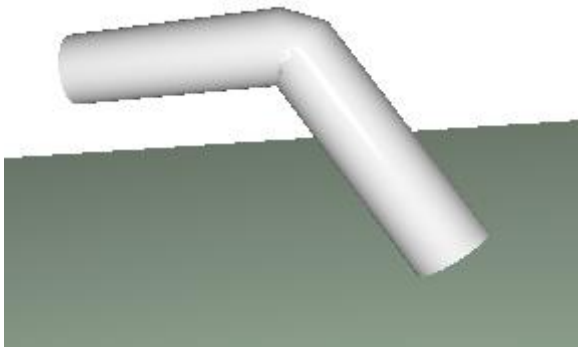
T_j correspond aux joints du squelette animé dans le repère global et B_j^{-1} correspond à l'inverse des repères de la *bind pose*. Comme nous utilisons des quaternions, calculer l'inverse de la *bind pose* est assez facile. En effet, il suffit de prendre l'opposé du vecteur position pour la position et de prendre le conjuguer du quaternion pour l'orientation. Il faut maintenant calculer la multiplication de T_j et B_j^{-1}, soit le quaternion q et la translation t du joint j résultat de la multiplication de deux joints j1 et j2, on a :

$$t = q1 * t2 + t1 \quad \text{et} \quad q = q1 * q2$$

Maintenant il suffit de calculer le résultat final du skinning avec la formule vue plus haut :

```
void mesh_skinned::apply_skinning(skeleton_geometry const& skeleton)
{
    int const N_vertex = size_vertex();
    ASSERT_CPE(N_vertex==int(vertices_original_data.size()),"Incorrect size");

    for(int k_vertex=0 ; k_vertex<N_vertex ; ++k_vertex)
    {
        //TO DO: Calculer deformation par skinning (question 18)
        vertex_weight_parameter const& skinning_info = vertex_weight(k_vertex);

        int const N_joint = skinning_info.size();

        vec3 p0 = vertex_original(k_vertex);
        vec3 p_final(0.0f,0.0f,0.0f);
        for(int k_joint=0 ; k_joint<N_joint ; ++k_joint)
        {
          float weight = skinning_info[k_joint].weight;
          int joint_id = skinning_info[k_joint].joint_id;
          p_final += weight * (skeleton[joint_id].orientation*p0 + skeleton[joint_id].position);
        }
        vertex(k_vertex) = p_final;
    }

}
```

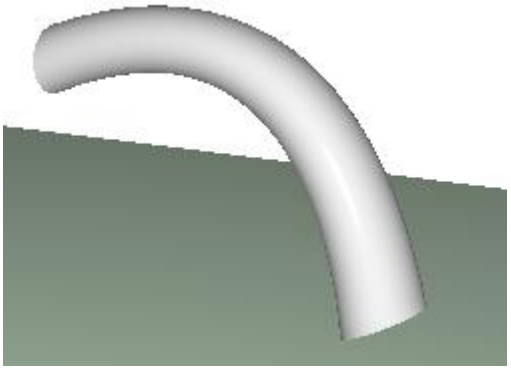
Voici une image du cylindre après l'application du skinning :



Comme nous pouvons le voir la déformation est importante à l'endroit de la pliure, ceci est due au fait que le skinning implémenté précédemment donne une dépendance rigide entre chaque sommet et un unique os. Pour avoir une déformation plus douce on peut considérer que l'influence du joint 0 décroit linéairement (entre 1 et 0) et l'influence du joint 1 croit linéairement (entre 0 et 1) en fonction de la distance le long de l'axe du cylindre :

```
weight_parameter[0].weight = 1.f - z/length;
weight_parameter[1].weight = 0.f + z/length;;
mesh_cylinder.add_vertex_weight(weight_parameter);
```

Voici une nouvelle photo du cylindre dans la même position que la photo précédente mais avec une dépendance linéaire des sommets et des os :



Notre animation pour le moment passe d'une pose clés à l'autre en faisant des sauts de 30 degrés à chaque fois ce qui la rend assez grossière. Pour donner une impression de continuité on peut interpoler la position des repères entre deux instants. Pour interpoler la position d'un point p variant de façon linéaire entre p1 et p2 on peut utiliser la formule :

$$p = \alpha p1 + (1 - \alpha)p2 \text{ avec } \alpha = (t - t_i)/(t_{i+1} - t_i)$$

