SIMPLIFIED MUSCLE DYNAMICS FOR APPEALING REAL-TIME SKIN DEFORMATION

LEE KENG SIANG[1] AND GOLAM ASHRAF[2]

School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543

ABSTRACT

We propose significant simplifications in muscle modeling and simulation to facilitate real-time anatomical skin deformation for full-body articulated characters. The muscle shape is a function of an animated quadratic Bézier action curve and control rings derived from it. The action curve is uniformly sampled to derive control rings driven by a scaled sinusoidal equation to model fusiform shapes. A single spring is attached between the central control point and the midpoint vector between the extreme control points of the Bézier action curve. Care is taken to stabilize local coordinates for each muscle vertex to enable glitch-free skin deformation. The character's polygonal mesh is smooth-skinned using a two-layered approach: first to the joints, and then to the muscle vertices. A typical 4000-vertex character skinned with sixty four 72-vertex muscles is able to run on an average CPU at 60-80 fps. Our main contribution is the simplified dynamics-driven curved action-axis, which enables economical and expressive muscle animation.

1 INTRODUCTION

Appealing character deformation could play an essential role in believable gaming and virtual world experience. Character traits like sluggishness, power, etc. can be emphasized much better with anatomy-driven deformation than current real-time skeletal deformation of surfaces. Such visual emphasis of traits affords better immersion and connection to virtual characters. However, the high simulation cost of anatomy-driven deformations has limited their application in the real-time domain. On the other hand, video games have been steadily pushing the quality envelop, given rapid advances in hardware and increasing consumer appetite. Realistic and appealing deformation for game characters is therefore a relevant topic for the modern game industry.

In this paper, we address the lack of efficient dynamics methods to achieve anatomy-driven body deformation in real-time. We propose a flexible muscle representation that can be economically driven by single-spring dynamics. We adapt the smooth-skinning metaphor to blend conventional skeleton-driven deformation with our muscle-driven deformation. The resulting muscle-based skin contouring and oscillation (commonly termed as jiggle) can generate significant appeal to real-time characters.

The focus of this paper is not on accurate muscle modeling but on simulating efficient local deformations and oscillations which increase visual appeal without violating significant physical constraints. We demonstrate how a variety of prominent muscles can be modeled with the same generic representation, aided with a set of intuitive design tools.

---

[1] Student

[2] Supervisor

2 PREVIOUS WORK

Multi-layered anatomical models have been organized into skeleton, muscle, fatty tissues and skin layers. The skeleton is first defined using joint design tools. Muscle primitives are then added, with the origin and insertion points of each muscle constrained to the appropriate joints. The skin surface could be handcrafted or "draped" over the muscles using contour detection and implicit functions. Different levels of dynamics and kinematics constraints control the deformation of these models, using procedural volume preservation, elasticity etc.

(Chadwick, 1989) used Free-Form Deformations (FFD) to approximate muscle deformation. (Moccozet, 1997) applied Dirichlet Free-Form Deformation method (DFFD) for hand deformation. It is generally hard to emulate muscles in complex regions such as the shoulders, since the relationships between FFD deformations and high-dimensional joint rotations are not well-defined. Finite Element Method (FEM) models accurately deform volumes defined by connected cells that impart torsion forces on each other using physics of energy transfer (Wu, 2001). However they are usually too complex for real time simulation.

(Scheepers, 1997) proposed several surface representations for muscle models that could be deformed with volume preservation formulae. They modeled simple fusiform muscles with ellipsoids and multi-belly sheet-like muscles with multiple instances of these fusiform muscles. (Wilhelms, 1997) uses similar volume-preserved cylinders, where each muscle segment's influence on skin vertices is analytically calculated. The fusiform model was further enhanced with spring constraints (Nedel, 1998), stable local coordinates (Aubel, 2001) and curved action axis (Zuo, 2003). Though these enhancements increase the generality of the fusiform muscle's shape and motion, they add significantly to the computation complexity. Real time performance of these models has been demonstrated using only a few muscles at a time, without any skin deformation.

