Skeletal Animation and Skinning

Rui Ying u1234364

Skeletal animation

With the help of the formula `p_global[j] = R(0) T(0) ... R(p_jointParent(j)) T(p_jointParent(j)) R(j) T(j)`, it's obvious we should calculate the final matrix using `while` loop.

```
for (int i = 0; i < p_numJoints; i++) {
    Matrix4f p_global_i;
    p_global_i.setIdentity();

    int j = i;
    do {
        p_global_i = p_offset[j] * p_local[j] * p_global_i;
        j = p_jointParent[j];
    } while (j != -1);
    p_global[i] = p_global_i;
}
```

Using `j` to traverse back from the current joint all the way to the root, we will stop when `p_jointParent[j]` is `-1` which means we have arrived at the root. Multiply matrices along the way and we can get the final matrix.

Linear blend skinning

Following the formula:

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

The final vertice position can be computed as a weighted sum of its transformed position affected by several joints.

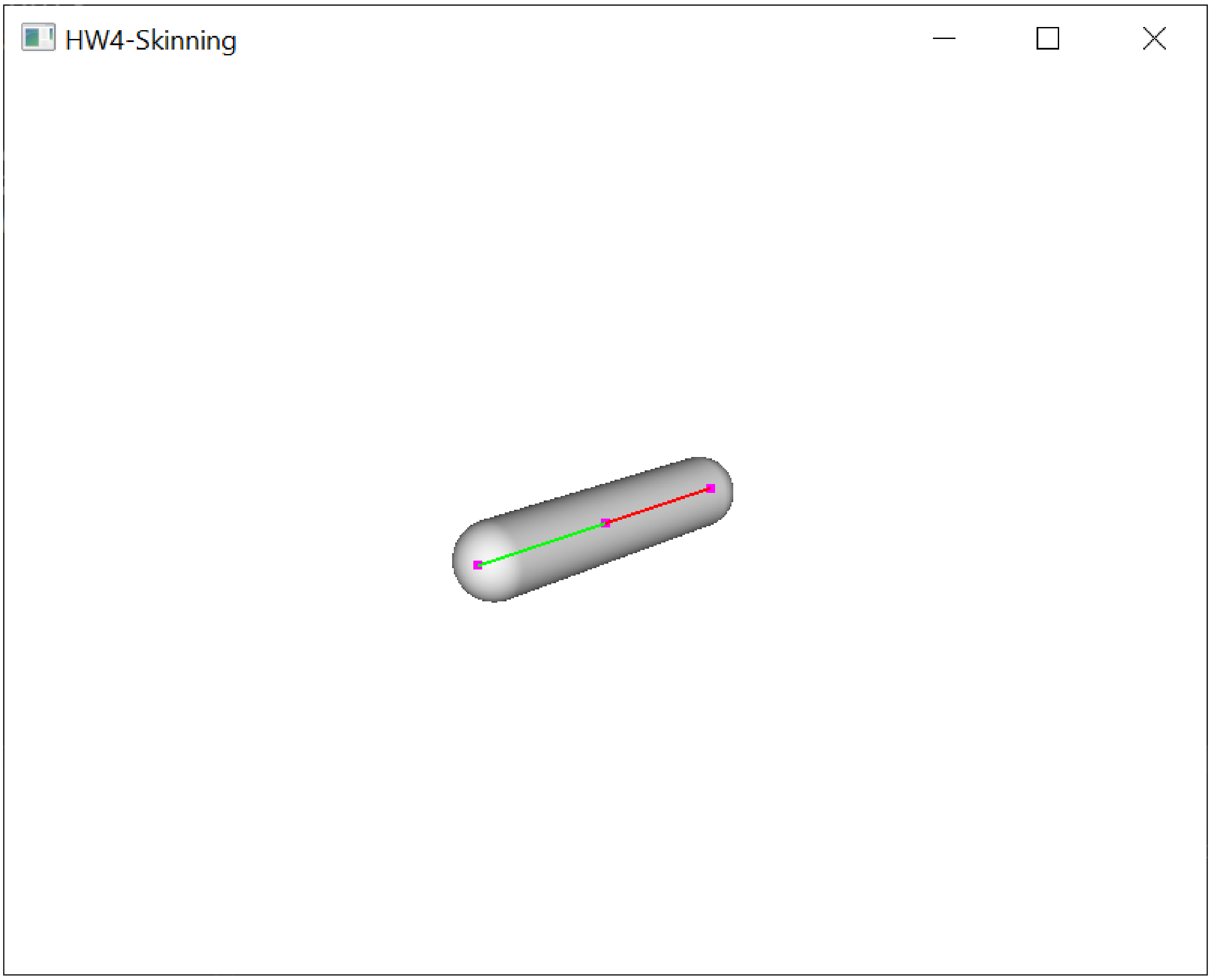
We just need to be sure to transform the vertice to homogeneous one first to match the multiplication and transform the result back.

```
for (unsigned int v = 0; v < p_vertices.size(); v++)
{
    Vector3f deformedVertice = Vector3f(0, 0, 0);
    for (unsigned int j = 0; j < p_numJoints; j++) {
        Vector3f p_vertice = fromHomog(p_jointTrans[j] * p_jointTransRestInv[j] *
toHomog(p_vertices[v]));
        deformedVertice += p_weights[j][v] * p_vertice;
    }
    p_deformedVertices[v] = deformedVertice;
}
```

