AUTOMATIC CREATION OF REAL-TIME MUSCLE SYSTEMS

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

By

Brent Mark Watkins, B.S.

* * * * *

The Ohio State University

2004

Master's Examination Committee:

Richard Parent, Adviser

Matthew Lewis

Approved by

Adviser

Department of Computer

and Information Science

ABSTRACT

Animating a living creature is one of the greatest challenges in computer graphics. One of the many details that need to be accounted for in order to achieve convincing results is the interaction of the skin with the subdermal structures such as the muscles, fatty tissue, and bones. The challenge is even greater when the body is in motion. This work presents a new approach for animating skin deformation that utilizes a simulation of muscles and computes their effects on the overlying skin. The muscle system is able to automatically generate muscles to match existing surface geometry in locations specified by the user. The deformation algorithm is designed in such a way as to handle first-order discontinuous surfaces. Additionally, the deformation algorithm is able to run in real-time while directly manipulating thousands of control points in order to improve usability by artists.

By specifying two curves on the skin geometry, the user determines the placement of a muscle that is automatically generated to fit the topology of the skin. A contraction model for these muscles is presented that mimics the effects of isotonic contraction. The contraction model is also able to simulate the effects of structures which influence the shape of a muscle during contraction, such as underlying bones. The resulting influence on the skin is generated by adjusting the location of skin control points using information derived from local muscle movement. Multiple falloff

functions are provided in order for the user to define a muscle's effect on the skin as the distance from the muscle increases.

ACKNOWLEDGMENTS

I would like to thank the Burpee Museum of Natural History and Brave New Pictures for allowing ACCAD to take part in the production of their television documentary, and ACCAD and Professor Maria Palazzi for allowing me to participate. Not only was the project the impetus for this research, it also provided me with valuable experience in a production environment.

I would like to thank Mike Altman, Ryan English, and Brent Zorich for creating the dinosaur models that appear in this document, and Jenny Macy for creating their textures. Ryan deserves additional recognition for pulling double duty as the rigger for the dinosaurs. His feedback, some of which was given while he was in the employ of Rhythm & Hues Studios as a professional rigger, was invaluable.

Many thanks to my master's examination committee, my adviser Dr. Richard Parent and Dr. Matthew Lewis, for their help with the thesis writing process. In addition, I would like to thank Dr. Parent for encouraging me to participate in the dinosaur production in the first place.

The human head model seen in this document was generated using FaceGen Modeler by Singular Inversions.

Maya®was provided to the Computer Science & Engineering department as a gift from Alias®.

VITA

April 15, 1976	Born - Spangdahlem AFB, West Germany
1998	B.S. Computer Science & Engineering, The Ohio State University
1999-2002	Member of Technical Staff I, Lucent Technologies, Columbus, OH
2002-2004	Graduate Teaching Associate, The Ohio State University
2004-present	Graduate Research Associate, The Ohio State University

FIELDS OF STUDY

Major Field: Computer and Information Science

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

One of the greatest challenges in computer animation is the simulation of living creatures. Living creatures are highly complex, consisting of complex internal structures surrounded by a flexible outer skin. The difficulty of creating a synthetic creature is further compounded by our familiarity with them - we have spent our entire lives observing the movements of humans. As experienced authorities, we are easily able to notice flaws in a simulation.

As complex as the internal structure of a creature is, it is only the surface that is visible. However, approaches that only model the skin and do not consider any other anatomical structures have not met with much success when it comes to creating a convincing artificial being. Models that take into account the underlying anatomy that influences the shape of the skin have been able to produce much more convincing results. This comes as no surprise - it is often the case that greater realism is achieved by carefully modeling the real world rather than using ad-hoc methods.

Other common approaches involve the use of free-form deformations or morph targets[1]. Free-form deformations have considerably flexibility, and therefore are applicable to a variety of situations. However, they also require a considerable amount of time to set up, especially when trying to achieve photorealistic results.

Morph targeting refers to the process of manually generating a number of poses for a character by adjusting the control points. Animation is then done by interpolating the control point positions between the poses. This process allows the desired appearances to be specified exactly, but is extremely time-consuming to set up. Even with this costly setup time, however, morph targets are often used because they allow complete control over the animation. For example, the facial animations for the character of Gollum in the *Lord of the Rings* films were performed by using 675 morph targets [6].

This document presents a new approach to simulating skin deformation. In order to achieve greater realism, the approach follows the model of simulating the underlying anatomy. Specifically, muscles are simulated. These muscles undergo isotonic contraction with adjustable bulging characteristics, and the skin is affected accordingly.

There have been a number of recent anatomically-based techniques that approach the problem by creating detailed anatomical models, including bones, muscles, tendons, and fatty tissue. In these approaches, creatures are created from the bottom up and the appearance of the skin is produced as an output based on the state of the anatomical model. While these techniques are able to provide realistic results, the actual appearance of the skin can be difficult to predict. Such unpredictability

[1]Morph targets are referred to in Maya as blend shapes

is a problem when a top-down approach is required - that is, when the outer surface specified first.

One example of a situation in which a top-down approach is needed is in the creation of "hero" characters. A hero character is one of the main characters in a piece of animation, and the appearance of the character is carefully designed in advance and therefore must look a certain way. Attempting to match a desired outer appearance when building a character with a bottom-up approach would be an exceedingly difficult task.

Another case in which a top-down approach is used is when surface data is obtained from the real world. This data could be obtained in a number of ways, including laser range finders and video-based techniques. Again, in this situation the outer surface is created first, making a bottom-up approach impossible.

These situations call for a top-down approach to modeling realistic skin. The system presented in this document allows for the creation of muscles given existing surface geometry, so that the surface may be specified first and the musculature may be added later. The modeler is allowed maximum artistic freedom and can design the surface without being required to worry about what sort of underlying anatomy may be required to produce such a surface.

A common challenge with physical models, regardless of whether they are included in a top-down or a bottom-up workflow, is that they are often computationally intensive. As a result, tuning the model to achieve the desired results can become a tedious task for an animator. After adjusting the parameters of the model, a lengthy delay may be incurred by computation before the results can be viewed.

This problem is amplified by the increasing power of 3D hardware available in workstations. Increasing power allows ever more complex models with greater numbers of control points to be designed and manipulated at interactive speeds. Unfortunately, physical simulations typically do not have linear computational complexity, which means that the time required to compute the physical model for a given piece of geometry grows faster than the complexity of the model itself.

The approach presented here trades off some physical accuracy in order to increase the speed of computation. The animator is therefore able to be more productive, because the results of altering parameters of the model may be immediately visualized.

The process of design is itself iterative by nature, meaning that the result is continually modified until a satisfying result is achieved. In this case, the satisfying result is realistic skin deformation. The interactive speed of the deformation algorithm results in immediate results for the designer, who is therefore able to work more efficiently while manipulating the system.

This research has evolved from the needs of a production that is currently in progress at the Advanced Computing Center for the Arts and Design, in conjunction with Brave New Pictures. A muscle system was desired in order to create more realistic skin deformations. The muscle system was intended to provide better results than the approach used to provide most of the other character deformations for the project, which utilized free-form deformations. The needs of the project resulted in the following design constraints:

- A top-down approach in which muscles are generated to fit existing models
- Real-time calculation of skin deformation based on muscle actions

- A skin deformation algorithm able handle first-order discontinuous surfaces

This document describes a system for simulating skin by modeling the underlying musculature that meets the above constraints. This system differs from other anatomically-based systems by emphasizing speed and top-down muscle creation. It differs from other skin animation approaches such as FFDs and morph targeting by producing visually pleasing results with much less setup time. Chapter 2 discusses previous approaches to the problem of representing skin. Chapter 3 describes the muscles themselves, including how they are created and how they behave when undergoing contraction. Chapter 4 describes how the contraction of the muscles influences the skin. Chapter 5 discusses the results of this research, and Chapter 6 presents conclusions and avenues for future research.

1.1 Artistic Anatomy

The motivation for this and other approaches that simulate skin deformation by modeling the underlying anatomy comes from the area of *Artistic Anatomy* [30]. Artistic anatomy is a specialization of general anatomy that is concerned with the structures of the body that influence surface form. This field has long been of use to artists in traditional media, and has become valuable to computer graphics as well.

The greatest influence on the shape of the surface is the skeleton. The general shape of the body is a direct result of the shape of the skeleton. Movement of the skeleton creates gross changes in the shape of the skin. However, fine details can also be produced by the skeleton itself. For example, the heads of the metacarpal bones can only be seen when the hand is clenched into a fist.

Located between the skeleton and the skin is the musculature. Besides being responsible for the motions of the skeleton, the muscles themselves have direct influence on the surface. Their shape can be seen on the surface as a series of convexities. Furthermore, the shape of a muscle changes as it contracts, resulting in a change in the shape of the surface.

A muscle consists of a bundle of muscle fibers which are connected, often with tendons, to the body in two locations. These locations are known as the *origin* and the *insertion*. The origin is the stationary end of the muscle and serves as its anchor. The insertion is the point of attachment to some structure that is moved when the muscle contracts. This is often a bone, but in some cases, such as the face, the muscle may insert directly into the epidermis.

