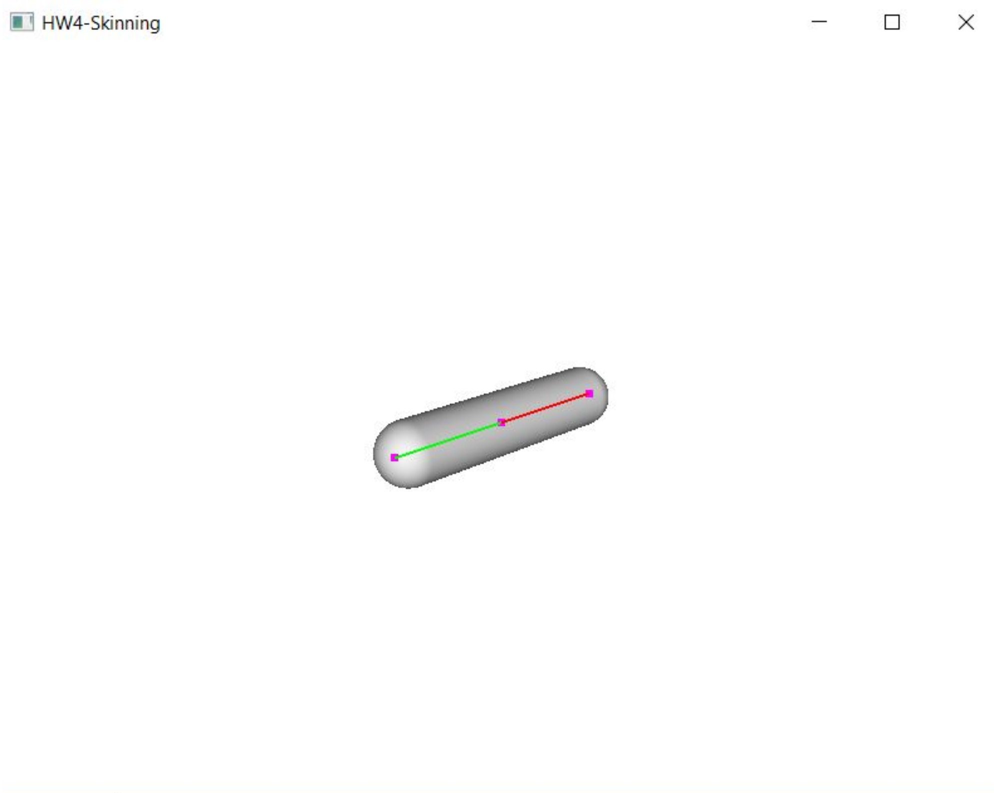
Robert Weischedel
CS 4600
7/14/17

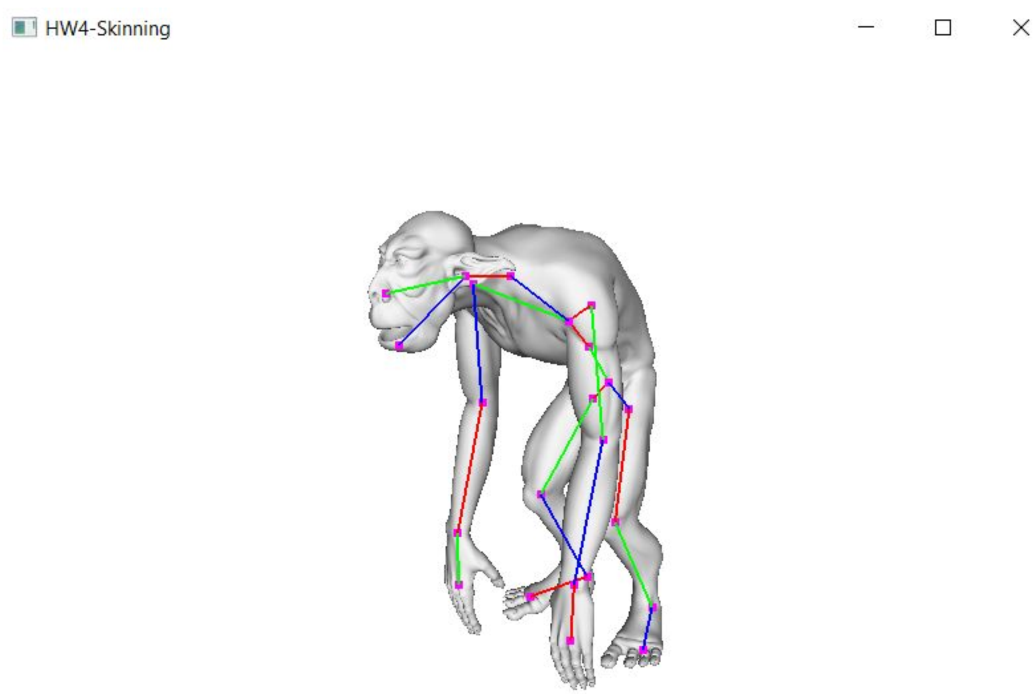
Assignment 4: Skeletal Animation and Skinning

