Real Time Skin Deformation with Bones Blending

Ladislav Kavan
Charles University
Faculty of Mathematics and Physics
Malostranske nam. 25
118 00 Prague 1, Czech Republic
lkav8604@ss1000.ms.mff.cuni.cz

Jiří Žára
Czech Technical University in Prague
Faculty of Electrical Engineering
Karlovo nam. 13
121 35 Prague 2, Czech Republic
zara@fel.cvut.cz

ABSTRACT

Skeletal animation with vertex blending is a popular method to model believable 3D characters. Its main advantage is its speed that allows real-time deformation even on low-level hardware. However it has some drawbacks as well: for some movements it produces non-natural postures like twisting elbows. We present a new method called bones blending that can be used as an alternative to vertex blending. Bones blending has a power to overcome the artifacts of vertex blending.

Keywords

Skeletal animation, digital skin, character modeling.

1. INTRODUCTION

Skeletal animation is a technique of skin deformation using associated bones. It has been developed mainly for the purposes of the game industry because of its modest hardware demands. The drawback is its performance in visual quality: the basic scheme propagates the non-smooth skeleton posture to skin in a straightforward manner. This has been improved by vertex blending which smoothes the skin, although not satisfactory for all (and even natural) skeleton postures.

For the purposes of this article we will define three steps of the skeletal animation evolution:

1. Basic skeletal animation
2. Vertex blending
3. Improved vertex blending

Articles dedicated to game programmers that explain the basic skeletal animation and vertex blending are [Lan98] and [Lan99]. An application presenting the basic skeletal animation method is based on public domain library [Por01]. [Web00] also noticed the artifacts of vertex blending and presented an improved vertex blending system. Our method called *bones blending* is an alternative to improvements of vertex blending. It does not use the concept of vertex blending - it builds only on the basic skeletal animation.

A different approach to real-time human body modeling, which is based on special software called BodyBuilder, is presented in [Kal98].

The next section describes the basic skeletal animation principles and its improvement by vertex blending. Our alternative to vertex blending is presented in Section 3. Section 4 compares discussed approaches in terms of visual results and rendering times.

WSCG SHORT PAPERS proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

2. SKELETAL ANIMATION

Let us consider the character model is composed from several elements:

- Joints – represented by homogenous matrix (usually only rotation and translation)
- Bones – the segments connecting two joints. The bone is usually identified with the joint where it begins.
- Skeleton – a tree. The nodes of the tree are joints and the edges are bones. The root of the tree is addressed as the root joint.
- Skin – mesh of triangles. Every vertex of every triangle is attached to one and only one bone.

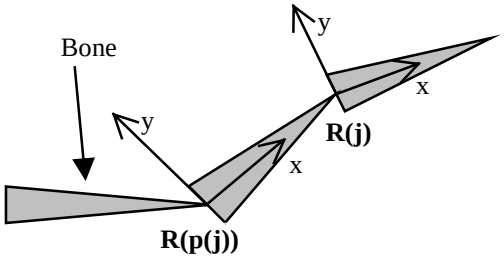

Figure 1. Reference skeleton joint transformation.

Both skeleton and skin are designed in some reference position (usually the "crucifixion" pose). The idea of skeletal animation is to move only the skeleton and automatically propagate this movement to the skin, which is displayed.

To be more precise, assume that the parent of joint j is denoted p(j) and the root joint has index zero. Further assume that R(j) is the 4x4 homogenous matrix describing the transformation from joint p(j) to joint j in the reference skeleton position. Now the transformation from the world coordinate system to the local frame of joint j can be written as

$$\mathbf{A}(j) = \mathbf{R}(0) \dots \mathbf{R}(p(j))\, \mathbf{R}(j)$$

The meaning of R is shown in Fig. 1. Here R(j) expresses the translation and rotation of coordinate systems from p(j) to j. To apply a movement to character we use transformation T(j) that expresses the rotation of the bone starting in joint j. Translations are not considered because they do not result in realistic skeleton deformation - the only exception can be the root joint. Its translation means translation of the whole skeleton. T(j) matrices can be computed for example by interpolation of keyframes or by inverse kinematics. Finally we define the final transformation of the joint with movement applied to it and all its ancestors:

$$\mathbf{F}(j) = \mathbf{F}(p(j))\, \mathbf{R}(j)\, \mathbf{T}(j)$$
$$\mathbf{F}(0) = \mathbf{R}(0)\, \mathbf{T}(0)$$