When a muscle is contracted, it changes in shape. Two types of contraction are possible: *isotonic* and *isometric* contraction. In both cases, the contraction causes the muscle to change in thickness. The result of this can be seen on the overlying skin. Furthermore, isotonic contraction results in a change of length. The result of a change in length is especially apparent in the face, where muscles insert directly into the epidermis. When these muscles shorten, they pull the skin near the muscle insertion point toward the muscle origin.

CHAPTER 2

PREVIOUS WORK

Animating convincing skin deformation has been an active area of research for the past 30 years. Parke presented the seminal work in human facial animation [17] in 1972. His approach modeled the face as a polygonal mesh consisting of about 250 polygons. This model was developed by drawing a mesh on the face of an assistant, who was then photographed in a variety of poses. Each of these poses was used to define virtual pose in the computer, and animation was done by blending between the shapes. This technique was among the earliest application of the morph target approach.

A variety of other strategies for animating the movement of skin have been utilized over the years, and will be discussed by category.

2.1 Surface Models

The term *surface model* is used to classify approaches that only use the skin surface itself when computing animation. For example, Platt and Badler [18] describe a system in which "muscles" are defined by identifying a muscle connection vertex and the direction of muscle action. Vertices are then moved based on their connectivity.

Waters proposed another surface approach [28]. This approach presented the concept of *muscle vectors*. Such a vector defined a direction of muscle contraction. The effect of the vector was scaled by a falloff function based both on distance from the muscle vector and deviation from the direction of the muscle vector.

Lewis et. al. introduced the concept of pose-space deformations [12]. This technique uses an internal skeleton whose joints have weighted effects on the skin control points. The Animator manually generates a small number of poses, and the weights are then calculated automatically.

Wang and Phillips [27] use a similar approach in which a character is animated through a "training exercise" by some external means. The resulting data is used to compute coefficients for a deformation equation. Unlike the Lewis approach, computing a new pose is not dependent on the number of training poses.

Wade [26] describes an approach with which a control skeleton can be automatically generated from a polygonal model. The skeleton is generated based on a a discrete medial surface, and each vertex of the model is bound to a particular bone in the control skeleton. Animation of the control skeleton therefore results in the animation of the polygonal model.

Bloomenthal [2] presents a system that automatically computes weights which determine an existing control skeleton's effect on a particular surface control point by convolving a filter through the medial axis/surface of the character.

Kry et. al. [10] describe a system in which predefined key poses are decomposed into principal components. The resulting "eigenbases" describe the influence of the joints in the control skeleton. Skeletal poses are expressed in terms of the eigenbases, and the "EigenSkin" is generated using vertex programming on graphics hardware.

Yoshizawa et. al. [32] present another approach for automatically generating a control skeleton. The skeleton is extracted using a Voronoi-based technique. This skeleton is then deformed using free-form deformations, and the character is adjusted to fit the new skeleton.

2.2 Free-Form Deformations

The concept of free-form deformations was proposed in 1986 by Thomas Sederberg [21]. These were used in 1988 to deform the skin by Komatsu [9].

In 1989, Chadwick et. al. [3] proposed a method for animating characters using FFDs. This approach used multiple, aligned FFDs along the limbs. The FFDs at the joints served to deform the skin at the joints, and the FFDs between the joints expand outward to simulate muscle bulging.

FFDs typically relied on rectangular lattices, which are not ideal for every situation. In 1996, MacCracken and Joy [13] proposed a method for performing free-form deformations with lattices of arbitrary topology. This technique was based upon subdivision surfaces and allowed to creation of more general control lattices for FFD-based approaches.

Singh and Fiume introduced the concept of wire deformers [22]. In this model, wires represent the shape of the skin, and deforming the wires results in a deformation of the surface.

2.3 Simulations

Terzoupolos and Waters describe a model [23] for modeling a human face that consists of three layers in a mass-spring system. Lee et. al. [11] slightly simplified

this approach using a skin layer, a muscle layer, and a bone layer. Vertices on the bone layer are fixed and attached with springs to vertices in the muscle layer, which in turn are attached with springs to the vertices in the skin layer. In addition, springs are embedded within the muscle and skin layers themselves. Muscle movements are simulated by only adjusting the vertices in the skin layer - the resulting forces propagate to the skin layer through the mass-spring system.

Thalmann and Thalmann [14] use finite element methods to model skin during contact with the environment and during flexion of joints.

Chen and Zeltzer [4] created animation using a detailed biomechanical simulation. This simulation included a very detailed muscle model that was contracted using a finite element model. The effects on the skin were also computed via a finite element model.

Turner and Thalmann present a system [25] in which an elastic surface is wrapped around a kinematic skeleton. As the skeleton moves, the skin is acted upon by spring and reaction forces.

Such techniques are also used for other purposes besides animation. Koch et al [8] describe a method simulating the face during surgery. This method also uses finite element techniques.

Thalmann et al [24] describe a method for shaping skin based on a simplified underlying anatomy. Muscles are represented as ellipsoids, and the skin is an implicit surface based on the muscles.

During the same SIGGRAPH conference, Scheepers et. al. [20] and Wilhelms and Van Gelder [29] presented character animation techniques based on a detailed anatomical model of subsurface elements that influence the skin. Scheepers et. al. represented muscles using either ellipsoids or bicubic patches which bulge under volume-preservation constraints. Wilhelms and Van Gelder utilize a deformable cylinder to represent muscles which also maintain volume while bulging. Both techniques build the skin implicitly as an output of the state of the anatomical simulation.

Nedel and Thalmann [16] present a similar approach that uses a skeleton layer, a muscle layer, and a skin layer. Muscles are represented as a polygonal mesh and are contracted by embedding a mass-spring system and applying forces. The skin is generated over the other two layers by casting rays outward from the skeleton into a cylindrical grid.

Kahler et. al. describe a an approach [7] that creates muscles based on existing polygonal skin. Each muscle is represented by a group of ellipsoids that bulge when the muscle is contracted. The model also incorporates a bone layer. The bone, muscles, and skin are all connected with springs and a spring-mass system is used to compute skin deformations.

2.4 Performance-Based Techniques

Performance techniques refer to animation methods that require a user to act out the desired animation. This animation is captured and then applied to a synthetic figure.

Allen et. al. [1] capture real-world poses with a combination of optical motion capture and laser rangefinders to gather data from the real world. Markers are placed

on the limbs of a character who is scanned with the laser rangefinder in a variety of poses. The optical motion capture markers allow the location of the skeleton to be determined, and the rangefinder provides a detailed surface model. Using this data, a parameterization is computed for the effect of the skeleton on the data. This parameterization allows new poses that were not captured to be interpolated.

Sand et. al. [19] use a similar approach, but use silhouettes from one or more cameras in place of a laser rangefinder. As a result, the capture process is much faster and the actor may move about the capture space freely, but the results are less precise.

Chai et. al [31] describe a technique for animation facial expression by acting out the desired motions in front of a camera. Features tracked in the input video are combined with knowledge embedded in a database of facial motion capture data in order to produce realistic facial poses.

2.5 Parameterization

There are two commonly used standards that parameterize facial skin motion: FACS [5] and MPEG-4 [15]. Both of these models define facial animation in terms of a number of parameters with specific meaning.

FACS, which was developed by psychologists Ekman and Friesen, breaks down facial movements into a series of *action units* (AUs). Each AU describes a particular facial movement from a high level, such as "inner brow raiser" or "lip tightener". In reality most AUs correspond to the movement of multiple facial muscles. Although not intended for animation, FACS is commonly used as an input technique into facial animation systems.

MPEG-4 [15] also provides a generic set of inputs. Each head used must be marked with a particular set of feature points called Facial Definition Parameters (FDPs). These points provide a correspondence between MPEG-4 heads. Facial Animation Parameters describe a particular facial movement in terms if FDP movements, and therefore may be applied to any face that as been specified with FDPs.

The MPEG-4 specification also defines BDPs (Body Definition Parameters) and BAPs (Body Animation Parameters). These are analogous to FDPs and FAPs, but are used to parameterize human bodies rather than faces.

CHAPTER 3

MUSCULATURE

This document presents a skin deformation algorithm that is based on the simulation of muscles. Therefore, the muscle model itself is crucial to the quality of the system. The material in this chapter presents the details of the muscle model. Both muscle generation and muscle contraction will be discussed.

The muscle model allows for the simulation of *linear muscles*. Such muscles are characterized by having two extremities known as the *origin* and the *insertion*. The origin is stationary and provides an anchor point for the muscle. The insertion connects to some part of the body that the muscle is intended to move. Typically this is a bone, but in some cases the muscle inserts directly into epidermal tissue. One place that this occurs is the face, where there are few movable bones but a large number of muscles that are able to generate a huge variety of facial configurations.

The human body also contains several significant *sphincter* muscles, most notably the *orbicularis oris* that surrounds the mouth. These muscles are rings and contract toward a central point. Sphincter muscles will be discussed briefly at the end of this chapter. The proposed implementation is a straightforward extension of the sphincter muscle described in [7].

When contracting, a muscle can undergo either *isotonic* or *isometric* contraction. In isotonic contraction, the muscle shortens while increasing in thickness. An example of this phenomenon is the bulging of the biceps muscle when the elbow is flexed. Isometric contraction occurs when a muscle bulges without changing length. In this case, no movement is produced at the muscle's insertion, but the skin is still affected by the bulging of the muscle beneath the surface.