Problem 1 - Skeletal Animations

For this part of the assignment I followed the “trick” algorithm that was given in the lecture 10 slides on slide number 10. I at first considered using the algorithm that was listed in the homework, but implementing it was a bit clunky and I found doing the “trick” algorithm to be much easier, cleaner, and faster. It also didn’t explicitly say anywhere in the assignment description that we couldn’t use it. So for my implementation I used the following algorithm: F(j) = F(p(j) * R(J) * T(j), assuming that the p(j) < j. This works because we only have to consider the case of the root vertex, which is the first vertex in F at index 0. So we need to consider the root vertex as a special case, where we don’t need to calculate the F(p(j)) for it, so it’s algorithm would be: F(j) = R(J) * T(j). But all the rest of the vertices we can perform the algorithm listed above. So basically all we have to do is to iterate through all the number of joints, performing the first mentioned algorithm on each joint. The image below will show the code that I wrote. And below that are the skeletons for the capsule and ogre models.

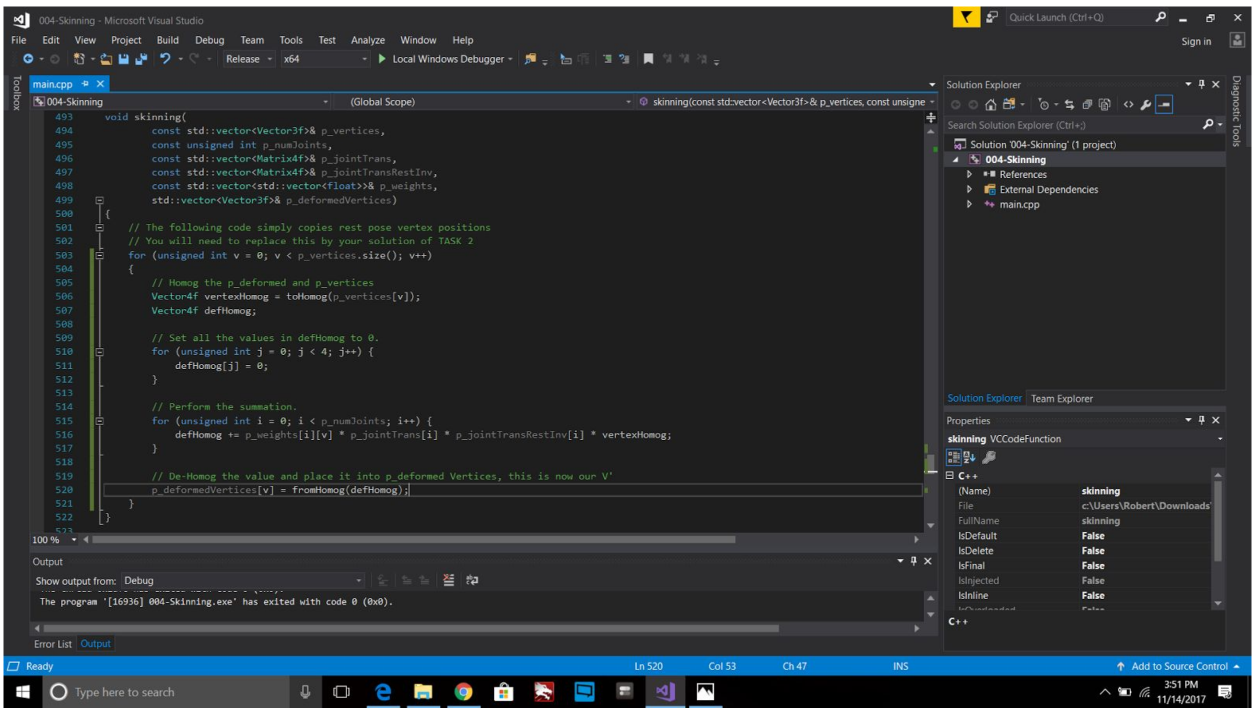
```
void computeJointTransformations(
    const std::vector<Matrix4f>& p_local,
    const std::vector<Matrix4f>& p_offset,
    const std::vector<int>& p_jointParent,
    const unsigned int p_numJoints,
    std::vector<Matrix4f>& p_global)
{
    // TASK 1 comes here

    // 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];

    }
}
```