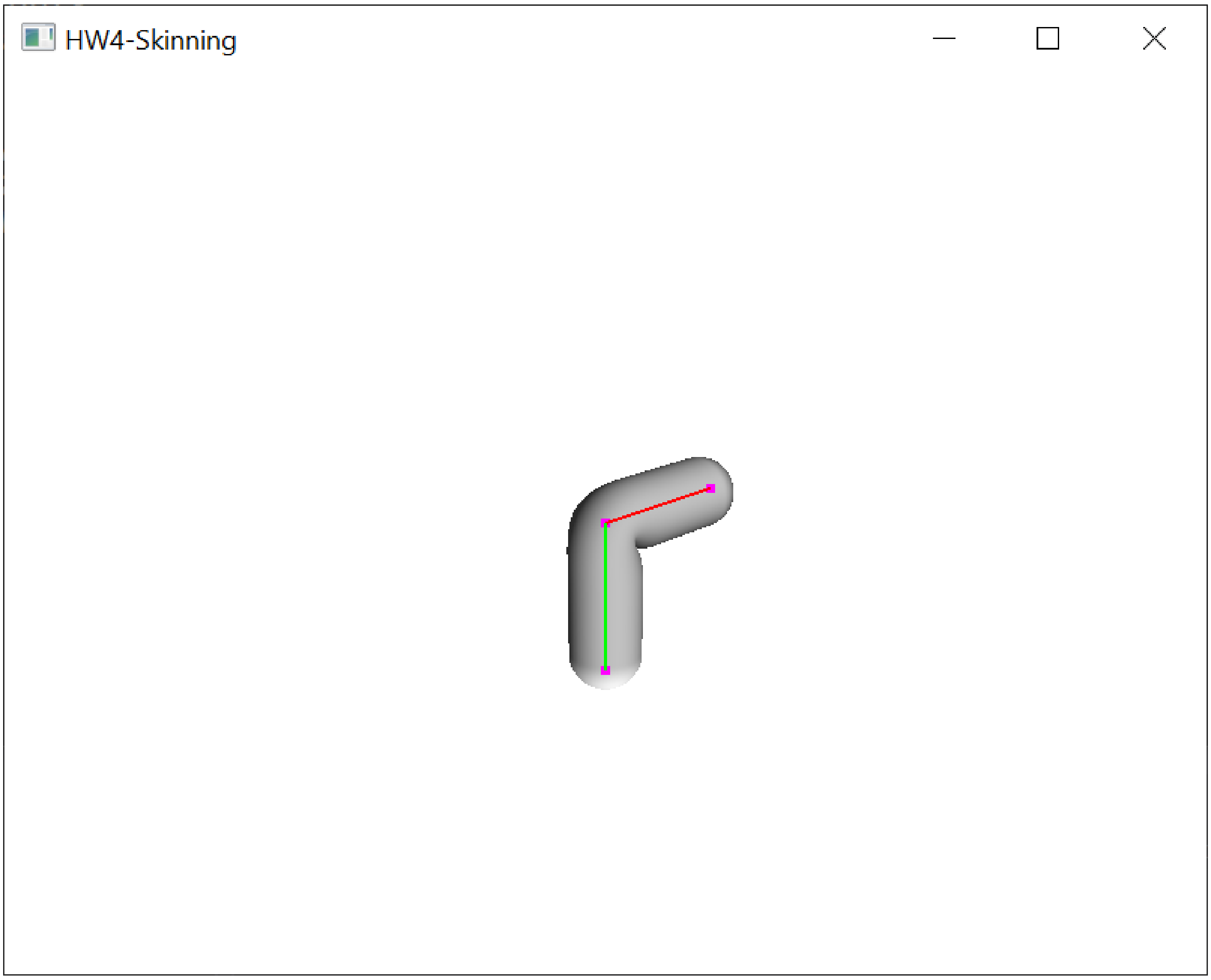
Results

Capsule

Pose0