The muscle model presented here is primarily concerned with isotonic contraction. Furthermore, the model is particularly suitable for facial muscles, because the deformation algorithm allows for both skin bulging and skin stretching. Stretching is typically seen on the face when muscles that insert directly into the epidermis are contracted.

Linear muscles come in multiple forms, from the broad sheets of the *pectoralis* in the chest to the narrow form of the *biceps*. In order to allow for a wide variety of shapes, muscles are represented using parametric surfaces. Such a representation allows for the specification of a smooth, first-order continuous surface using relatively few control points. Continuity is needed in the muscle model in order to achieve continuity in the skin deformation, as will be seen in Chapter 4. Additionally, a smooth surface provides a much better approximation of the shape of a real muscle than other approaches, such as using a single or multiple [7] ellipsoids.

A parametric surface is defined by a two-dimensional mesh of control points. Therefore, operations on the surface consist of modifying the mesh. All muscle actions will therefore be operations that manipulate the muscle's control mesh. Muscle generation will be performed by calculating the location of of all points in the control

mesh, and likewise muscle contraction and bulging will be performed by adjusting the control mesh.

Two different meshes will be discussed in this chapter. The first mesh, known as the *basis grid*, consists of a mesh of points that all lie at approximately the same depth below the surface. The basis grid represents the middle of the muscle in terms of depth. The actual control mesh of the muscle is generated above (closer to the surface than) the basis grid, using a thickness function that varies across the muscle surface.

The two dimensions of a parametric control mesh are referred to as the u and the v directions. By convention, the v direction will always extend in the direction from muscle origin to muscle insertion, and will therefore be the direction in which contraction is performed. Likewise, the u direction will be across the body of the muscle. Recall that while the u and v directions are orthogonal in parametric space, in world space they may follow curved paths, allowing for muscles with organic shapes.

The specific parametric surface that was chosen for the implementation of the system is the NURBS surface. The primary motivation for this choice is the pre-existing availability of a rich set of NURBS functionality in the Maya API, which was used for the implementation. Using the existing NURBS functionality saved the work of implementing a parametric surface, allowing effort to be spent on the muscle model itself.

3.1 Muscle Generation

One of the primary goals of the muscle generation procedure is to allow for the creation of muscles based on existing surface geometry. Basing new muscles on an

existing surface allows for the use of a top-down approach in character modeling. To this end, the first step in muscle generation is for the user to define where the new muscle should be located by outlining its location on the surface. From this information the basis grid is created. The points on the basis grid all lie at approximately the same user-defined depth beneath the surface, within the area delimited by the boundaries drawn by the user.

There is a one-to-one correspondence from the basis grid points to the control points of the muscle surface. Therefore, there are the same number of basis grid points as surface control points. Each surface control point is located at a calculated distance from its corresponding basis grid control point. This distance is defined by a quadratic thickness function.

τ_s	skin thickness
τ_m	muscle thickness
p	a control point on the user-drawn curve
$\vec{n}$	the surface normal at a control point p
$p_{i,j}$	a point on the basis grid
$q_{i,j}$	a control point on the muscle surface
$\vec{n}_{i,j}$	the interpolated normal at $p_{i,j}$
$x_{i,j}$	the distance $\|q_{i,j} - p_{i,j}\|$
i_{tot}	the number of control points in the u direction
j_{tot}	the number of control points in the v direction

Table 3.1: Variables used in muscle generation

From the point of view of the user, muscle generation consists of the following steps:

1. The user draws two curves on the surface of the geometry that define the extent of the muscle.

2. The user specifies τ_s, the thickness of the skin, τ_m, the thickness of the muscle, and optionally the degree of the parametric surface and the number of patches in the u and v directions.

3. The system generates a muscle which lies below the surface, conforms to the surface topology, and honors the extents defined by the user.

The location of the muscle is drawn by the user on the surface geometry by using two curves. These curves define boundaries for the muscle in one dimension. The boundaries for the muscle in the second dimension are defined by imaginary lines joining the endpoints of the curves. The boundaries in the third dimension are provided by the two thickness parameters, τ_s and τ_m. The relationship between the curves, the thickness parameters, and the muscle boundaries is shown in Figure 3.1.

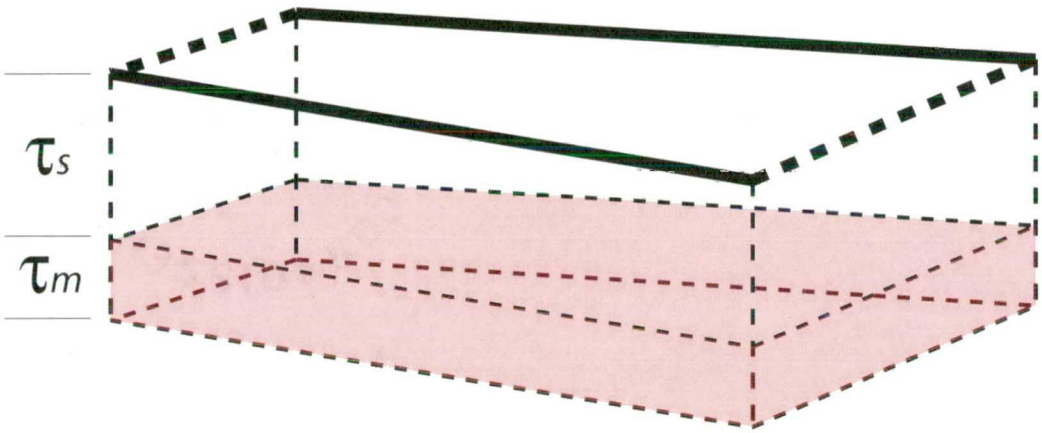

Figure 3.1: The thick solid lines represent curves drawn by the user. These curves, together with the thickness parameters τ_s and τ_m, determine the boundaries of the muscle, which is shown in pink.

The curves also determine which end of the muscle is the origin and which end is the insertion. Therefore, the curves determine the v direction of the parametric muscle surface. The imaginary lines that connect the endpoints of the curves lie in the u parametric direction. The muscle origin is defined to be at the end at which the curves begin. Likewise, the muscle insertion is defined to be at the end at which the curves end. The "beginning" of a curve is defined as the first control point, and the "end" of a curve is the last control point.

The parameters for degree and number of patches are optional. Increasing or decreasing these values allows the user to generate muscles that exhibit greater or lesser local detail as desired. For example, greater detail may be required to create an adequate muscle for a surface with high-frequency features. However, empirical observation has found that a degree of three is sufficient for most situations.

The inputs into the muscle generation algorithm are given in Table 3.2.

- the original surface
- the curves drawn by the user
- τ_s, the thickness of the skin
- τ_m, the thickness of the muscle
- the degree of the muscle surface
- the number of patches on the muscle surface in the u direction
- the number of patches on the muscle surface in the v direction

Table 3.2: Muscle generation inputs

3.1.1 Algorithm

The first step in muscle generation is to compute the basis grid. Because of the one-to-one correspondence of the basis grid to the actual control points of the muscle surface, the number of points required for the basis grid is determined by the requested degree and the number of patches. The relationship of the degree and number of patches to the number of points is:

$$\text{number of points} = \text{degree} + \text{number of patches} \tag{3.1}$$

The number of patches may be different in the u and v directions, so the basis grid may not be a square array.

The control points of the user-defined curves serve as the initial points for the basis grid. Each of these points must be moved to the proper depth. This will be done by moving them along the normal of the skin surface at the control point's location.

The curves are generated by using Maya's "draw on surface" capability. When this feature is utilized, the control points for the curves are placed directly on the surface. In addition, Maya provides functionality to lookup the normal at a given surface location. By leveraging this functionality, the surface normal can be obtained at each control point. Each control point is then displaced inward along its normal to a distance of $\tau_s + \tau_m$ beneath the surface.

$$p = p - \vec{n} \cdot (\tau_s + \tau_m) \tag{3.2}$$

Once they have been moved to the proper depth, the displaced control points are used to initialize the basis grid. Equation 3.1 determines how many points are needed along each (u or v) direction. The curves lie in the v direction, so the first step is

to generate the required number of points in the v direction by linear interpolation between the existing control points of each curve.

After this step, there are two "columns" of control points - one for each of the curves used to define the muscle location. Each of these two columns has the needed number of points in the v direction. The remaining $u - 2$ columns are generated by interpolating the corresponding vertices, resulting in u columns of v control points. This process is illustrated in Figure 3.2.

Figure 3.2: Interpolation of basis grid points from the original control points. In this example, 5 points are needed in the v direction and 4 points are needed in the u direction. From left to right: the original control points, interpolation within each column to generate 5 points, interpolating additional columns to generate 4 columns.

In order to place each basis grid point at approximately the same depth beneath the surface, the normals for the new points are also generated by interpolating between the normals at the displaced control points. This interpolation is performed linearly through quaternion space[2]. Once a normal $\vec{n}_{i,j}$ has been defined for each $p_{i,j}$, the locations of the points may be refined such that they are all at a distance of $\tau_s + \tau_m$ beneath the surface, measured in the direction of the normal.

