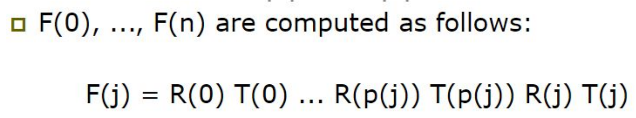
Yingjie Lian
CS 4600
November 5th, 2018

HW4 - Character Animation (Skinning)

1. Skeletal animation

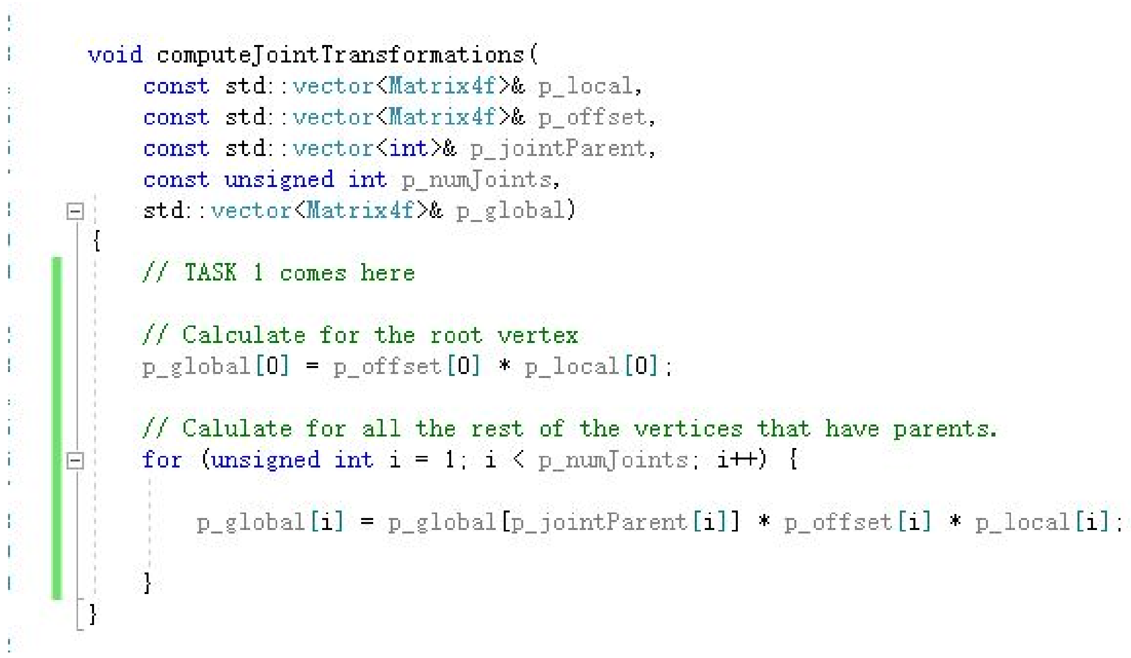
Implementing this function was straightforward. I followed the logic from the lecture slides:



□ F(0), ..., F(n) are computed as follows:

$$F(j) = R(0)\,T(0) \ldots R(p(j))\,T(p(j))\,R(j)\,T(j)$$

And here is my source code:



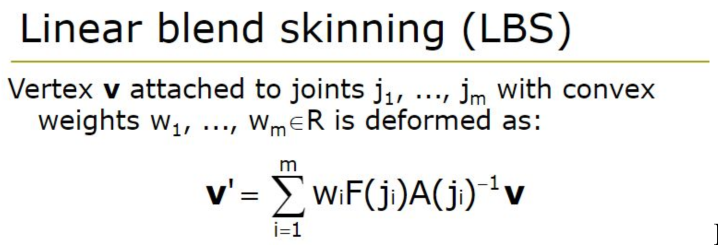
```
void computeJointTransformations(
    const std::vector<Matrix4f>& p_local,
    const std::vector<Matrix4f>& p_offset,
    const std::vector<int>& p_jointParent,
    const unsigned int p_numJoints,
    std::vector<Matrix4f>& p_global)
{
    // TASK 1 comes here

    // Calculate for the root vertex
    p_global[0] = p_offset[0] * p_local[0];

    // Calulate for all the rest of the vertices that have parents.
    for (unsigned int i = 1; i < p_numJoints; i++) {

        p_global[i] = p_global[p_jointParent[i]] * p_offset[i] * p_local[i];

    }
}
```

Since we can assume that j < p(j), we can assume that the parent has stored the joint transformations that it is relative to. The main logic can just be

F(j) = F(p(j)) * R(j) * T(j) where F is p_global, p is p_jointParent, R is p_offset, and T is p_local. See screenshots below to see the skeletal structure and animation.

2 Linear blend skinning

Again, I followed the logic from the lecture slides:

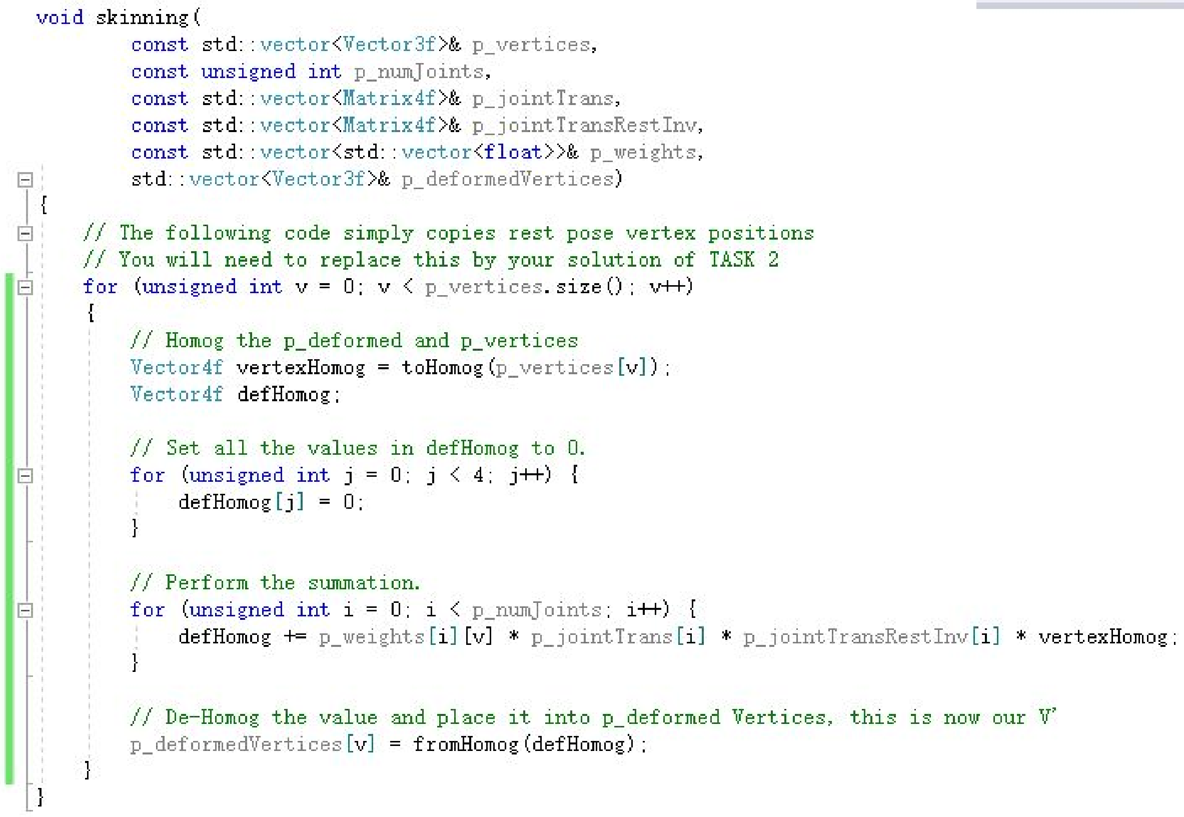


Linear blend skinning (LBS)

Vertex **v** attached to joints $j_1, \ldots, j_m$ with convex weights $w_1, \ldots, w_m \in R$ is deformed as:

$$\mathbf{v}' = \sum_{i=1}^{m} w_i F(j_i) A(j_i)^{-1} \mathbf{v}$$

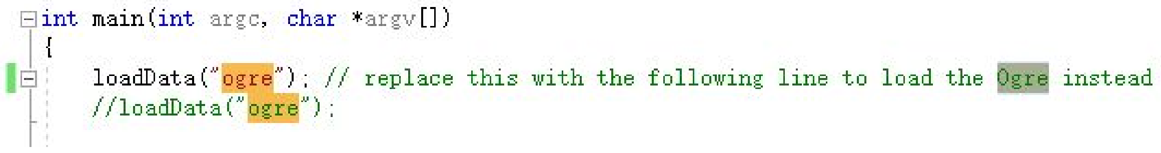
Here is my source code:



```
void skinning(
        const std::vector<Vector3f>& p_vertices,
        const unsigned int p_numJoints,
        const std::vector<Matrix4f>& p_jointTrans,
        const std::vector<Matrix4f>& p_jointTransRestInv,
        const std::vector<std::vector<float>>& p_weights,
        std::vector<Vector3f>& p_deformedVertices)
{
    // The following code simply copies rest pose vertex positions
    // You will need to replace this by your solution of TASK 2
    for (unsigned int v = 0; v < p_vertices.size(); v++)
    {
        // Homog the p_deformed and p_vertices
        Vector4f vertexHomog = toHomog(p_vertices[v]);
        Vector4f defHomog;

        // Set all the values in defHomog to 0.
        for (unsigned int j = 0; j < 4; j++) {
            defHomog[j] = 0;
        }

        // Perform the summation.
        for (unsigned int i = 0; i < p_numJoints; i++) {
            defHomog += p_weights[i][v] * p_jointTrans[i] * p_jointTransRestInv[i] * vertexHomog;
        }

        // De-Homog the value and place it into p_deformed Vertices, this is now our V'
        p_deformedVertices[v] = fromHomog(defHomog);
    }
}
```

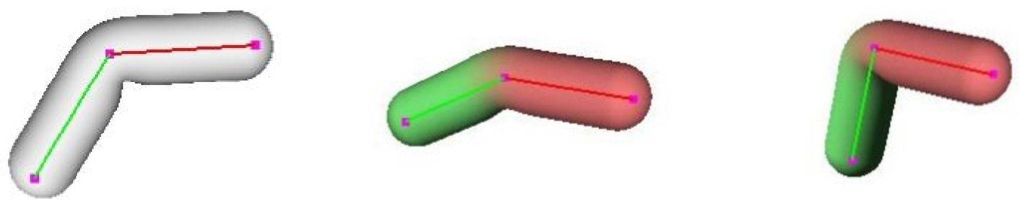
The algorithm simply sums the influences that each joint has on the position of the vertex that is in the skin mesh. You can see the results for the two parts in the images for both the capsule and the ogre below.

By changing “Capsule” to “Ogre” I changed this code:



```
int main(int argc, char *argv[])
{
    loadData("ogre"); // replace this with the following line to load the Ogre instead
    //loadData("ogre");
```

Capsule



Ogre