From this formulation the computation of F(j) is straightforward with some kind of tree traversing algorithm. The hierarchical structure of a skeleton is advantageous for the automatic propagation of rotations T to all the children. Now we can write what happens to vertex v from reference posture when transformed to vertex v' in the actual posture. Assume v is attached to a bone beginning at joint j:

$$\mathbf{v}' = \mathbf{F}(j)\, \mathbf{A}(j)^{-1}\, \mathbf{v}$$

The interpretation of the above formula is that vertex v is first transformed from the reference position to normalized position (the joint centered at the origin). The reason for doing this is that we need the rotation T(j) to run upon the joint j. Following multiplication by F(j) means: first apply T(j) and then return the vertex from the normalized to the actual location. Note that when all T(j)'s are identities then the vertices are not translated at all. The multiplication $A(j)^{-1}v$ can be pre-computed when the model is loaded since it depends only on the reference posture. For nicely commented source code of this algorithm see [Por01].

Spherical Linear Interpolation

The transformations T(j) can be determined for example by interpolation of keyframe transformations. However, the naive way of matrix interpolation (element by element) performs very badly – it lacks the intuitive idea of rotation interpolation.

The Spherical Linear Interpolation, known as SLERP is usually explained with connection to quaternions, although it is possible to compute it just from matrices (using conversions to axis-angle representation). The SLERP between two orthogonal matrices K and L can be defined as

$$\mathbf{K}(\mathbf{K}^{-1}\mathbf{L})^t,\ t \in <0,1>$$

Note that for t=0 it gives K and for t=1 we obtain L. For non-integer t we define the power of a matrix as follows: let $M=K^{-1}L$. M is also an orthogonal matrix, i.e. a rotation. We can extract an axis **a** and angle ϕ of M, see for example [Ebe01] (this point is trivial when using quaternions). Now M^t can be defined as rotation around axis **a** by angle $t\phi$. Opposite conversion from the axis-angle representation to matrix and multiplication by K yields the result.

Vertex Blending

The basic skeletal animation scheme does not produce a smooth shape nearby joints due to sudden change in attached bones as seen in Fig. 2. The color of vertices indicates assignment to bones. Note that the assignment is not clear for highlighted vertices.

Our goal is to model smooth skin deformation using a non-smooth, segmented skeleton. Allowing attachment of one vertex to more than one bone and averaging the results can do this. More formally, if vertex v is attached to joints $j_1, \dots, j_n$ then the formula for resulting vertex position is

$$\mathbf{v}' = \sum_{i=1}^{n} w(i)\, \mathbf{F}(j_i)\, \mathbf{A}(j_i)^{-1}\, \mathbf{v}$$

where w(1), …, w(n) are weights of attachment to mentioned joints such that w(i) sum to 1. Then the resulting location of vertex v' is just the convex combination of individually transformed vertices.

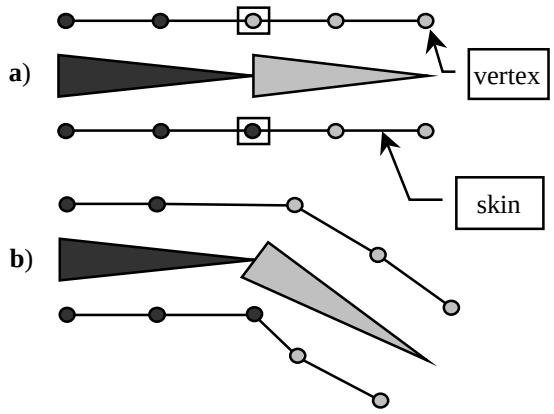

Figure 2. Basic skeletal animation: a) reference position, b) after rotation

w(i) can be interpreted as influence of joint j_i on the resulting vertex position. The 'n' is usually a small number from 2 to 4. Skin deformation with vertex blending is shown in Fig. 3.