[2]Maya provides this functionality. The interpolation is done "by rotating vector a into vector b about their mutually perpendicular axis by a given factor ($\in [0, 1]$)"

The complete basis grid has now been computed. The basis grid is composed of a sufficient number of points to define the parametric muscle surface, all of which lie at a distance of approximately $\tau_s + \tau_m$ beneath the surface.

The final control points for the muscle surface will be computed using the basis grid as an input. By offsetting each control point $q_{i,j}$ from its corresponding basis grid point $p_{i,j}$ by a different amount, a curved muscle surface will be generated.

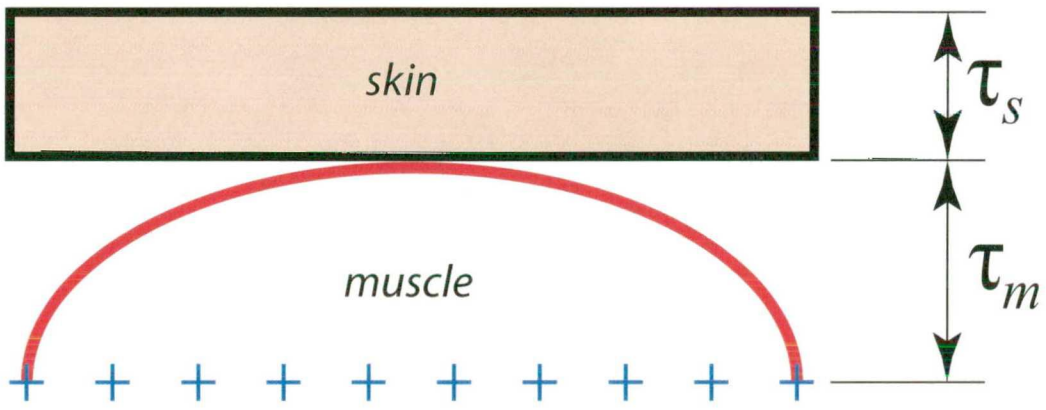

Figure 3.3: The basis grid (blue crosses) is created at depth $\tau_s + \tau_m$ below the surface. The muscle has maximum thickness τ_m.

The relationship between the basis grid, the muscle surface, the skin surface, and the thickness parameters τ_s and τ_m is demonstrated Figure 3.3. Notice that the muscle surface actually represents only half of a muscle, and τ_m gives the half-thickness rather than the full thickness. The reason for this is that the bottom half of the muscle is not actually needed by the surface deformation algorithm. It was therefore omitted in order to speed computations.

In order to generate each control point $q_{i,j}$, its displacement from the corresponding basis grid point $p_{i,j}$ must be computed. The displacement will be toward the surface, in the direction of $\vec{n}_{i,j}$, the normal at $p_{i,j}$. The distance of the displacement, $x_{i,j}$, is initialized to be τ_m.

If each control point $q_{i,j}$ were displaced from the corresponding $p_{i,j}$ by the same amount, the result would be a muscle of uniform thickness that conforms to the shape of the surface above it. However, in order to create a better muscle model the thickness should be allowed vary, such that the muscle is thickest in the center and thinnest at the edges.

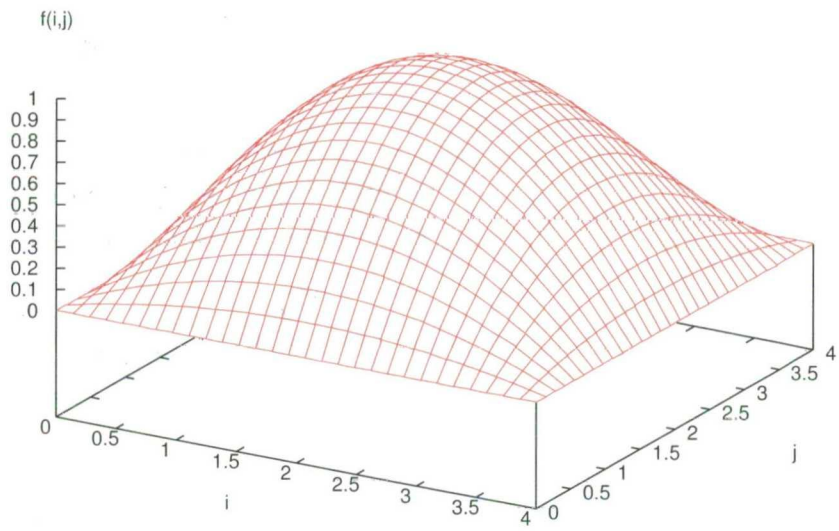

Figure 3.4: A plot of the function used to scale the thickness of the muscle. For this example, i_{tot} and j_{tot} are equal to 5.

In order to produce a muscle that reaches maximum thickness in the middle and minimum thickness at the extremities, the distance $x_{i,j}$ is scaled by a simple quadratic function. This function takes i and j as input and is equal to 1 at the center of the basis grid and 0 at the edges. The function, $f(i,j)$, is based on the one used by [7], but has been extended to three dimensions.

$$f(i,j) = \left[1 - \left(\frac{2i}{i_{tot}-1} - 1\right)^2\right] \cdot \left[1 - \left(\frac{2j}{j_{tot}-1} - 1\right)^2\right] \tag{3.3}$$

i_{tot} and j_{tot} are the number of basis grid points in the u and v dimensions, respectively. i and j are 0-based, so $i \in [0, i_{tot}-1]$ and $j \in [0, j_{tot}-1]$. A plot of this function is shown in Figure 3.4.

After computing $f(i,j)$, it can be used to scale the final value of $x_{i,j}$.

$$x_{i,j} = \tau_m \cdot f(i,j) \tag{3.4}$$

Due to the properties of the quadratic thickness function (Equation 3.3, Figure 3.4), the value of $x_{i,j}$ will be the greatest in the middle and the least near the edges. Because $f(i,j) \in [0,1]$, the result of multiplying by τ_m is that $x_{i,j} \in [0, \tau_m]$. Therefore, the muscle reaches a maximum thickness of τ_m, the user-supplied muscle thickness.

Once $x_{i,j}$ has been calculated, all the information necessary to produce the muscle surface control points is available. Each control point is computed by displacing from the corresponding basis grid point a distance of $x_{i,j}$.

$$q_{i,j} = p_{i,j} + \vec{n}_{i,j} \cdot x_{i,j} \tag{3.5}$$

This model allows for muscles that are thicker in the middle than toward the extremities. Some muscles, especially sheet muscles such as the pectoralis, are in reality broad sheets of mostly uniform thickness. Such muscles may still be created

using this model, by setting τ_m to be very small or even zero. It should be noted, however, that a side effect of setting τ_m to zero is that there will be no bulging when the muscle undergoes contraction. The reason for this will become clear in Section 3.3.

Once the control points for the muscle surface have been generated, the basis grid is not discarded. In fact, the basis grid is key to both the contraction and bulging algorithms. Neither operation directly manipulates the muscle control points. Contraction is accomplished by modifying the basis grid points, and bulging is accomplished by modifying $x_{i,j}$, the distance from a basis grid to its corresponding muscle control point.

The basis grid was generated by moving all of its points to be the same distance beneath the surface. By maintaining constant depth, the basis grid follows the contours of the surface. Furthermore, the shape of the muscle is determined by the shape of the basis grid. As a result, the shape of the muscle will be similar to the shape of the surface. Figure 3.5 demonstrates a basis grid generated from a random surface and the corresponding muscle.

In reality, the shape of the surface is determined by the shape of the underlying anatomy. With this algorithm, the shape of the underlying anatomy is determined by the shape of the surface. In both cases, the surface and the underlying anatomy have similar shapes. This approach is therefore able to mimic reality, while at the same time allowing for a top-down approach to character construction.

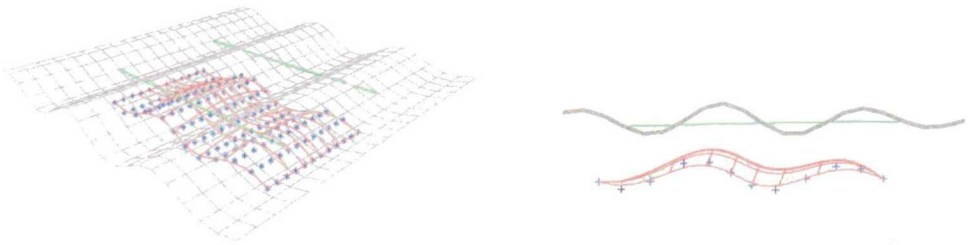

Figure 3.5: Perspective and side views of a muscle generated from a random surface. Both views include the larger skin surface above the smaller muscle surface. Note the blue cross-shaped locators corresponding to the basis grid and the shape of the resulting red muscle.

3.1.2 Shortcomings

The surface normals are used to project the curve control points below the skin surface. These projected control points serve to initialize the basis grid. Although it has not occurred in practice, if the curve control points are placed on discontinuous surface features, the resulting normals would result in the basis grid not being placed at the intended depth below the surface. This situation is demonstrated in Figure 3.6.