(Turner, 1993), (Wilhelms, 1997) have modeled sophisticated elastic skin models that are first deformed by muscle slices, and then readjusts skin triangles based on spring forces on each edge. Though such elastic models achieve several interesting effects like skin sliding and stretching, the simulation cost is again too high for real time performance. (Wilhelms, 1997) choose to statically bind the skin layer and move it relative to underlying tissue deformation. A different approach is adopted in (Aubel, 2000) who use per-frame mesh regeneration by sampling the implicit surface envelop of underlying tissue.

We conclude that a fair amount of research has been dedicated towards creating multi-layered anatomical models. However, most of these models possess negligible dynamics attributes, and do not easily afford real-time deformation of surfaces. For real-time dynamics simulation, pre-computation and model simplification have been used with mixed success to tackle large dimensional, computation-intensive problems. In this paper, we propose an adaptation to the surveyed musculature modeling methods for efficient dynamics simulation.

3 MUSCLE CREATION

3.1 REPRESENTATION USING QUADRATIC BÉZIER ACTION CURVE

A quadratic Bézier curve requires 3 control points (P_0, P_1, P_2) which can be mapped to the muscle origin, mid-point, and insertion point (P_{org} , P_{mid} , P_{ins}), as shown in Fig. 1. During muscle creation, the origin and insertion joints of the skeleton are chosen, and P_{org} and P_{ins} are parented to these 2 joints respectively. These two control points can be in any arbitrary position in their respective parent's coordinate frame. By default, P_{const} is constrained to the midpoint of P_{org} and P_{ins}, but the user can shift this point to change the muscle curvature.

The second control point of the Bézier curve is represented by a freely-hanging dynamic control point P_{mid}, which is attached to P_{const} via a spring. The Bézier action curve is created from the control points (P_{org}, P_{mid} , P_{ins}), so that the muscle moves with soft-body dynamics as the joints (and consequently P_{const}) move in space.

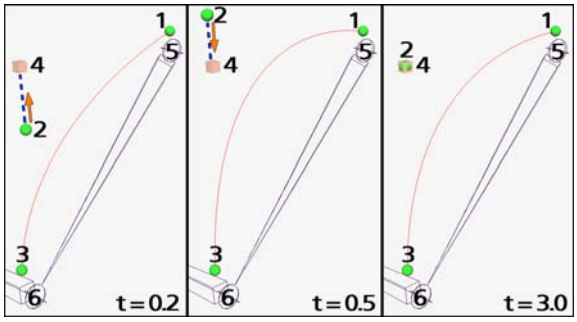

Figure 1: The three control points P_{org} (1), P_{mid} (2) and P_{ins} (3) of the muscle action curve, the constrained control point P_{const} (4), and the two attachment joints (5 and 6). Notice how P_{mid} moves towards P_{const} when perturbed, causing the Bézier curve to deform smoothly.

3.2 Construction of Local Coordinate Frame

We sample N_{cs} uniformly spaced points along the curve, where N_{cs} is a user specified number. Each of the points represents the center C_i of each cross-sectional slice i of the muscle. The user first defines an up-vector v (default: joint's local y-axis) and then the local coordinate frame for each cross-section at point p(u) is constructed:

$$\hat{a}_i = \frac{p'(u)}{|\,p'(u)\,|} \qquad [1]$$

$$c_i = \hat{a}_i \times \hat{v} \qquad [2]$$

$$b_i = \hat{c}_i \times \hat{a}_i \qquad [3]$$

The local x, y and z axes are then $\hat{a}_i$, $\hat{b}_i$ and $\hat{c}_i$ respectively.

3.3 Construction and Control of Muscle Shape

We characterize the fusiform muscle with two main properties: bulge size S and taper T. The bind pose radius r_i of each muscle cross-section is calculated as:

$$r_i = S \sin^T\left(\frac{i-1}{N_{cs}-1}\pi\right) \qquad [4]$$

Each muscle cross-section i has a user-defined N_{div} number of muscle vertices it will control. Each of these vertices v_{ij} (where j=1, 2,…, N_{div}) has an offset o_{ij} from the cross-section center C_i. This offset is stored in the local coordinate system of cross-section i. Initially, each of the vertices v_{ij} for cross-section i will get the offset value assigned as $o_{ij} = r_i$. The local coordinate frame of each muscle vertex v_{ij} is a simple addition of its offset o_{ij} with the local coordinate frame of cross-section i. During simulation, the cross-section offsets are proportionally scaled by volume-preserved shape distortion functions (see Sec 3.4).

