Visualization of Bone Weights

Jörn Loviscach[*]
Hochschule Bremen

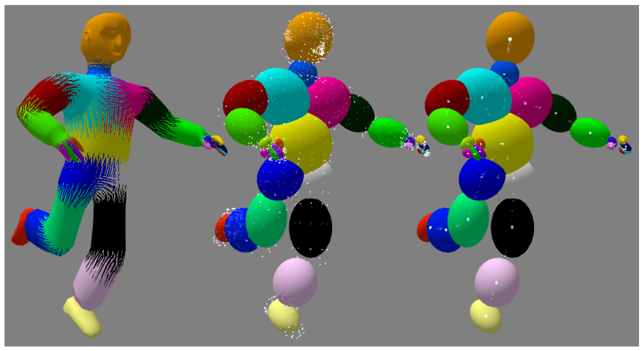

Figure 1: Stripe patterns facilitate to read off the weight ratio (left). Ellipsoids illustrate the distribution of the weights (middle). To display the weight ratio, the vertices may be shifted to the correspondingly averaged spatial position (right).

1 Introduction

This work presents a set of visual helpers that aid in the editing of smooth skinning (a.k.a. skeleton subspace deformation), see Figure 1. These visualizations can easily be integrated with standard 3D software to leverage the skills of trained animators, in contrast to disruptive solutions such as [Mohr et al. 2003].

For smooth skinning, every vertex is equipped with weights that control the influence of the bones. Current 3D software allows to paint bone weights onto the mesh's surface. It displays the weights for one single bone at a time or it applies a different color to every bone and blends these colors appropriately. Even in the latter case, the ratio of the weights, which is the most important quantity, is hard to read off. For instance, it's not intuitively clear which color results from a 30 : 70 mix of blue and yellow. Furthermore, some software does not immediately adjust the painted weights to sum to one for every vertex. In the latter case, vertices may be influenced equally, even though they are displayed with different colors.

2 Method

The user applies weights with a tool that acts more like an airbrush than like a paint brush: The weight of a specified bone is boosted, the other weights are cut to immediately enforce normalization. Like most real-time applications, the system stores up to four bone indices plus four corresponding weights per vertex. If a vertex already contains four non-zero weights and the user tries to assign a fifth bone, the bone with the smallest weight is replaced.

Weight ratios are represented by stripe patterns. Whereas patterns would be easy to apply via a 2D post-process, this work features patterns applied onto the 3D surface, because they enhance comprehension and do not lead to shower-door effects. In contrast to many other pattern types such as dots, stripe patterns facilitate an accurate read-off and allow to emphasize geometric features such as curvature direction. Stripe patterns work well even if three or four bones are blended, see the chest of the character in Figure 1.

[*]e-mail: jlovisca@informatik.hs-bremen.de

Since the mesh may not yet possess texture coordinates, the stripes are generated through a local alignment process in four rendering passes on the GPU. The first pass computes the animated vertex positions and normals and stores them in textures. The second pass computes the local per-vertex direction of maximum curvature. A local plane wave is formed perpendicular to this direction. The third rendering pass continuously adjusts the phases of these local plane waves to minimize the misfit between neighbors. This relaxation method is adapted from [Loviscach 2006]. The fourth rendering pass determines the phase of the blended linear waves per pixel and converts it to the corresponding color of the stripe pattern. To keep the clarity of the colors, no antialiasing is applied.

The user can interactively control the width of the stripe pattern. Furthermore, the width depends on the distance to the viewer: All stripes have a similar width on the screen. Thus, near parts of the surface show more detail; distant parts do not suffer from aliasing due to small patterns. To reveal further details of the weight assignment, the phase of the stripe pattern may be animated.

The bones can be displayed as ellipsoids approximating the distribution of the positions and weights of the vertices. This allows for instance to easily check for symmetry between left and right limbs. The ellipsoids are described and rendered as deformed unit spheres. The deformation of the unit sphere's vertices and the corresponding deformation of its normals are handled in a vertex shader.

The colors of the bones are optimized for maximum local contrast, where "locality" is defined through the approximating ellipses. This operation is applied continuously, so that the colors adapt to the user's actions. The high contrast allows to apply lighting on top of the coloring, as opposed to the display in standard software.

The ratio of the weights can be illustrated spatially: The vertices may be moved to the weighted average of the centers of the bones according to the weights. Thus, one can quickly spot outliers. The motion is done as an animation to show which vertex goes where.

3 Results

Even though four rendering passes are required to display the stripe patterns, this method runs at highly interactive speed: The first three passes write into textures and produce only one pixel per vertex of the original mesh. The method also lends itself to real-time texture generation, e. g., for hatching. The representation by ellipsoids and by weighted position averages helps to "debug" smooth skinning.

References

Loviscach, J. 2006. Wrinkling coarse meshes on the GPU. *Computer Graphics Forum 25*, 3, in press.

Mohr, A., Tokheim, L., and Gleicher, M. 2003. Direct manipulation of interactive character skins. In *SI3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, 27–30.