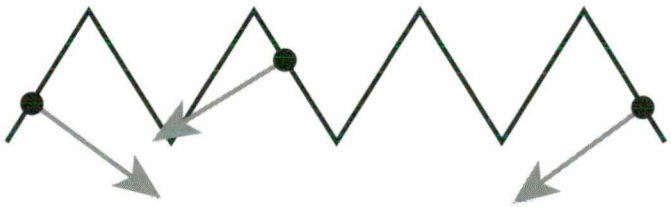

Figure 3.6: Jagged surface normals. The surface normals at the selected control points do not point in a direction that result in an accurate measurement of depth.

In this figure, the control points will be projected along the arrows, which represent the negated surface normals. The points will be located a distance of $\tau_s + \tau_m$ from their original positions, measured along the normal. However, the projected points will clearly be closer to the surface when measured in other directions. Specifically, directions that are more truly "outward" toward the surface of the skin.

A possible solution to this issue would be to not use the surface normals. Instead, consider the grid on the surface determined by the control points on the user-defined curves. At least four points are required (two each for two curves), which would result in a grid with a single cell. Additional control points would result in additional cells. The normal for each cell could be determined by treating as a 4-sided polygon. Then, the normal for each control point could be calculated by averaging the normals of each cell that is incident on the point. This process is similar to the way normals are calculated for polygonal meshes. The resulting normals are shown in Figure 3.7.

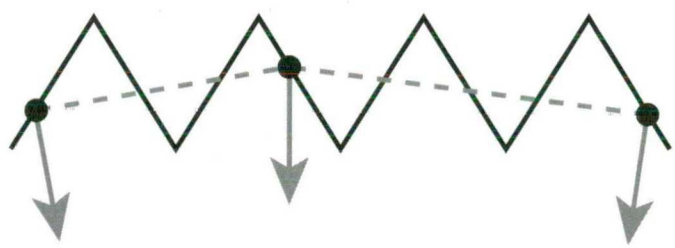

Figure 3.7: Jagged surface normals solution. The surface normals at the selected control points are based on the curves drawn by the user instead of the surface itself.

This approach results in a depth measurement that is truer to the overall shape of the surface, rather than a local feature.

3.2 Contraction

Muscle contraction is an important part of any muscle model. It is muscle contraction that is responsible for causing visible deformations of the skin. When a muscle contracts, it pulls on some piece of anatomy, typically bone or skin. This pulling can cause the skin to be stretched, especially when the muscle inserts directly into the skin itself. Contracting muscles also affect the skin by bulging. A bulging muscle pushes outward against the skin and causes a visible bulge on the surface.

$p_{i,j}$	a point on the basis grid
$p_{k,j}$	a column of points on the basis grid
c	contraction amount ($c \in [0, 1]$)
t_j	normalized distance of point $p_{k,j}$ along basis grid column k
$\tilde{t}_j$	t_j contracted using c
a_j	the index of the point $p_{k,j}$ that starts the line segment containing $\tilde{t}_j$
$p'_{i,j}$	a point on the basis grid after contraction

Table 3.3: Variables used in muscle contraction

This muscle model performs isotonic contraction. During isotonic contraction, a decrease in the length of the muscle causes a corresponding increase in the thickness of the muscle. The increase in thickness serves to maintain the volume of the muscle. Typical physical simulations [20],[29] incorporate volume preservation constraints that would create the exact amount of bulging necessary to compensate for the length lost during contraction. This model instead follows an artistic approach, where the amount of bulge is a parameter than may be adjusted by the artist. Bulging is discussed further in Section 3.3.

When undergoing contraction, the muscle shortens. This shortening occurs along the v direction of the parametric surface, as was established in Section 3.1. Therefore, an algorithm is needed that will move the control points of the muscle surface closer to the muscle origin with respect to the v direction.

Instead of moving the control points themselves, the contraction algorithm will instead move the points on the basis grid. Once the basis grid points have been adjusted, the muscle control points can then be recomputed using Equation 3.5.

Real muscles experience certain constraints while undergoing contraction. While there is freedom to expand outward because the skin is flexible, typically there is not freedom to move inward because of the rigid bone beneath them. Instead, muscles slide over the surface of bones as they decrease in length. Although bones are not explicitly modeled in this system, this property of contraction should still be accounted for in order to increase realism.

The contraction algorithm will ensure that the muscle does not penetrate any deeper into the body. This property of the algorithm serves to simulate the effect that underlying bone would have on a real muscle.

The means by which this constraint will be satisfied is to adjust the basis grid points such that their new positions lie along the line segments which connected the initial points (as created by the muscle generation algorithm). An example of this is illustrated in Figure 3.8.

3.2.1 Algorithm

The v direction on the muscle surface is the direction along which contraction occurs. Therefore, the points of the basis grid will be contracted in the v direction.

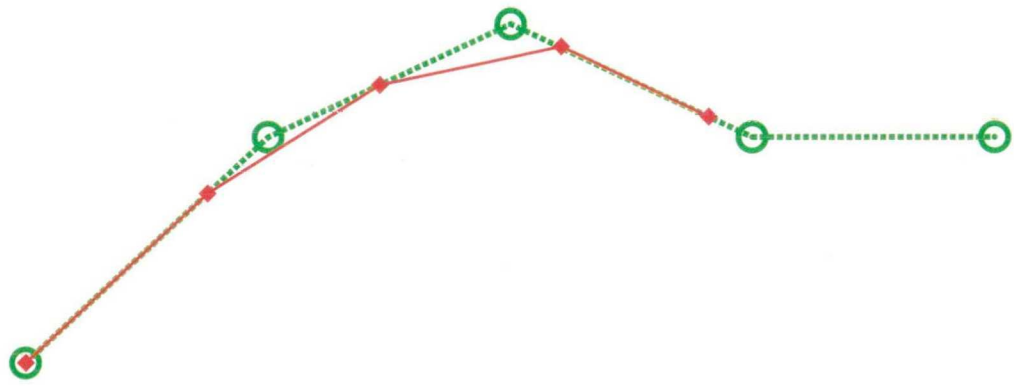

Figure 3.8: While undergoing contraction, the basis grid points do not penetrate any deeper than the line segments joining their initial positions. In this example, the open points represent the initial positions and the filled points represented the contracted positions. The points have been contracted by 25%.

Each "column" of grid points in the v direction is contracted separately, in the manner that is demonstrated in Figure 3.8.

Given a basis grid point $p_{i,j}$, i represents an index along the u direction and j represents an index along the v dimension. Therefore, define a column of basis grid points to be a collection of points with the same index along the u direction.

For each fixed u index k, there exists a column in the basis grid $p_{k,j} : j \in [0, j_{tot} - 1]$ that lies along the direction of contraction. The points in this column will be contracted in such a way that the length is decreased and the points do not penetrate any deeper than their initial positions. The algorithm for this is based on the one used in [7].

The amount of muscle contraction will be defined by the parameter $c \in [0, 1]$. A value of $c = 0$ signifies that the muscle has undergone no contraction (i.e. is completely relaxed), while a value of $c = 1$ signifies that he muscle is fully contracted. Here fully contracted means in the mathematical sense - the muscle has contracted so far that its length is approaching zero. Such behavior would not be possible in physical muscles, but setting an artificial upper bound on contraction could prevent an animator from achieving a desired look. Therefore the entire range of contraction is made available.

The inputs into the contraction algorithm are given in Table 3.4. The output of the contraction algorithm is a new column of basis grid points, $p'_{k,j}$.

c	The amount of muscle contraction
$p_{k,j}$	A column of points from the original basis grid

Table 3.4: Muscle contraction inputs

From this point forward, the fixed u index k will not be used in variable subscripts. The contraction algorithm works on a one-dimensional column of points, not the two-dimensional array that represents the entire grid. The k is dropped for brevity, except when referring to a basis grid point.

The contraction algorithm works by moving the points along the line segments joining their initial positions. The first step in determining their new positions is to compute the combined length of all the line segments; this is the total distance along the line segments from the first point to the last point. After determining the overall length, each control point $p_{k,j}$ is assigned a parameter $t_j \in [0, 1]$ which represents the normalized distance of that control point along entire path. The first point, $p_{k,0}$, will be at distance $t_0 = 0$. The last point, $p_{k,j_{tot}-1}$, will be at distance $t_{j_{tot}-1} = 1$.

In order to normalize t_j such that all values are between 0 and 1, the distance to each control point is divided by the total distance. This is shown in Equation 3.6.

$$t_j = \begin{cases} 0 & \text{if } j = 0 \\ \frac{\sum_{i=1}^{j} \|p_{k,j} - p_{k,j-1}\|}{\sum_{i=1}^{j_{tot}-1} \|p_{k,j} - p_{k,j-1}\|} & \text{else} \end{cases} \tag{3.6}$$

Figure 3.9 shows a simple running example of the contraction algorithm. In the example, each point is an equal distance from its preceding point, so the values of t_j are evenly spaced.