3.4 Volume Preservation and Deviation

We can calculate each new offset o_{ij}' as the length l of the muscle changes, by using a modified proportional relationship derived from the standard ellipsoid formulae:

$$o_{ij}' = (1 + B\sin(\frac{\pi}{2}\Delta L))\sqrt{\frac{l_{rest}}{l_{new}}}\, o_{ij} \quad [5]$$

$$\Delta L = \begin{cases} \dfrac{\Delta l}{l_{rest}} & if\ \Delta l > 0 \\ \dfrac{\Delta l}{l_{rest}} \times \dfrac{1}{R} & otherwise \end{cases} \quad [6]$$

where l_{rest} is the rest length of the muscle and l_{new} is the new muscle length at a certain time. Δl is the normalized length difference (l_{new} - l_{rest}). B is the bulge multiplier, which is used to scale the sinusoidal function used for non-linear increase in offset, while R is the bulge-to-stretch-ratio used to counter the rapid thinning of the muscle when it is stretched.

3.5 Muscle Dynamics

A spring is attached to the control points P_{mid} and P_{const}. This spring produces a force at each time-step Δt that causes P_{mid} to move towards P_{const}. First, the force exerted on P_{mid} (Hooke's Law, Eqn. (8)) and subsequently its acceleration (Newton's Law, Eqn. (9)) is calculated:

$$\Delta x(t) = x_{mid}(t) - x_{const}(t) \quad [7]$$

$$F_{spring}(t) = -k_s \Delta x(t) \quad [8]$$

$$\ddot{x}_{mid}(t) = F_{spring}(t) / m_{muscle} \quad [9]$$

where $x(t)$ represents control point position at time t, k_s is the user-defined spring constant and m_{muscle} is the user-defined mass of the muscle.

The Verlet integration method is used to find the position of P_{mid} at the end of each time-step. We have used a modified version which allows for a simple damping effect, where f is the damping factor:

$$x_{mid}(t + \Delta t) = \ddot{x}_{mid}(t)(\Delta t)^2 - (1 - f)x_{mid}(t - \Delta t) + (2 - f)x_{mid}(t) \quad [10]$$

4 SKIN DEFORMATION

We obtain v_{joint} (the position of a vertex after smooth skinning to joints) using the standard smooth skinning algorithm. Each skin vertex is influenced by a maximum of 3 joints.

Next we make an association between the skin vertices and the muscle vertices. Each skin vertex in bind-pose is associated with the nearest N_m muscle vertices. The muscle-vertex-to-skin-vertex weight is obtained by taking the inverse of the separation distance so that nearer points get more weight. After these weights are normalized, the user can assign weights for the association between skin vertices and entire muscles. This explicitly allows users to control the amount of influence from underlying muscles on various parts of the skin mesh.

Each of the skin vertices are affected by the muscle vertices via smooth skinning too:

$$v_{muscle} = \sum W_i' w_i' M_i' (B_i')^{-1} v_0 \qquad [11]$$

where v_{muscle} is the final smooth-skinned point in world space after muscle deformation, W_i' is the weight of the i-th muscle that is affecting the skin vertex, w_i' is the weight of the i-th muscle vertex that is affecting the skin vertex and v_0 is the bind pose skin vertex position in world space. M_i' is the current world matrix and B_i' is the bind pose world matrix for each associated muscle vertex for the skin vertex. These matrices are derived from the local coordinate frames described in Sec. 3.2.

The final vertex position is calculated by blending the two positions calculated above. Note that the total weight from the joints and muscles for each skin vertex adds up to 1.

$$v_{final} = v_{joint} + v_{muscle} \qquad [12]$$

5 RESULTS

A well-defined primary character might contain around 70-80 muscles while a secondary character might contain around 30-40 muscles by simplifying muscle groups. We assign 72 vertices per muscle to achieve reasonable binding results without flickers or sharp edges.