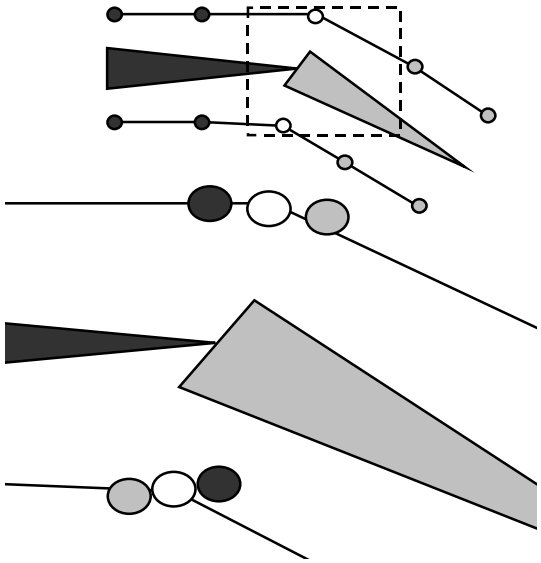

Figure 3. Vertex Blending.

In Fig. 3 the vertices with non-clear assignment (see Fig. 2) were assigned to both bones with equal weights 0.5 and 0.5. The white vertices represent the results of averaging which is demonstrated in the detail view.

Although vertex blending produces smooth skin deformation, it can have problems when the rotation T(j) is very large. A typical example is a twist of the elbow about 180 degrees (see Fig. 4). Imagine the bones model an arm and the light bone represents the forearm. The weights of the highlighted vertices are 0.5 and 0.5 as above. In Fig. 4b the elbow is twisted 180°. When averaging the highlighted vertex locations after transformation they both end up in the same position: the joint itself. The overall result is not acceptable at all (see 3D example in Fig. 8a).

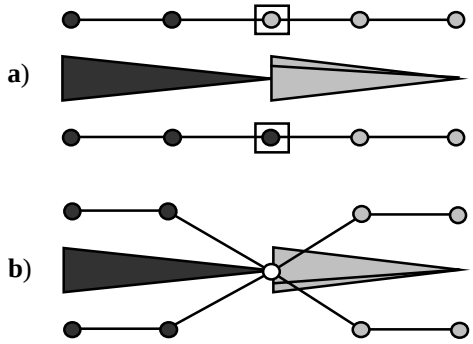

Figure 4. Twisting Elbow Problem: a) reference position, b) elbow after 180° twist

One may object that the 180 degrees elbow twist is non-natural posture. However even for natural poses the artifacts were observed: especially in joints with a large range of motion such as shoulder (see natural posture in Fig. 11a).

3. BONES BLENDING

The problems of vertex blending arise from the fact that averaging is done on the vertices. If we could arrange the blending on a lower level, such as bones, it would give better results. As mentioned earlier the joints are essentially rotations. The main idea of bones blending technique is not to apply the whole rotation at once, but to perform a blend of previous rotation and the current one. Blending of two rotations is easily accomplished by SLERP. The only tricky part is where to take the interpolation parameter. We shall denote it by *t*.

Intuitively, it should be connected to the distance along the bone in the reference posture. To specify the distance along the bone we use the normalization matrix $A(j)^{-1}$ that transforms the vertices from the reference position to the position with joint *j* centered at the origin. If the following joint is *k* then the bone segment is determined by R(*k*), more exactly by the translation component of R(*k*) – we shall denote this vector **n**. Then the distance of the normalized vertex along the bone can be defined as distance from the plane with normal **n** containing origin. It can be computed by the dot product:

$$t' = < \mathbf{v}_0, \mathbf{n} >, \mathbf{v}_0 = \mathbf{A}(j)^{-1}\,\mathbf{v}$$

where $\mathbf{v_0}$ is the normalized version of the given vertex **v**. The *t'* is positive for points in the **n** direction, zero for points incident to the plane and negative for points behind the plane (in the –**n** direction). Let us assume we have two values:

- t_{min} – the value of *t'* where only the parent joint's rotation is applied
- t_{max} – the value of *t'* where only the child joint's rotation is applied

Now it is straightforward to derive the interpolation parameter:

$$t = \frac{t' - t_{\min}}{t_{\max} - t_{\min}}$$

clamping values to interval <0,1>. Note that *t'* and therefore *t* depend only on the reference posture, so they could be pre-computed. For good results this method assumes the limbs for bones blending are outspread in the reference posture (which is often the case).