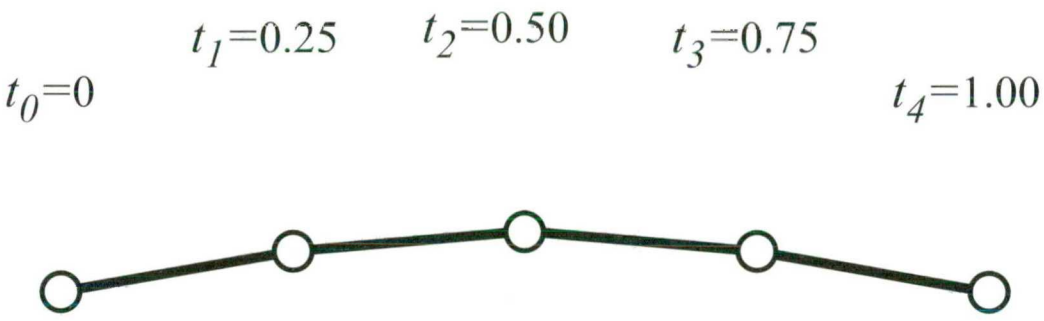

Figure 3.9: Contraction example 1

The contraction parameter, c, represents how much contraction the muscle has undergone as a fraction of the original length. Let ℓ_{orig} represent the original (uncontracted) length of the muscle, and ℓ_{new} represent the contracted length. The amount of contraction can be expressed as:

$$\ell_{orig} - \ell_{new} = \ell_{orig} \cdot c \tag{3.7}$$

Solving for ℓ_{new} gives:

$$\ell_{new} = \ell_{orig} - \ell_{orig} \cdot c \tag{3.8}$$

$$\ell_{new} = (1 - c) \cdot \ell_{orig} \tag{3.9}$$

Therefore, $1 - c$ represents the contracted length of the muscle, as a fraction of the original length.

$$(1 - c) = \frac{\ell_{new}}{\ell_{orig}} \tag{3.10}$$

In order to avoid contracting the muscle so far that it has zero length, the value of $1 - c$ is clamped to $[0.01, 1]$. It is then used to scale each t_j, effectively remapping the values of t_j from $[0, 1] \rightarrow [0, 1 - c]$.

$$\tilde{t_j} = \max\{0.01, (1 - c)\} \cdot t_j \tag{3.11}$$

Figure 3.10 again shows the running example, with the values of t_j and $\tilde{t_j}$ for each point. A value of $c = 0.25$ was used for the example.

t_j represents the uncontracted distance of point $p_{k,j}$ along the entire path. Therefore, multiplying this value by $1-c$, the fraction of the new length over the old length, gives a new set of values $\tilde{t_j}$, each of which represents the contracted distance of point $p'_{k,j}$ along the path.

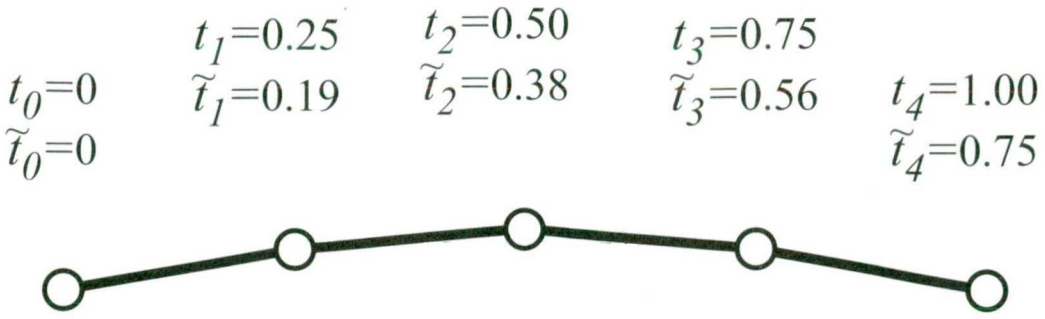

Figure 3.10: Contraction example 2 ($c = 0.25$)

Now that the distance of each $p'_{k,j}$ along the path is known, the location of each of these distances along the path must be found. Each new position $p'_{k,j}$ will be computed by linearly interpolating between some pair of points on the original path.

Each $\tilde{t}_j$ is mapped to an index $a_j \in \{0, \ldots, j_{tot} - 2\}$, which represents the index of the original point that immediately precedes the distance $\tilde{t}_j$. In other words, the distance $\tilde{t}$ lies on the line segment joining the points p_{k,a_j} and p_{k,a_j+1}.

$$a_j = \begin{cases} 0 & \text{if } j = 0 \\ m : t_m < \tilde{t}_j \leq t_{m+1} & \text{else} \end{cases} \tag{3.12}$$

Figure 3.11 again shows the running example, this time with the values of a_j included. Note that each a_j represents an index such that $\tilde{t}_j > t_{a_j}$ and $\tilde{t}_j \leq t_{a_j+1}$.

At this point, it is known which of the uncontracted points $p_{k,j}$ that each distance $\tilde{t}_j$ lies between. All that remains is to perform the linear interpolation.

$$p'_{k,j} = p_{k,a_j} + (p_{k,a_j+1} - p_{k,a_j}) \frac{\tilde{t}_j - t_{a_j}}{t_{a_j+1} - t_{a_j}} \tag{3.13}$$

Figure 3.12 shows the running example in the final configuration. The original path is grayed out and the original points $p_{k,j}$ are connected with dashed lines. The new points $p'_{k,j}$ are shown in red and connected with solid lines.

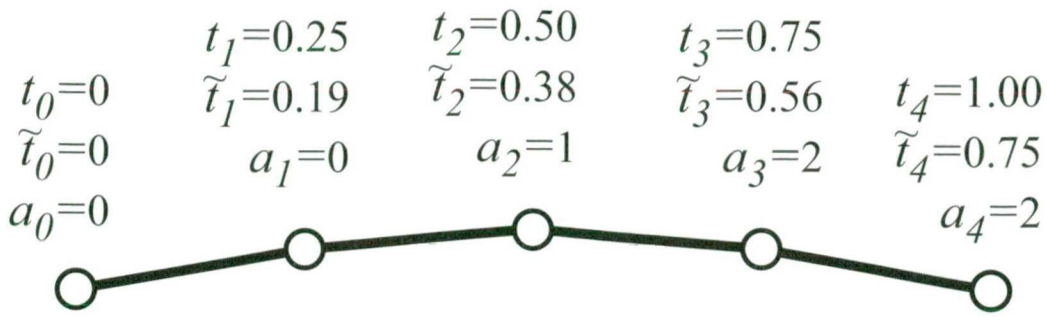

Figure 3.11: Contraction example 3 ($c = 0.25$)

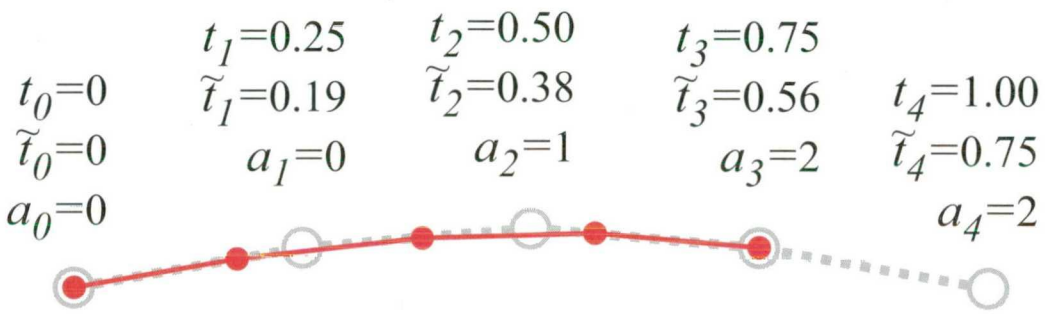

Figure 3.12: Contraction example 4 ($c = 0.25$). The gray path represents the original path, and the red path represents the contracted path.

After the contraction algorithm has been performed on each column from the basis grid, a new, contracted basis grid has been created. This new grid has been shortened toward the muscle origin by a factor of c. Furthermore, the points on the new basis grid do no penetrate any deeper into the body than the original basis grid.

The final step required in order to complete the muscle contraction is to calculate the new locations of the control points for the muscle surface. Using the adjusted

basis grid points $p'_{i,j}$, the muscle control points $q_{i,j}$ can again be found using the algorithm given in Section 3.1.

3.3 Bulge

The last remaining task of the muscle model is to simulate the muscle bulging that arises from isotonic contraction. The contraction model from Section 3.2 has already adjusted the basis grid points in order to shorten the muscle. Therefore, modeling bulge will simply require increasing the muscle's thickness.

As described in Section 3.1, the muscle thickness is determined by multiplying τ_m, the muscle thickness value provided to the muscle generation routine, by the quadratic thickness function (Equation 3.3). Modeling bulge merely requires increasing this thickness value and again recomputing the muscle control points.

Furthermore, the amount of bulge should increase as the muscle length decreases. More specifically, the muscle thickness should be increased by an amount that is inversely proportional to the length of the muscle. As was stated previously, the amount of bulge will not be constrained in order to preserve muscle volume but will instead be left in the control of the artist.

3.3.1 Algorithm

The value $x_{i,j}$ represents the original muscle thickness as generated by Equation 3.4. That is, the distance from the basis grid point $p_{i,j}$ to the corresponding muscle surface control point $q_{i,j}$. The bulge algorithm will scale this value by a function $g(c, b)$. c is the same contraction parameter given in Section 3.1. b is the bulge factor, which is initialized to 2 and is adjustable by the user. A value of $b = 0$ signifies that the muscle will undergo no bulge during contraction.