Pour interpoler de la même façon les quaternions q entre deux quaternions q1 et q2 on utilise la méthode SLERP :

$$\mathrm{SLERP}(q_0, q_1, \alpha) = \frac{\sin(\alpha(1-\theta))}{\sin(\theta)} q_0 + \frac{\sin(\alpha\theta)}{\sin(\theta)} q_1$$

$$\cos(\theta) = q_0 \cdot q_1$$

$$\alpha \in [0, 1]$$

Pour finir nous implémentons la fonction *interpolated()* avec ses deux formules et nous obtenons une animation continue du cylindre.

Skinning d'un chat :

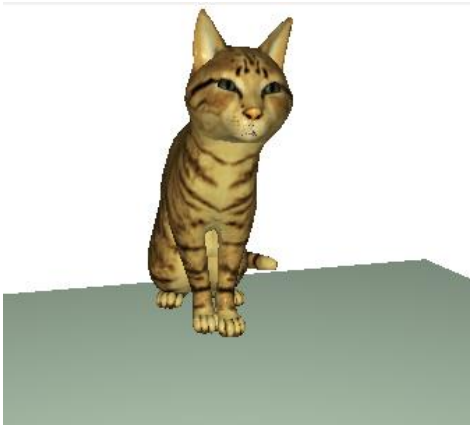
Maintenant que nous avons réussi la déformation d'une forme simple telle que le cylindre, il serait intéressant de déformer une forme un peu plus complexe : on va prendre, dans ce TP, l'exemple d'un chat. En soit, la forme que l'on veut déformer importe peu, la méthode est la même que pour une forme simple. Dans un premier temps, il faut s'assurer que le chat au repos soit bien affiché correctement et que son squelette s'affiche également bien. Il faut ensuite réussir à bien animer le squelette de la façon que l'on souhaite et enfin déformer la surface du chat.

L'archive que l'on possède est déjà très complète, puisqu'elle possède déjà les poids de skinning qui sont associé au squelette. L'animation étant déjà fournie, et partant du travail que l'on a effectué précédemment, la difficulté de la partie skinning du chat résidera dans l'adaptation des nouvelles données dans le code déjà existant.

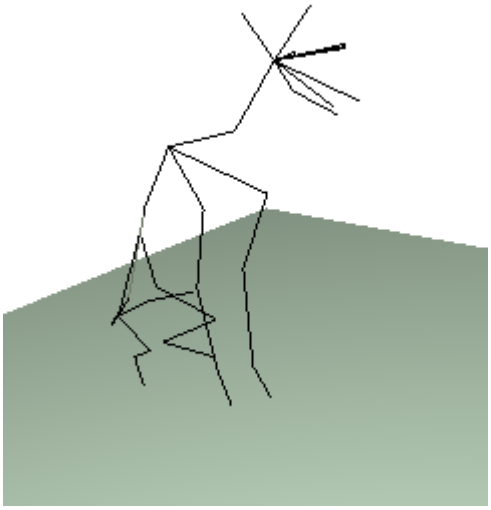
En effet, il a fallu s'assurer de bien charger le fichier .obj de notre chat et de bien définir la position de repos du squelette. En fonction de l'animation ou non du squelette, on a rempli la variable

sk_cat_interpolated différemment. S'il n'y a pas d'animation, on lui affecte la première case du tableau de *sk_cat_animation* (qui correspond à la *bind pose*) et sinon on utilise la variable *frame_cat_current* pour jouer l'animation comme pour le cylindre. Enfin, on utilise les mêmes fonctions que pour le cylindre pour animer notre squelette, puis pour déformer le chat (*local_to_global*, *extract_bones* , *inversed, multiply)*

L'exécution du code nous fournit un résultat plutôt convaincant sur l'animation du chat. L'animation n'est plus aussi triviale que pour le cylindre (où on déformait le squelette suivant un angle). On remarque qu'il y a des animations sur différents joints du squelette comme par exemple la queue du chat ou encore la tête. Voici différentes animations du chat :