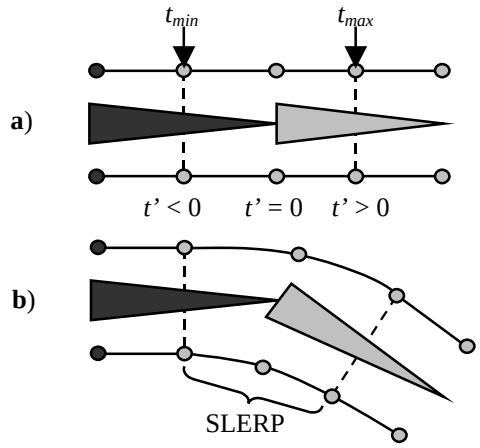

Figure 5. Bones Blending: a) vertex assignment and parameters, b) after rotation

Parameter tuning

The parameters t_{min} and t_{max} are objects of editing and tuning, as well as vertex-bones assignment we have not mentioned yet. The vertices are usually attached to bones manually – using an editor, although some heuristic algorithms also exist [Web00]. The general rule is to assign the vertices to the nearest bones. If there is more than one near bone (as in Fig. 2a for highlighted vertices) they are weighted according to the respective distance and vertex blending is applied.

For bones blending the situation is somewhat different. The vertex assignment in this case is connected with t_{min} and t_{max} parameters. The general rule can be now: assign all the vertices that should be modified when joint *j* rotates to the bone beginning in joint *j*. It means that all the vertices around the joint *j* should be connected to bone beginning in *j* instead of the bone that ends in *j*. Those newly assigned vertices will be those that give negative *t'*. The vertices with non-negative *t'* are still assigned to joint j, unless they are subject of blending with the following bone – it depends on parameter t_{max} (see Fig. 5a).

If t_{max} - t_{min} is large then the bend is broad, spread over all attached vertices. On the other hand when it is too small the result is similar to basic skeletal animation. The optimal values should be tuned in an editor. An example of correct parameter settings for bones blending is in Fig. 6.

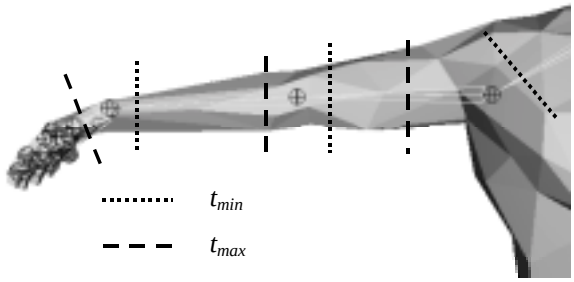

Figure 6. Our choice of t_{min} and t_{max} parameters in the reference posture

Triangle Subdivision

Bones blending brings little improvement if the model mesh is too coarse - the triangles are large and the effect of smooth bone deformation vanishes (the smooth curves drawn in Fig. 5b can be thought as a limit case for an infinite number of vertices). It is no problem if there are the original surfaces (Bezier, B-spline) the model was built of.

However, we often know only the triangular mesh location (and even worse optimized/decimated to the reference position). In this case it is necessary to subdivide large triangles in the most strained regions – around the joints with wide range of motion. The *triangle connected to joint j* will be a triangle with all vertices attached to *j*. The triangles for subdivision due to joint *j* can be defined as the triangles connected to joint *j*. There are two basic methods for triangle subdivision: 1:3 and 1:4 triangles (see Fig. 7).

Each of them has its pros and cons: 1:4 produces T-vertex discontinuities, which should be corrected by induced division of neighboring triangles. On the other hand 1:3 subdivision tends to produce "long" triangles (i.e. with high ratio of circumscribed and inscribed circle radius). To conclude: 1:4 offers higher quality and 1:3 gives faster animation.

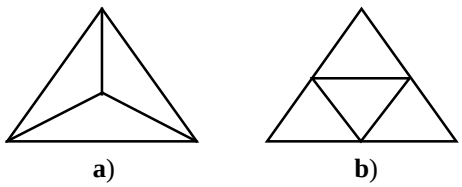

Figure 7. 1:3 (a) and 1:4 (b) triangle subdivision.

