Real-time Skeletal Skinning with Optimized Centers of Rotation

Problem Statement

Skinning algorithms like Linear Blend Skinning(LBS) and Dual Quaternion Skinning(DQS) are widely used in real time applications. While they have their advantages they have certain well known artifacts like Collapsing elbows, candy wrapper twists for LBS and bulging around joints for DQS. We try to implement a skinning algorithm which corrects those artifacts and that works in real time.

Challenges in Solving the problem

1. **Artifacts:** While LBS and DQS are highly efficient and work in real time they have their drawbacks as mentioned above. There is volume loss artifact in LBS and a bulging artifact in DQS. We need a algorithm that corrects all the artifacts these methods produce and it shouldn’t create any new artifacts on it’s own

2. **Real-time rendering of the scene:** While there are indirect methods that solve the above problem of artifacts, they use multiple passes or some extra calculations which increase the computational overhead of the algorithm. This prevents real-time rendering of the scene. There physical simulations and data driven methods that are more accurate than geometric methods like LBS but they require intervention of general

graphics pipeline and increase computational overhead. So, our algorithm should work in real-time.

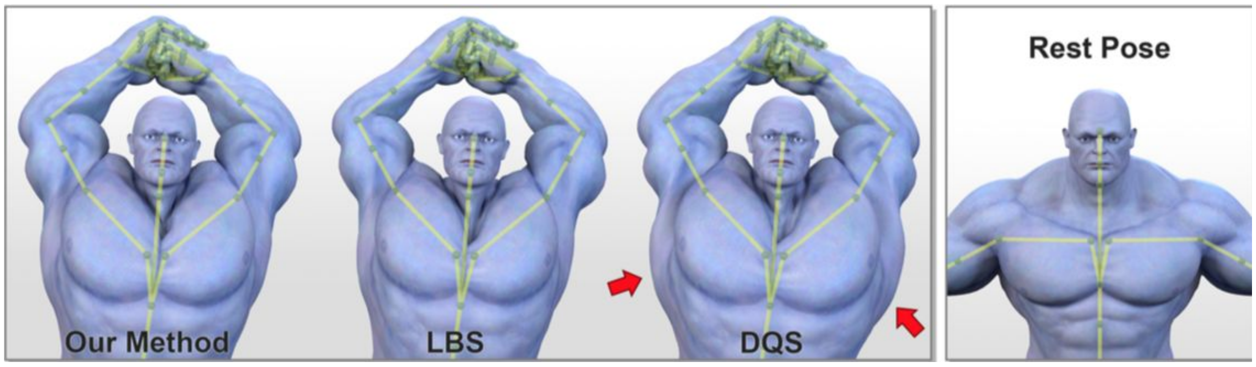
Key Ideas:

Correcting the LBS artifacts: LBS produces artifacts like elbow collapsing and candy wrapper twist. There is a volume loss associated with these artifacts. This implementation relies on two assumptions that local transformations are rigid and vertices with similar skinning weights follow similar transformations. If vertices with similar skinning weights follow similar transformations , it prevents volume loss thus corrects LBS artifacts.

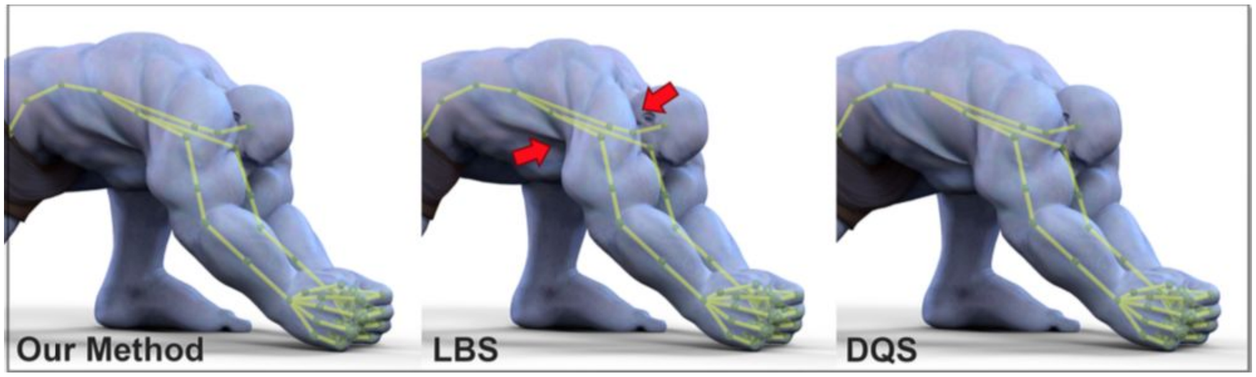
Correcting DQS artifact: DQS has a bulging artifact. This occur during transformation there is only on centre of rotation. In this method we compute optimised Centres of Rotation(CoR) for groups of vertices which have similar skinning weights thus removing the bulging artifacts.

Real-time Implementation: One would think that computing many CoR for every transformation results in computational overhead. But we use a similarity measure to group vertices together and calculate a CoR for them. We can show these CoR doesn’t depend on bone transformations but on skinning weights which know before the transformations. We can precompute these CoR for each vertex and cache them to load them at runtime. This way we can implement the skinning algorithm in real time.

Result Summary:



The bulging artifacts of the DQS are removed.



Skin collapsing artifact of LBS is also not present when rendered with this method.

Implementation:

Library used: **Three.js**

With Three.js, model loading and rendering becomes very easy. It also provides default **LBS** skinning method. It allows us to write custom shaders and add custom uniforms and attributes.

DQS: To implement DQS, custom functions for quaternion and dual quaternion operations are needed. Once these are available, DQS is direct.

For implementing method explained in this paper, we are planning to pre-calculate the optimized centers of rotations and save it in a json file so that the calculation will not be repeated when we render next time. These optimized centers can be passed to vertex shader as an attribute. Rest of the shader code will be fairly simple.