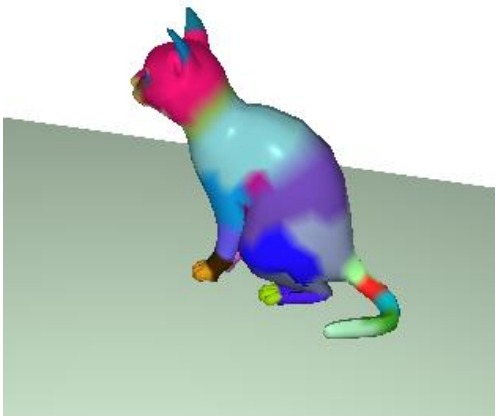
Et voici le squelette du chat :



Maintenant que le skinning a été correctement fait, nous allons maintenant visualiser les différents poids de skinning sur la surface du chat en fonction des os pour mesurer l'importance de l'attachement d'un sommet sur un autre. Pour ce faire, nous avons créé un vecteur de couleurs aléatoire de la taille du nombre de joint du chat. Pour chaque sommet du chat nous regardons tous les joints dont il dépend et gardons l'indice de celui qui a le plus gros poids. Nous colorons enfin le sommet avec la couleur présente dans le vecteur à l'indice récupéré précédemment :

```
void scene::set_color_weight()
{
    int nb_id = sk_cat_parent_id.size();
    std::vector<vec3> color_id;
    srand(time(NULL));

    for(int i=0;i<nb_id;i++)
        color_id.push_back(vec3((float)rand() / (float)RAND_MAX,(float)rand() / (float)RAND_MAX,(float)rand() / (float)RAND_MAX));

    int N_index = mesh_cat.size_vertex();

    for(int i=0;i<N_index;i++)
    {
        vertex_weight_parameter weight_data = mesh_cat.vertex_weight(i);
        int id = 0;
        float max = 0;
        for(int k=0;k<weight_data.size();k++)
        {
            if(weight_data[k].weight>max)
            {
                max = weight_data[k].weight;
                id = weight_data[k].joint_id;
            }
        }
        mesh_cat.color(i) = color_id[id];
    }
}
```

Et voici le résultat sur le chat :



Extensions :

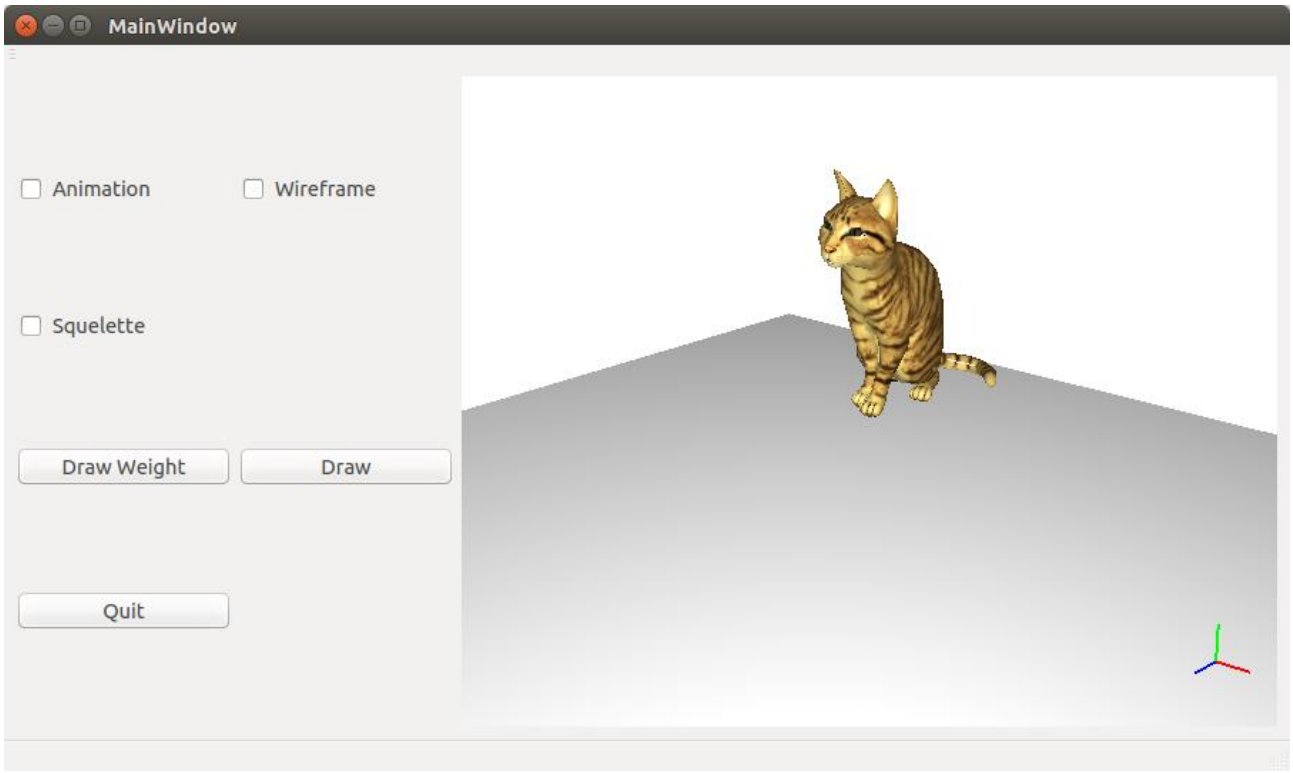
Notre première extension a été d'améliorer l'interface basique que nous avions. Au début, les seuls boutons que l'on disposait étaient le bouton « draw » qui affichait ou non toute la scène et le bouton « wireframe » qui servait à afficher les mesh.