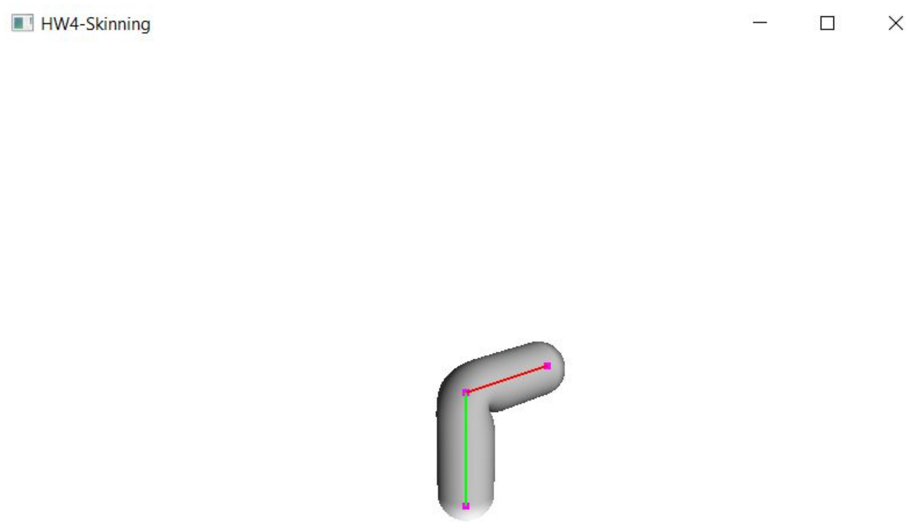

HW4-Skinning

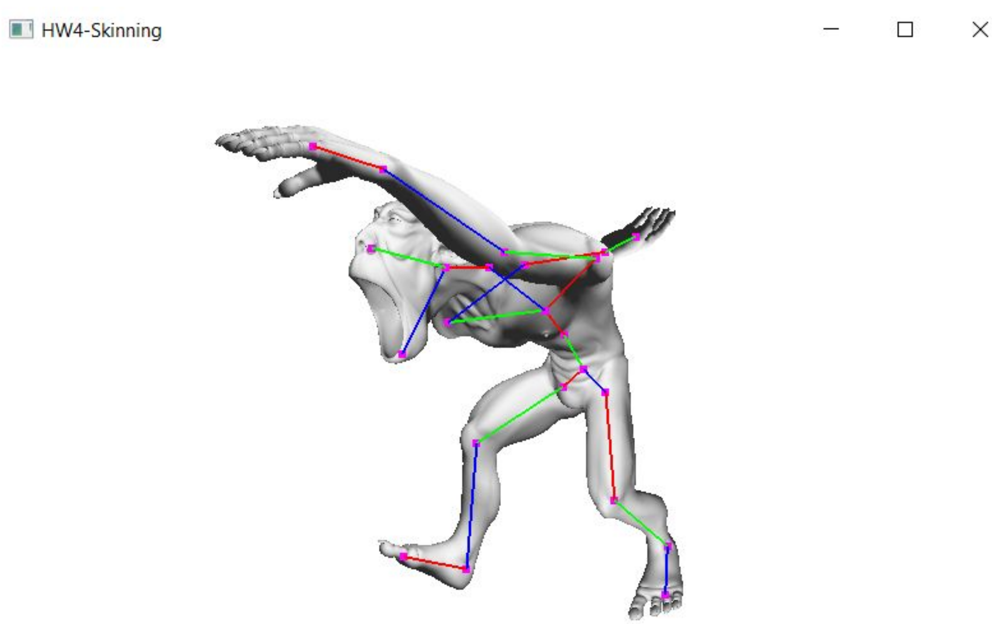

HW4-Skinning

Problem 2 - Linear Blend Skinning

For this problem I followed the algorithm that was given in the lecture 10 slides on slide number 15. I took the algorithm that is found in those slides and adapted it to be used according to the variables that we were given in the method. For this algorithm I had to do one additional step before calculating v', we have to homogenize both the p_vertices before we use them in the summation. I also created another Vector4f and filled it with 0's which I can then use to store the summation of the algorithm in before I de-homogenize the variable and store it i p_deformedVertices. Below is the code for this algorithm, which consists of two for loops, the outer loop iterates through the vertices, and the inner iterates through the number of joints that we have. The inner for loop effectively serves as the summation portion of the algorithm, so that once it is finished we effectively have our v' value and then store it into p_deformedVertices. Below that is the code for the algorithm which shows both the capsule and orge in position 1 or the "animated" pose. I have left the skeleton on for these images as well.