Another question is which joints should be selected for bones blending and then possibly for subdivision. Apparently not all the joints deserve the special treatment by bones blending: [Web00] states that for

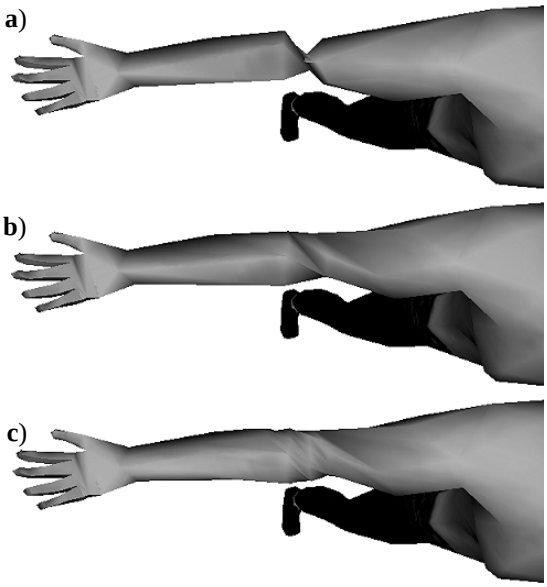

Figure 8. Elbow twisted 180°: a) vertex blending, b) bones blending applied to original mesh, c) bones blending applied to 1:4. subdivided mesh

rotations[1] less than 60° the vertex blending produces good results. This problem as well as the problem of t_{min} and t_{max} parameters tunning is of rather artistic nature and could be solved during the virtual humanoid design in an interactive editor.

4. IMPLEMENTATION

Fig. 8 compares the visual quality of discussed methods. The cost increase of bones blending against skeletal animation is only one SLERP, dot product and linear scaling per one vertex. The SLERP on matrices clearly dominates with its 58 additions, 77 multiplications and 1 division [Ebe01]. However it is not necessary to compute the full SLERP for every vertex; it is possible to determine the axis **a** and the angle ϕ of rotation only once for each joint and cache the results for every vertex attached to this joint. Then in every vertex it is sufficient to perform only multiplication $t\phi$ and conversion from axis-angle to matrix representation: 13 additions and 15 multiplications [Ebe01]. In our application these optimizations need not to be implemented.

Performance was measured on an original triangle mesh and on a model with right arm subdivided 1:4. Both models were rendered with vertex blending (No BB) and with bones blending applied on the right arm (BB). The results are in Table 1 and Table 2.

In our implementation the parameters t_{min} and t_{max} do not influence the simulation time. However, it would be possible to cancel the SLERP when the interpolation parameter t is either 0 or 1. Then the number of vertices between t_{min} and t_{max} (with t satisfying $0 < t < 1$) would affect the rendering time.

	Original	**1:4 subdiv**
No BB	16.57	23.31
BB	19.04	32.7

Table 1. Rendering time in milliseconds

	Original	**1:4 subdiv**
Vertices	1507	2065
Triangles	1372	1930

Table 2. Size of input data

The rendering time increment for the original model is acceptable when compared to improvement seen in Fig. 8a, b. The comparison is slightly worse for the subdivided models but it still retains nice real-time properties. It means that the subdivision is useful only if visual quality is a priority.

To measure the improvements of the triangle subdivision we define the *curvature of a triangle T* that has three neighboring[2] triangles T_1, T_2 and T_3 as the maximum of angles inclined by normals of (T, T_1), (T, T_2) and (T, T_3). Then the total *curvature of a triangle mesh relative to joint j* can be defined as the maximum of curvature of all the triangles connected to joint *j*. Lower curvature number means smoother mesh. As the model approaches to the mathematically smooth surface, the curvature (relative to any joint) approaches to zero.