Nous avons rajouté plusieurs fonctionnalités, en voici la liste :

- Possibilité de contrôler l'animation du chat, c'est-à-dire gérer le cas d'animation ou non
- Affichage du squelette ou non. L'affichage du squelette peut bien évidemment être couplé à l'animation pour pouvoir contrôler la déformation du squelette.

- Contrôle du mouvement du chat lorsque celui bouge la tête et cligne des yeux. On peut faire bouger que la tête et les yeux si on appuie sur la touche Y. Pour désactiver cette fonction il faut réappuyer sur la touche Y. Attention, il faut que la fenêtre de la scène soit active pour que cela marche, donc il faut cliquer dans la scène avant de pouvoir utiliser cette caractéristique.

La gestion de ces fonctionnalités se déroule toutes de la même façon. Dans un premier temps, on rajoute un bouton sur l'interface. L'appuie sur ce bouton ou sur une touche du clavier, va entrainer une action, c'est-à-dire un appel à une fonction. Dans toute les différentes fonctions, l'objectif est le même : changer la valeur d'une variable booléenne, en lui appliquant son état inverse. Ce sont ces même booléens qui vont déclencher les caractéristiques que l'on souhaite.
L'animation du mouvement de la tête a été la partie la plus complexe de ces extensions. En effet, il a fallu créer un nouveau frame pour les yeux du chat. Nous avons remarqué que le mouvement de la tête s'effectuait pour des frames compris entre 85 et 135. Nous avons alors gardé que celle-là pour l'animation lorsque l'on appuie sur Y. Il est à noter que l'animation de la tête n'est possible que si l'animation générale est effective.



Finalement, au cours de ce TP, nous avons réussi à coder une déformation d'objet 3D par la méthode du skinning. Plusieurs étapes ont été nécessaire à cette réalisation. Il a d'abord fallu définir un squelette à un objet 3D, puis articulé ce squelette. Nous avons ensuite, déformé la surface de manière à ce qu'elle suive le mouvement du squelette. Deux étapes sont nécessaires : attribution des poids et définition de l'attachement entre chaque sommet de la surface.
Ces différentes étapes nous ont permis de réaliser la déformation d'un cylindre (formé de deux os) et d'une forme plus complexe avec le chat. Nous avons pu apprécier les différences entre un skinning de type rigide et un skinning de type lisse. En effet, la principale différence repose sur la discontinuité lorsqu'il y a déformation. Par exemple, pour la déformation du cylindre, un skinning lisse donne plus une impression de réalité et la courbure semble plus naturelle. De plus la déformation du chat a été un bon exemple pour se rendre compte de l'influence des poids de skinning sur l'appartenance de la surface à un os.