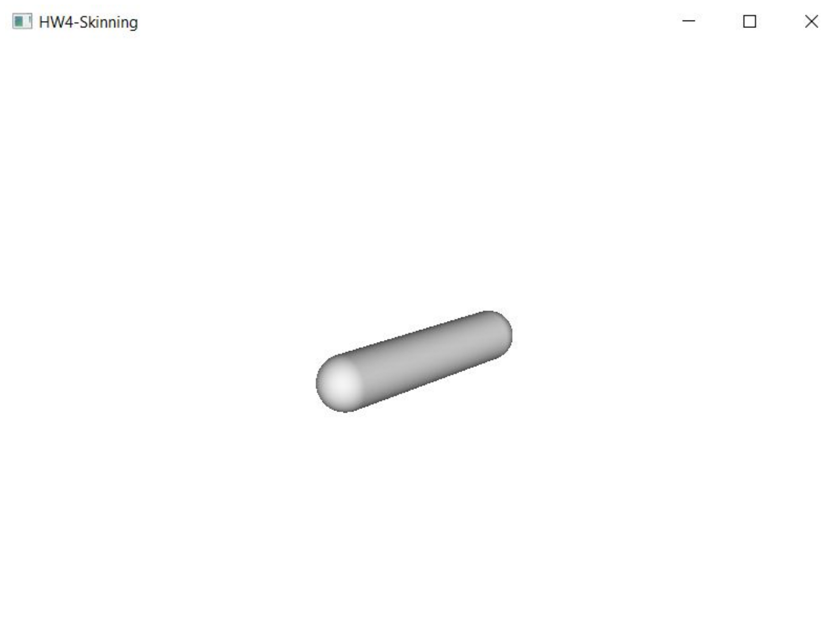


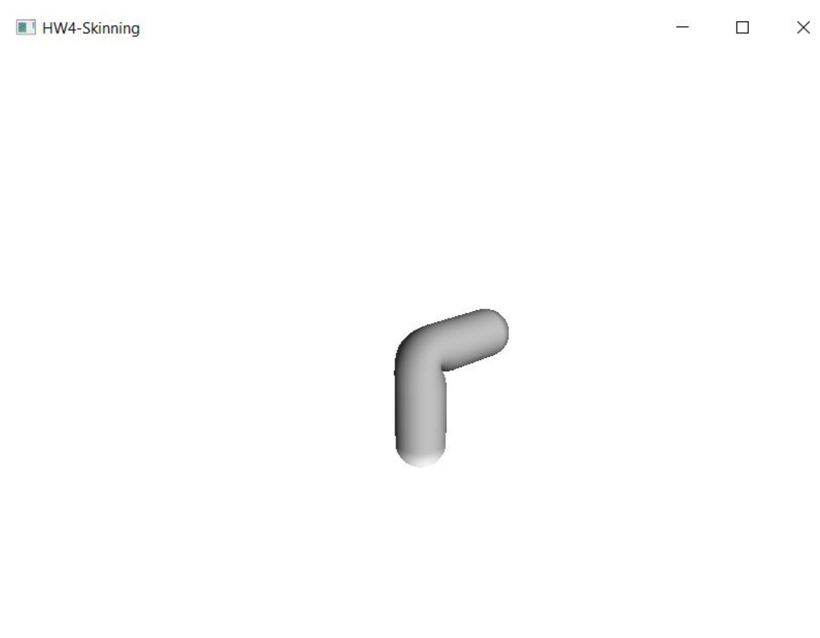

HW4-Skinning

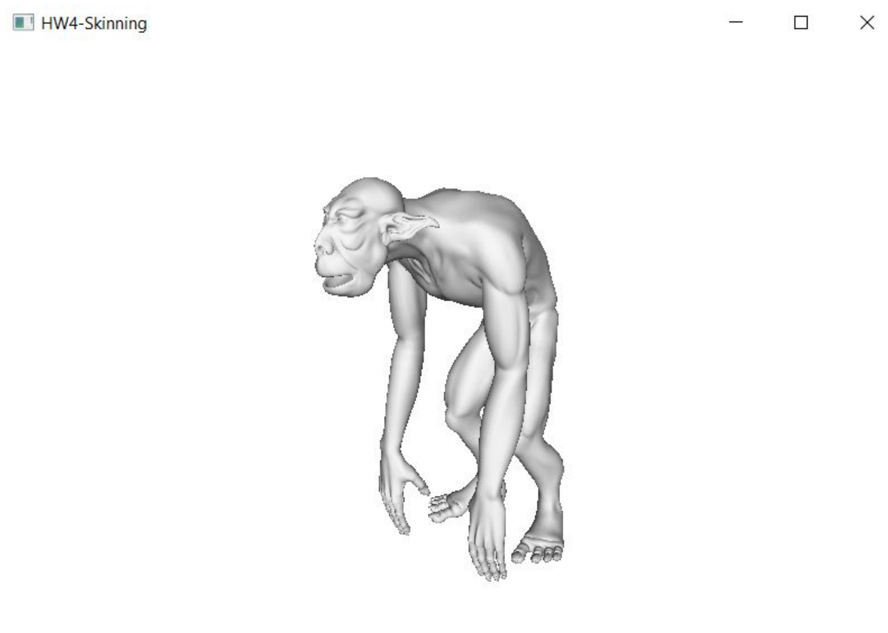

HW4-Skinning

