Extraction of skinning data by mesh contraction with Collada 1.5 support

Martin Madaras[1]
Supervisor: Tomáš Ágošton[2]

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Introduction

A frequently used approach for animation and modification of 3D models is based on creating articulated hierarchical structures - skeletons. Skinning data as a skeleton tree and weights can be either assigned manually or computed from an input mesh. The first option is most often chosen by artists, although sometimes it is unnecessary and time consuming. The skeleton has to be created (or used from templates), rigged into the mesh and influence weights have to be set. In this paper we present how to automatically compute the hierarchical skeleton and skinning weights from an input mesh and use them in a skinning animation using Collada 1.5 as export format between our application and a graphic animation software such as 3D Studio Max, Blender or Maya.

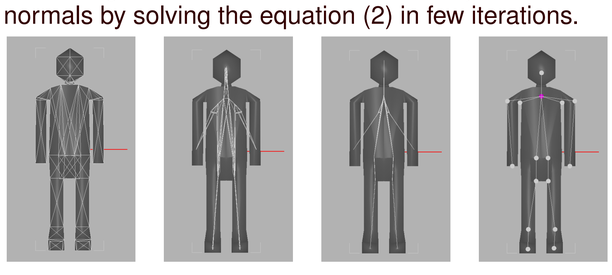
Graph conversion

For running our graph algorithms, we need to have the input mesh as one connected object. The object needs to be converted into a 3D graph, defined by an edge matrix E. It is quite common that models are composed of more objects. These objects appear to be connected visually, but edges in the model structure between these objects are not defined. Also the opposite problem has to be considered. There can be edges defined in an input mesh, which connect parts that should not been connected. These edges are remains of the work of graphic designers or artifacts after format conversion and therefore have to be excluded. Using a simple depth-first search suitable joining distances can be found to connect or disconnect all graph components. This condition joins the closest vertices in neighbouring components which creates one-component graph.

Joining (left) and splitting (right) of the mesh is needed.

Mesh contraction

For contraction of the generated mesh graph we use a contraction algorithm using Laplacian smoothing proposed by [Au et al., 2008]. The algorithm does not alter geometry connectivity, is noise sensitive and works directly on mesh geometry. Geometry contraction removes details from the surface applying Laplacian smoothing. Vertex positions are smoothly contracted along their normals by solving the equation (2) in few iterations.



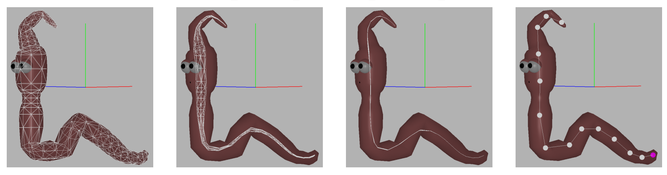
Results on a low resolution model.

The Laplacian smoothing operator (1) was introduced by [Desbrun et al., 1999] for surface smoothing. Laplacian smoothing operator L is $n \times n$ square matrix. This operator is applied on n vertices in vector V as a filter. Term LV approximates curvature flow normals, so solving $LV^{'} = 0$ removes normal components of vertices and contracts the geometry, resulting into a new set of vertices $V^{'}$.

$$L_{ij} \begin{cases} w_{ij} = \cot \alpha ij + \cot \beta ij & \text{if } (i,j) \in E \\ \sum_{(i,k) \in E}^{k} -w_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where:
E – is set of edges generated in previous section during graph conversion process
$\alpha ij, \beta ij$ – are the opposite angles corresponding to the edge (i,j) [Desbrun et al., 1999]

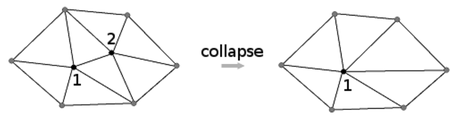
$$\begin{bmatrix} \mathrm{W}_L L \\ \mathrm{W}_H \end{bmatrix} V' = \begin{bmatrix} 0 \\ \mathrm{W}_H V \end{bmatrix} \quad (2)$$



Results on a middle resolution model.

Skeleton construction

The contracted mesh graph from the last iteration is simplified. A greedy algorithm is used to select the most important control points. During the collapsing process, for each control point, the collapsed vertices into this control point are stored in a hash map. Vertices are merged into control points and every control point is shifted into the center of its local mesh area. For each control point, the volume of the mesh region the control point represents in the original mesh is computed. The control point, which represents the largest volume is chosen as the skeleton root.



The half-edge collapse $(\tilde{v_2} \rightarrow \tilde{v_1})$.

The first step in the mesh graph simplification is to collapse all edges, whose vertex distance is smaller than a predefined threshold. Vertices in such pairs can be interpreted as the same control point, because the distance between them is very small and the influence over the mesh vertices is almost the same. The second step is to collapse edges which are the least important. For every edge the cost value is computed and edges with minimum sampling cost [Garland and Heckbert, 1997] are collapsed. For simplicity we apply half-edge collapse. The half-edge collapse $(i \rightarrow j)$ merges vertex i to vertex j and removes all the faces that are incident to the collapsed edge. This step is repeated so many times that in the end we will have the desired number of control points.

Skinning weights

Skinning indices are computed by finding a set of closest control points to each vertex. The geodesic distance is used as a distance measure. A distance between each pair of vertices on the mesh graph from $0th$ iteration (after conversion from an input mesh) is calculated and stored in matrix D. For each control point C_k, the closest mesh graph vertex is found, for instance v_j. Then, the resulting geodesic distance between the control point C_k and the mesh vertex v_i is computed as a sum of distance between v_j and v_i on the mesh graph calculated by Floyd-Warshall algorithm and the euclidean distance between C_k and v_j.

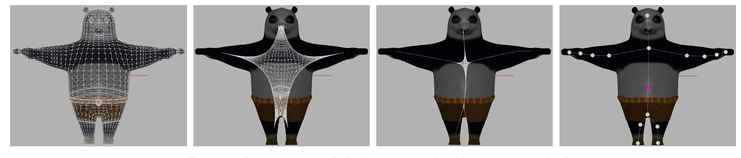
$$\mathrm{gd}(i,k) = \mathrm{D}[i,j] + \mathrm{d}(C_k, v_j) \quad (3)$$

where:
$\mathrm{d}(C_i, v_k)$ – is euclidean distance between the mesh vertex v_j and k^{th} control point C_k

Weights are assigned in a way that weight sum for each vertex is equal to 1.0. Fractions are constructed in a way that weights are indirectly dependent on the geodesic distance. This construction guarantees that the closer control points will have greater influence over mesh vertices than the further ones. The geodesic distance is a real-value function defined on the mesh surface. Because the function varies smoothly along the mesh, the resulting weights are fluently distributed over the mesh regions.

$$\mathrm{weight}(i,k) = \frac{\frac{1}{\mathrm{gd}(i,k)}}{\sum_{k' \in S} \left(\frac{1}{\mathrm{gd}(i,k')} \right)} \quad (4)$$

where:
$\mathrm{gd}(i,k)$ – is geodesic distance between the mesh vertex v_i and k^{th} control point C_k
S – is the set of control point indices controlling one vertex



Results on a high resolution model.

References

[Au et al., 2008] Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., and Lee, T.-Y. (2008). Skeleton extraction by mesh contraction. *ACM Transactions on Graphics*, 27(3).

[Desbrun et al., 1999] Desbrun, M., Meyer, M., Schroder, P., and Barr, A. H. (1999). Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of ACM SIGGRAPH 99*.

[Garland and Heckbert, 1997] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. *International Conference on Computer Graphics and Interactive Techniques*.