$p_{i,j}$	a point on the basis grid
$p'_{i,j}$	a point on the basis grid after contraction
$q_{i,j}$	a control point on the muscle surface
$x_{i,j}$	the distance $\|q_{i-j} - p'_{i,j}\|$
c	contraction amount ($c \in [0, 1]$)
b	bulge factor

Table 3.5: Variables used in muscle bulging

The bulging function $g(c, b)$ exhibits the following characteristics. First, when $c = 0$, the value of $g(0, b) = 1$. Therefore, when there has been no contraction the thickness values are scaled by 1, resulting in no change. Second, when $c = 1$ the value of $g(1, b) = b+1$. Therefore, under full contraction, each thickness value will be multiplied by $b+1$. $b+1$ was chosen in favor of simply b so that a value of $b = 0$ gives a result of 1, meaning no bulging, rather than 0, which would zero out the thickness that was otherwise calculated by Equation 3.4.

The formula for $g(c, b)$ which exhibits these characteristics is a simple one:

$$g(c, b) = 1 + b \cdot c \tag{3.14}$$

Using this result, the new thickness value after bulging will be:

$$x_{i,j} = \tau_m \cdot f(i, j) \cdot g(c, b) \tag{3.15}$$

This replaces the previous value calculated in Equation 3.4. As in section 3.2.1, the muscle surface control points again need to be recalculated from the corresponding basis grid points, using the new values of $x_{i,j}$.

Note that the value of the function g is based *only* on b and c, and is therefore constant across all grid points. The value of the thickness function $f(i, j)$ serves to

scale the bulge such that there is more bulge in the center of the muscle and decreasing bulge near the edges of the muscle.

Although the system is intended to simulate isotonic contraction, isometric contraction may also be approximated. Recall that during isometric contraction, the muscle increases in thickness without decreasing in length. This can be done by increasing the value of b without changing the value of c. By doing so, the value of $g(c, b)$ will increase, resulting in an increase in thickness. Note that before doing this, the muscle must be contract by at least some small amount. Otherwise, if $c = 0$, the value of $g(0, b)$ will be equal to 1 regardless of the value of b.

3.4 Summary

While undergoing contraction, the muscle decreases in length and bulges. This is accomplished by moving the basis grid points, modifying the thickness values, and then recomputing the muscle surface control points. This process is outlined in Figure 3.13

1. Compute the new basis grid points $p'_{i,j}$, using the algorithm given in Section 3.2.1
2. Compute the new thickness values $x_{i,j}$, using the algorithm given in Section 3.3.1
3. Using these new values, compute the new positions of the muscle surface control points using Equation 3.5
4. Move the muscle control points to the new locations

Figure 3.13: Muscle contraction summary

3.5 Sphincter Muscles

Unlike the linear muscles described previously, a sphincter muscle takes the form of a ring that is able to expand and contract like a rubber band. A proposal for implementing such muscles within the existing framework is given below.

Muscle generation would be performed in a similar way, except that the curves used to define the extents of the muscle would need to be closed in order to provide the ring shape. Once these closed curves were drawn on the surface, the basis grid could be projected beneath the skin in the same way. The resulting muscle surface would also be closed along one parametric dimension to produce the ring shape.

Rather than using the contraction algorithm discussed previously, all points would be contracted toward a defined center of contraction. This center point could either be automatically generated, perhaps as the centroid of all basis grid points, or user-definable. As the contraction factor c is increased, each basis grid point would be moved closer to the center of contraction. This concept is demonstrated in Figure 3.14.

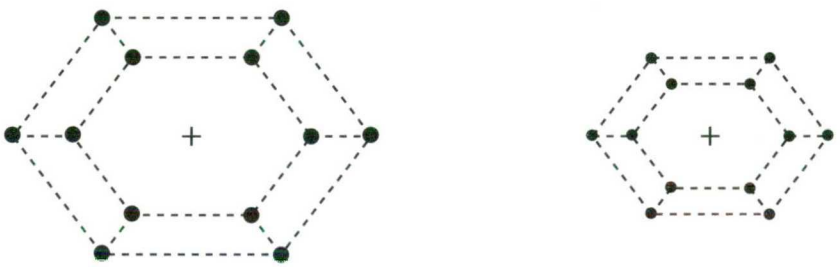

Figure 3.14: Sphincter muscle contraction. During contraction, all basis grid points are moved closer to the center of contraction, which is displayed as a cross.

CHAPTER 4

SURFACE DEFORMATION

Once muscles have been created for a character, they are used to deform the skin. The effects of real muscle bulging and contraction are seen in bulging and stretching of the skin, respectively. Simulating these behaviors is the goal of the surface deformation algorithm.

Due to the needs of the project, the surface deformation strategy was subject to the design constraints listed in Table 4.1.

1. It must be fast, so that users can interactively visualize the results of muscle contraction on the surface.
2. It must be able to handle challenging surface geometry, especially:
   (a) Polygonal meshes with vertices of high degree
   (b) Polygonal meshes with faces having more than four sides
   (c) Polygonal meshes that do not represent a smooth, first-order continuous surface
   (d) Non-polygonal geometry such as NURBS and subdivision surfaces

Table 4.1: Surface deformation design constraints

The first constraint, that the system be fast enough to produce results interactively, is a result of the nature of the project. The animators and lighters on the project are all students working with very limited time. Additionally, the computing resources available belong to an instructional institution and are not exclusively available for the project. Therefore, the computing time required to perform a more rigorous simulation is not available.

The second constraint, that the system must be able to handle challenging surface geometry, exists precisely because of the models used in the project. The main characters in the project are dinosaurs, and were modeling using polygons. The models have many locations in which the vertices have high degree, or the faces have more than four sides. Furthermore, the shape of the skin is not smooth and first-order continuous, as the skin on a human being typically is. Instead, the skin surface has many rough edges and spiky protrusions, due to the reptilian nature of dinosaurs.

The resulting algorithm is an attempt to produce skin movement that appears as realistic as possible, while still meeting the design constraints. As in any system, however, tradeoffs had to be made. In this case, both constraints demanded a tradeoff of some physical accuracy in order to be satisfied.

The first constraint, speed, requires that the strategy not be overly complex. A common approach to performing skin simulation is to use finite element analysis or a spring-mass system [11, 7]. Unfortunately, both of these techniques are expensive computationally. A spring-mass approach was attempted, and was able to reach near-interactive speed when used to manipulate skin with a relatively low number of skin vertices, but was ultimately rejected. One reason for this is that the actual character models in use have on the order of 10,000 vertices. Solving a spring-mass system is

not linear in computation complexity, and therefore does not scale up well to large problem sets. The other reason for rejecting the spring-mass approach is related to the second constraint.

The second constraint requires that challenging geometry is able to be handled by the algorithm. As mentioned above, the dinosaur models in use by the project have many vertices of high degree, faces with more than four sides, and first-order discontinuous surface features. Using a spring-mass system required embedding the system into the surface geometry, such that masses existed at the vertices and springs existed at the edges. Because of the nature of the geometry, this resulted in an unstable spring-mass system. One example of an unstable feature is given in Figure 4.1.

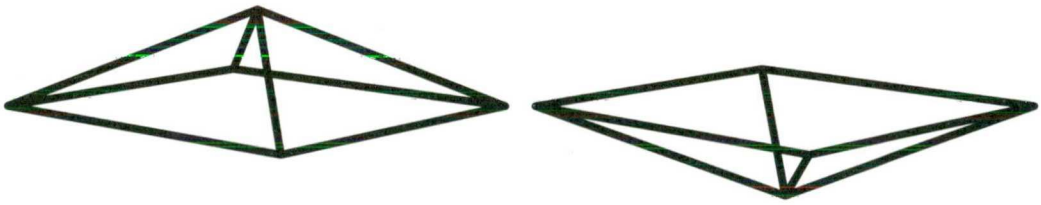

Figure 4.1: An example of a surface feature that results in an unstable spring-mass system. The initial configuration is shown on the left. However, both configurations are equally satisfying when solving the spring-mass system. Assuming that there is a spring in each edge, in both cases the corresponding springs have the same length.

In order to satisfy both design constraints, the resulting algorithm adjusts the control points of the surface geometry based only on the local movement of the influencing muscle(s). Each control point is deformed individually and does not depend on any connectivity information with its neighboring control points. Therefore, vertex

degree is irrelevant. Furthermore, the algorithm is able to work on skin geometry that is composed of polygonal, parametric, or subdivision surfaces.

The computational complexity of the surface deformation algorithm is linear in the number of control points being deformed. Therefore, the computation time scales relatively well with the complexity of the surface geometry.

4.1 Algorithm

The surface deformation process proceeds by iterating over every control point s_i of the surface geometry and performing the steps described in Figure 4.2. *Control point* is used here as a generic term that applies regardless of the surface type. For example, in the case of a polygonal mesh, a control point represents a vertex. In the case of a parametric surface, a control point represents a points in the control mesh.

1. Determine the location on the muscle surface that is closest to s_i
2. Determine the displacement of that location on the muscle from the rest position ($c = 0$) to the current position
3. Apply this displacement to s_i, scaled by a falloff function.