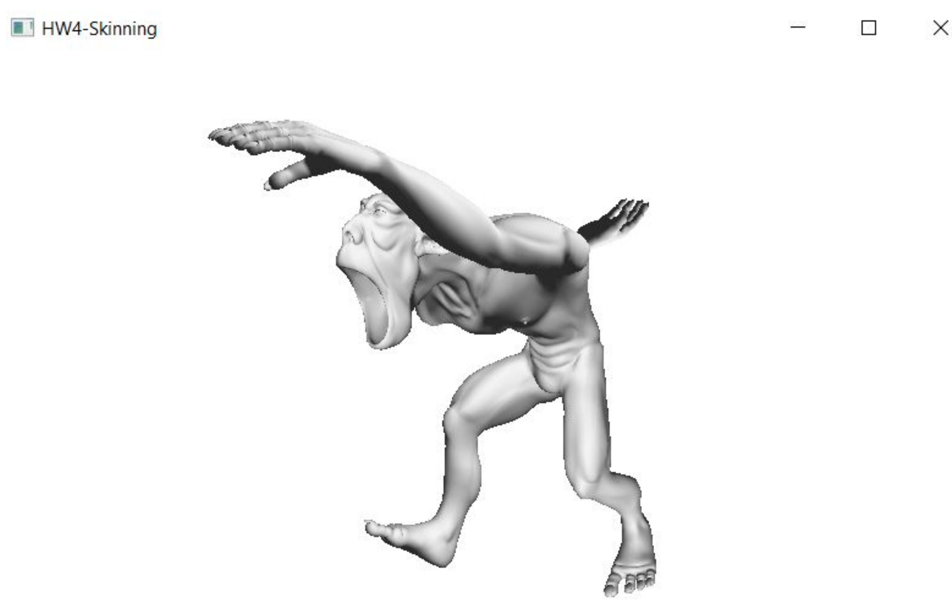
Result

This shows both positions of the capsule and ogre the skeleton turned off.






HW4-Skinning


HW4-Skinning