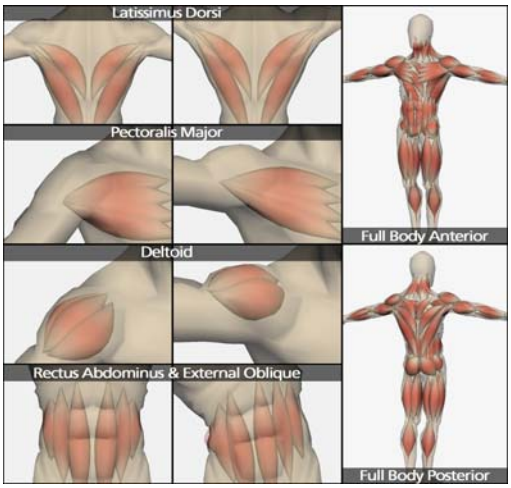

Figure 2: Approximation of different muscle shapes with only fusiform muscles.

5.1 Muscle Dynamics Test

We tested out how well a maximum of 200 muscles (about 2.5 main characters) runs on an average system (2.6GHz, 2.0GB RAM, GeForce 7800GS). These muscles are subjected to constant movement for 20 seconds, and the average FPS is recorded. As we can see, our system is efficient in delivering real-time dynamics-based movements, for 2-5 characters.

Muscles	**20**	**40**	**60**	**80**	**100**	**120**	**140**	**160**	**180**	**200**
Avg FPS	267	157	111	85	70	59	50	44	39	35

Table 1: Average muscle deformation and dynamics performance without skinning

5.2 Skinning Test

We then put our skin deformation algorithm to test by using a typical game character setup. This character is skinned to 21 joints (with each skin vertex affected by a maximum of 3 joints) and 0-80 muscles (with each skin vertex affected by a maximum of 4 muscle vertices). We varied the number of skin vertices from 5,000 to 20,000 vertices and subjected the character to constant motion for 20 seconds. The average FPS is recorded, as follows:

No. of Muscles	5000 vertices	10000 vertices	15000 vertices	20000 vertices
0	146	105	68	58
20	81	48	32	27
40	67	41	28	24
60	59	36	26	22
80	55	33	24	20

Table 2: Average muscle simulation and skin deformation FPS for varying skin resolutions

6 CONCLUSION

In summary, we believe that we have addressed several major limitations that have restricted use of anatomy based skin deformation in real time applications. This is due to a combination of simplified representation and intuitive design workflow. We have demonstrated our system with a full-body game resolution character that performs vigorous motions with pleasing deformations at 60-80 fps on an average PC.

7 REFERENCES

[1] A. Aubel, R. Boulic and D. Thalmann (2000), "Realtime display of virtual humans: Levels of details and impostors", *Circuits and Systems for Video Technology, IEEE Transactions on*, 207–217.

[2] J. E. Chadwick, D. R. Haumann and R. E. Parent (1989), "Layered construction for deformable animated characters", *SIGGRAPH 1989*, 243–252.

[3] L. Moccozet, and N. M. Thalmann (1997), "Dirichlet free-form deformations and their application to hand simulation", *Computer Animation*, 93.

[4] L. P. Nedel and D. Thalmann (1998), "Real time muscle deformations using mass-spring systems", *Computer Graphics International,* 156–165.

[5] F. Scheepers, R. E. Parent, W. E. Carlson and S. F. May (1997), "Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes", *Computer Graphics Forum*, 349–358.

[6] X. Wu, M. S. Downes, T. Goktekin and F. Tendick (2001), "Layered construction for deformable animated characters", *SIGGRAPH 1989*, 243–252.

[7] J. Wilhelms and A. V. Gelder (1997), "Anatomically based modeling", *SIGGRAPH 1997*, 173–180.

[8] L. Zuo, J. T. Li and Z. Q. Wang (2003), "Anatomical human musculature modeling for real-time deformation", *WSCG 2003*.