TP2 - Rendu

1 Calcul de l'ombrage d'un maillage

1. Complétez la fonction computeSkinningWeights(Skeleton const& skeleton) de Mesh.cpp pour calculer les poids basés sur la distance euclidienne (cours p.58).
2. Mettre à jour la fonction draw pour afficher votre résultat (une couleur représentant les poids par sommet) avec la même échelle de couleur que la slide 51 en utilisant la fonction scalarToRGB.
3. Mettre à jour la fonction drawTransformedMesh(SkeletonTransformation const& transfo) afin d'appliquer la transformation subie par les os (BoneTransformation, décomposée en rotation et translation) aux sommets du maillage en utilisant les poids que vous avez calculé pour la question précédente.
4. Augmenter n du calcul des poids de skinning. Qu'observez-vous ?

1. Ici, j'ai fait deux fonctions. La première est la projection d'un point sur un segment :

```
Vec3 Mesh::projectPointSegment(Vec3 a, Vec3 b, Vec3 c) {
    double r = Vec3::dot(b - a, b - a);
    if (fabs(r) < 1e-12) return a;
    r = Vec3::dot(c - a, b - a) / r;
    if (r < 0) return a;
    if (r > 1) return b;
    return a + (b - a) * r;
}
```

La seconde, computeSkinningWeights utilise cette fonction dans sa boucle principale :

```
void Mesh::computeSkinningWeights(Skeleton &skeleton, int n) {
    for(size_t i = 0 ; i < vertices.size() ; ++i) {
        MeshVertex &vertex = vertices[i];
        // Suppression des vertex précédents
        vertex.weights.clear();
        double weights = 0.;
        for (size_t j = 0; j < skeleton.bones.size(); ++j) {
            // Calcul du poids
            const Bone& bone = skeleton.bones[j];
            // Les points par lesquels passe l'os
            Vec3 a = skeleton.articulations[bone.joints[0]].position;
            Vec3 b = skeleton.articulations[bone.joints[1]].position;
            Vec3 c = projectPointSegment(a, b, vertex.position);
            double dist = (vertex.position - c).length();
            double w = pow(1/dist, n);
            vertex.weights.push_back(w);
            weights += w;
        }
        // Normalisation
        for (size_t j = 0; j < vertex.weights.size(); ++j) {
```

```
            vertex.weights[j] /= weights;
        }
    }
}
```

2. Il suffit d'utiliser le poids de l'os displayed_bone et de rendre cette couleur :

```
void Mesh::draw(int displayed_bone) const {
    glEnable(GL_LIGHTING);
    glBegin(GL_TRIANGLES);
    for (size_t i = 0; i < triangles.size(); ++i)
        for (size_t j = 0; j < 3; ++j) {
            const MeshVertex &v = vertices[triangles[i].v[j]];
            if( displayed_bone >= 0 && v.weights.size() > 0 ) {
                Vec3 rgb = scalarToRGB(1 - v.weights[displayed_bone]);
                glColor3f(rgb[0], rgb[1], rgb[2]);
            }
            glNormal3f(v.normal[0], v.normal[1], v.normal[2]);
            glVertex3f(v.position[0], v.position[1], v.position[2]);
        }
    glEnd();
}
```

3. Il suffit d'utiliser la matrice de rotation et la matrice de transformation fournie :

```
void Mesh::drawTransformedMesh(SkeletonTransformation &transfo) const {
    std::vector<Vec3> new_positions(vertices.size());

    for(size_t i = 0 ; i < vertices.size() ; ++i) {
        Vec3 p = vertices[i].position;
        new_positions[i] = Vec3(0, 0, 0);
        for (size_t j = 0; j < transfo.bone_transformations.size(); ++j) {
            BoneTransformation &bone = transfo.bone_transformations[j];
            // wij * (r * p + t)
            new_positions[i] += vertices[i].weights[j] *
                (bone.world_space_rotation * p + bone.world_space_translation);
        }
    }

    glEnable(GL_LIGHTING);
    glBegin(GL_TRIANGLES);
    for (size_t i = 0; i < triangles.size(); i++)
        for (size_t j = 0; j < 3; j++) {
            const MeshVertex &v = vertices[triangles[i].v[j]];
            Vec3 p = new_positions[triangles[i].v[j]];
            glNormal3f(v.normal[0], v.normal[1], v.normal[2]);
            glVertex3f(p[0], p[1], p[2]);
        }
    glEnd();
}
```

4. Il suffit d'ajouter une variable globale global_n dans le switch de la fonction key et de l'incrémenter ou de la décrémenter selon la touche appuyée. On l'a déjà fait dans les 2 TP précédents, c'est la même chose, pas forcément besoin de montrer le code ici.