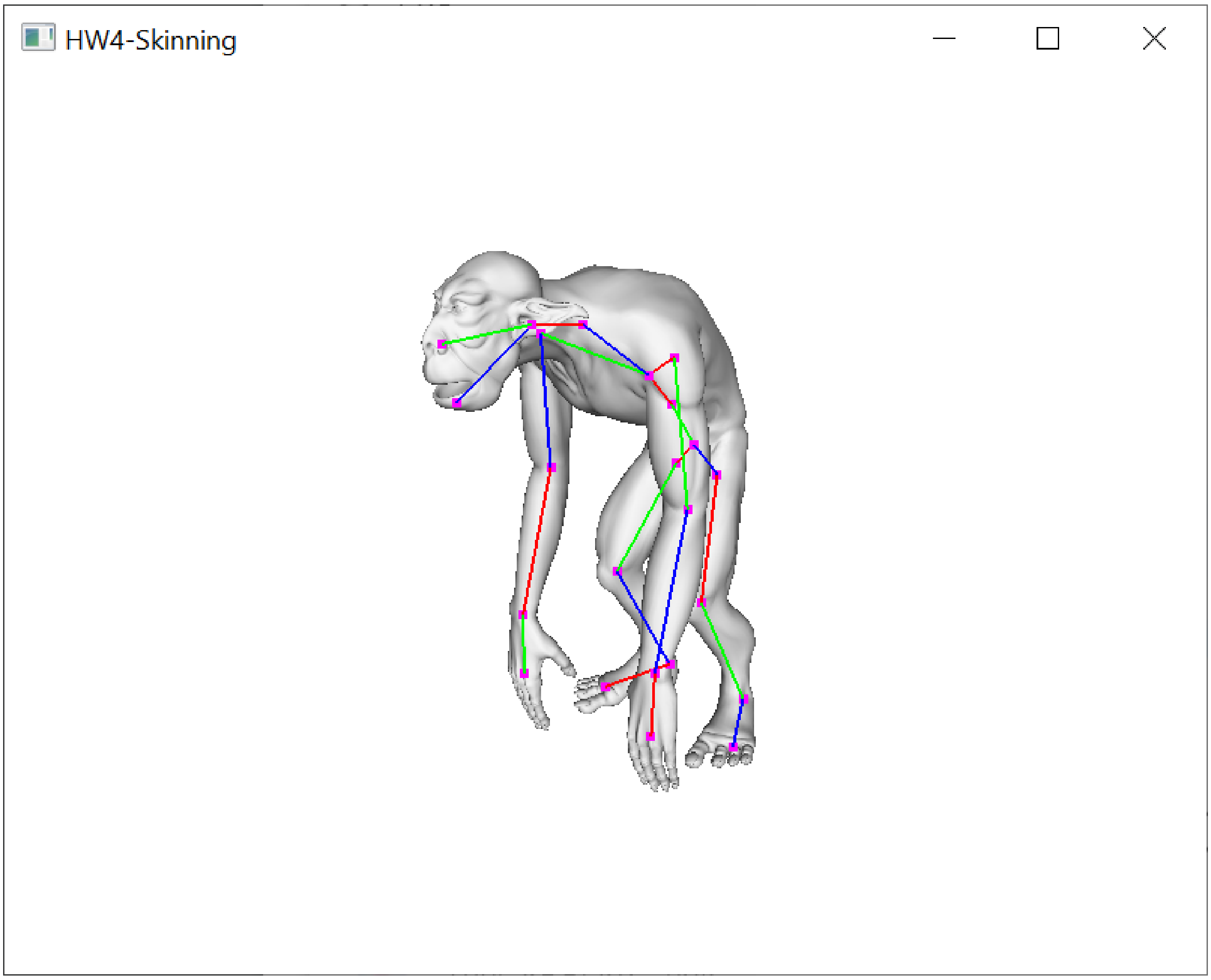


Pose1