A graph of our model's curvature relative to the elbow joint is in Fig. 9. The curvature values were measured in the animation of the elbow bend: in the beginning the arm is completely stretched – in fact it is the reference posture. In the end the elbow is rotated about 135°. We see that the subdivided model

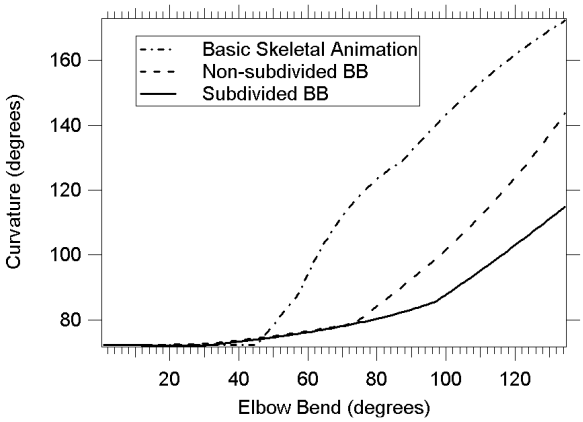

Figure 9. Curvature measurement

[1] We can more precisely say "both swing and twist component of a rotation"

[2] With a common edge

indeed leads to better curvature than the non-subdivided one.

Application

The bones blending technique was successfully used in our project of virtual reality training of fencing (see snapshot in Fig. 10). In this application the armed (right) arm is controlled by inverse kinematics and it is necessary to model a broad range of joint motions for the realistic fencing simulation.

The results of pure vertex blending were not acceptable: in the reference posture the right hand's palm is facing down. Therefore when forced to posture as in Fig. 11a, a 90° twist in the shoulder joint was necessary, producing artifacts when rendered with vertex blending. Our technique's result is in Fig. 11b.

Figure 10. Application: Virtual Fencing

5. CONCLUSION

The bones blending algorithm is useful when one needs to model real-time virtual humanoids with a wide range of joint movement, typically for a sport simulation. The troubles of collapsed digital skin, like in the twisting elbow problem, do not occur in bones blending because the original bone to vertex distances remain the same during the animation.

The proposed bones blending method could not compete with high-level animation systems that use layered humanoid model composed of bones, muscles and fat tissues. On the other hand it can be easily included in the existing skeletal animation system when artifacts of vertex blending become a problem. This upgrade is modest in development time and it does not lead to serious drawbacks in performance. Another advantage is the method's scalability: finer

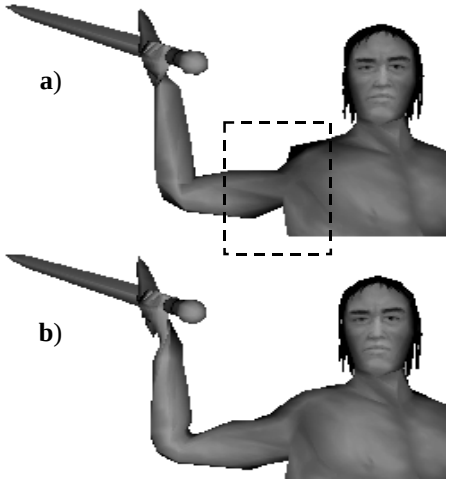

Figure 11. Real-life posture: a) vertex blending, b) bones deformation with 1:4 subdivision

mesh provides automatically finer results. Bones blending is a method of choice for complex real-time character modeling. Future work can be oriented on considering other methods of bones blending than SLERP (e.g. a higher order spherical interpolation) as well as non-linear derivation of the interpolation parameter *t*. The subdivision step could be performed automatically depending on the level of detail.

6. ACKNOWLEDGEMENTS

The 1st author was supported by the Czech Ministry of Education under project MSM 212300014, the 2nd author was supported by the European Union under project IST-2001-32184 and by the Czech Ministry of Education under project LN00B096. We would also like to thank to the CharacterFX development team for providing a character model and animation software.

7. REFERENCES

[Ebe01] Eberly, D. 3D Game Engine Design. 2001

[Kal98] Kalra, P., Magnenat Thalmann, N., Moccozet, L., Sannier, G., Aubel, A., Thalmann, D. Real-time Animation of Realistic Virtual Humans. IEEE Computer Graphics and Applications,Vol.18, No.5, pp.42-55, 1998

[Lan98] Lander, J. Skin Them Bones: Game Programming for the Web Generation. Game Developer Magazine, pp. 11-16, May 1998

[Lan99] Lander, J. Over My Dead, Polygonal Body. Game Developer Magazine, pp. 17-22, Oct 1999

[Por01] Porter, B. Skeletal Animation Tutorial. PortaLib3D, http://rsn.gamedev.net, Feb 2001

[Web00] Weber, J. Run-Time Skin Deformation. Intel Architecture Labs, 2000