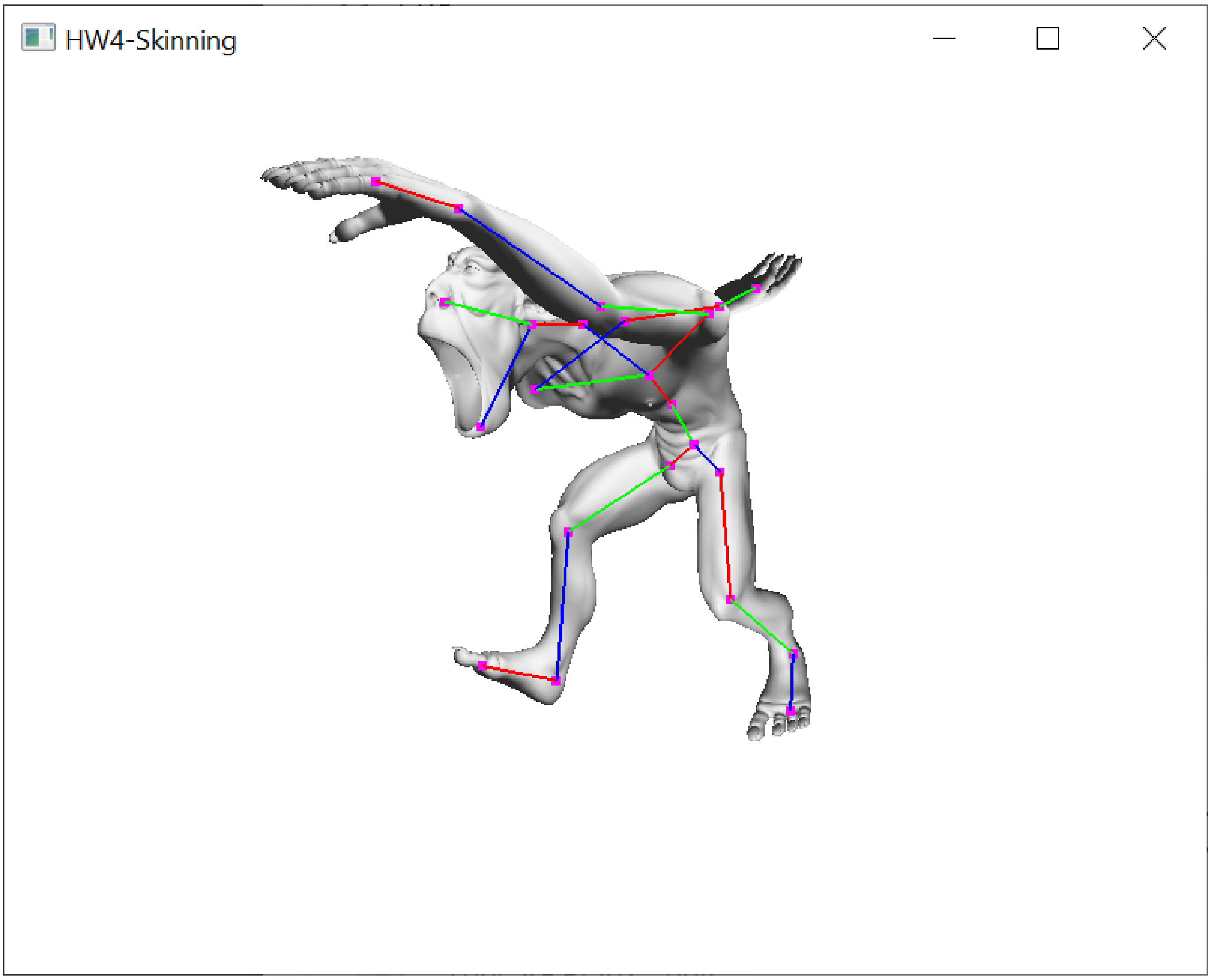


Orge

Pose0



Pose1


HW4-Skinning