Figure 4.2: Outline of deformation algorithm

The surface control points are given the one dimensional index i because in the most general case control points do not have a multi-dimensional organization. Recall that the control points for the muscle surface have a two-dimensional index i, j (Table 3.1). This is because the muscle surface is a NURBS surface, whose control

points inherently possess two-dimensional structure. Because the surface deformation algorithm can also handle surfaces such as polygonal meshes, a two-dimensional organization cannot be assumed.

s_i	a skin point being deformed
u_i, v_i	uv-coordinates of the point on the relaxed muscle closest to s_i
p_i^0	the world space location of the point u_i, v_i on the relaxed muscle
p_i^1	the world space location of the point u_i, v_i on the contracted muscle
$\vec{\delta_i}$	the vector from p_i^0 to p_i^1
ψ_i	the falloff value applied to the deformation of s_i ($\psi_i \in [0, 1]$)
s_i'	the deformed location of s_i

Table 4.2: Variables used in surface deformation

The first step is to find the closest location on the muscle surface to the skin control point s_i. This step assumes that the closest location to s_i is likely to have the greatest influence on s_i. Note that this location is determined from the muscle's rest position.

The second step is to determine how the location found in step one has been affected by the muscle contraction and/or bulge. This can be represented as a simple vector that points from the rest position to the current position.

Finally, the displacement found in the second step is scaled by a falloff function and then applied to the control point. If the result of the falloff function is equal to 1, then the surface control point exactly mimics the movement of the muscle control point. The falloff function serves to decrease the effect of the muscle as distance from the muscle increases.

The speed advantage of this algorithm over finite element or mass-spring systems is clear. There is only a single loop, and none of the operations within the loop have any dependency on the number of control points on the skin surface. The most expensive operation is determining the contracted position of the closest point on the muscle surface, as will be discussed in Section 4.1.1.

The algorithm is also able to handle all of the types the challenging geometry listed in Table 4.1. The degree of the vertices is not relevant to the algorithm. Nor is the number of edges on a polygonal face. The continuity of the surface does not affect the algorithm. Finally, the algorithm is able to work on polygonal surfaces, NURBS surfaces, and subdivision surfaces. All of these properties are due to the fact that the deformation algorithm handles the control points entirely independently of each other.

The ability to handle these types of geometry allowed the deformation algorithm to successfully deform the dinosaur model shown in Figure 4.3.

The movement of each surface control point is linked directly to the movement of the muscle itself. Therefore, the quality of the surface deformation algorithm is directly affected by the quality of the muscle model. In order to obtain deformations that vary smoothly across the skin control points, it is necessary to use a muscle model that exhibits a smooth, first-order continuous surface. This is precisely the reason that a parametric surface is used to represent the muscles in the muscle model. Alternative muscle models that are composed of a combination of smaller elements, such as those used in [20] and [7], do not exhibit first-order continuity, and therefore would not be suitable for this deformation algorithm.

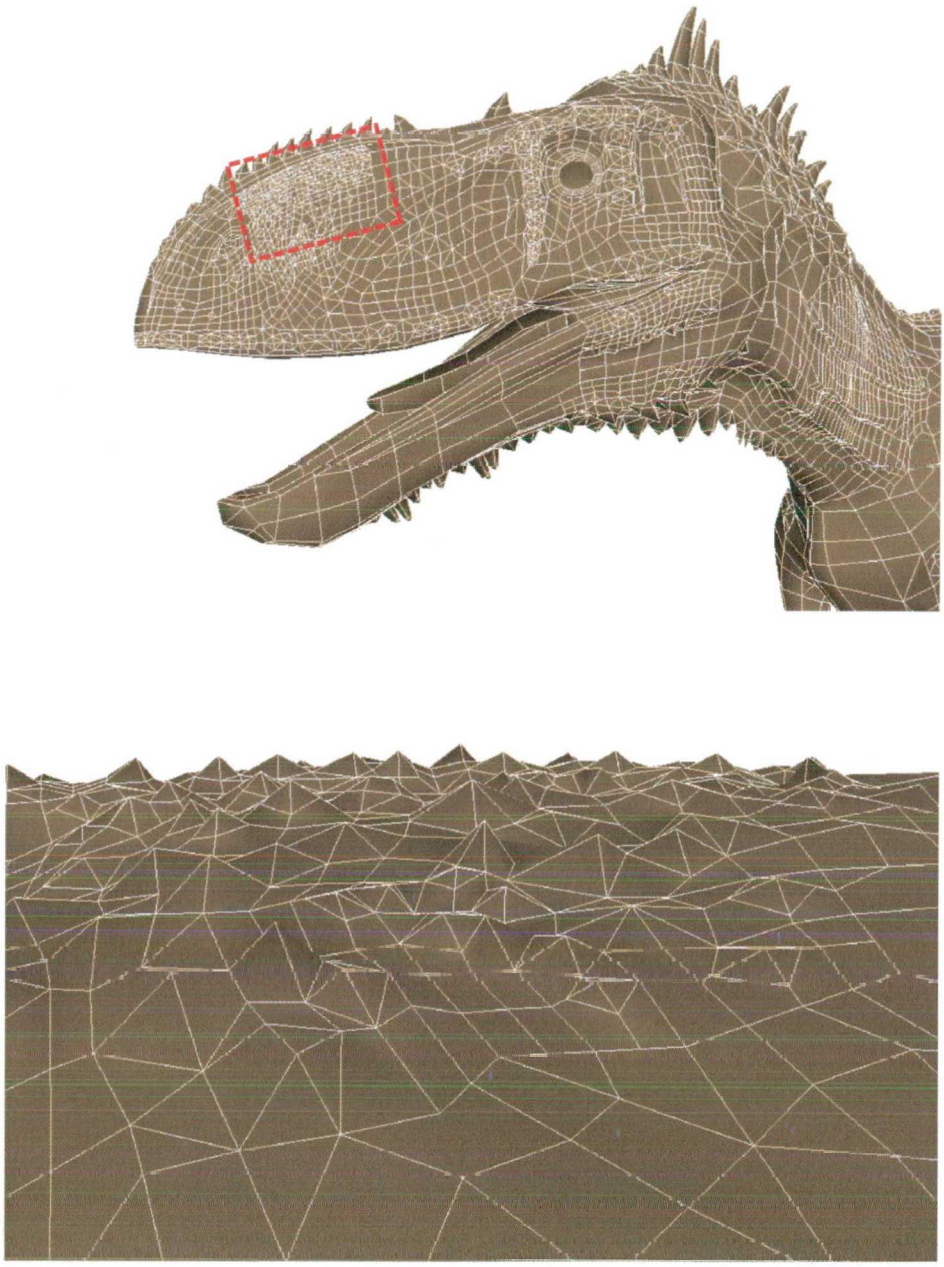

Figure 4.3: Challenging geometry that the system was called upon to deform. Above: close-up of dinosaur head. A particularly difficult section is outlined with a red box. Below: enlarged and rotated view of the difficult section.

4.1.1 Implementation Details

The first step in the skin deformation algorithm is for each influenced surface control point s_i to determine the closest point on the surface of the muscle when the muscle is at rest ($c = 0$). Because this calculation is expensive, these locations are precomputed.

After creating a muscle, it must be "attached" to the surface. At the time that attachment occurs, a loop is run that determines the closest point on the muscle surface to each skin control point. Each of these closest locations is then cached so that when the deformation algorithm is run, the value can simply be looked up rather than computed.

The movement of these locations when the muscle contracts needs to be found in the second step of the surface deformation algorithm. To facilitate this, the locations are represented in the (u, v) coordinate space of the muscle surface. The contraction and bulging algorithms reshape the muscle surface by moving its control points. Therefore, finding the new world-space coordinates of the location can be done by simply plugging the (u, v) coordinates into the parametric equation representing the muscle surface.

Pseudocode for the surface deformation algorithm is shown in figure 4.4.

The `lookupClosestPoint` function retrieves the cached location of the point on the muscle surface that is closest to s_i. The `lookupRelaxedMusclePoint` and `lookupCurrentMusclePoint` functions take as input a location in (u, v) coordinates, and output the world-space coordinates of that location on the muscle in the relaxed or current states, respectively. Once these two world-space positions are known, a simple subtraction gives the vector that describes the movement of the point on the

```
foreach undeformed surface vertex s_i
    u_i, v_i = lookupClosestPoint( i )
    p_i^0 = lookupRelaxedMusclePoint( u_i, v_i )
    p_i^1 = lookupCurrentMusclePoint( u_i, v_i )
    δ_i = p_i^1 − p_i^0
    ψ_i = calculatefalloff( ||s_i − p_i^0|| )
    s'_i = s_i + δ_i · ψ_i
```

Figure 4.4: Pseudocode for the surface deformation algorithm

muscle surface. This vector can be added to the undeformed coordinates of the surface point s_i, after being scaled by the falloff function.

4.1.2 Falloff

A real muscle affects the skin by pushing it upward from below as it bulges. Muscles that insert directly into the epidermal layer can also effect the skin by pulling it during contraction. In both situations, the elastic nature of skin causes the deformation to be spread across the surface, beyond the point of direct effect. As the distance from the muscle increases, the amount that the muscle affects the skin decreases.

This property is modeled in the skin deformation algorithm by the falloff function. The falloff function takes as input the distance of the skin surface point from the muscle, and produces as output a scaling factor in the range $[0, 1]$. This scaling factor is then multiplied by the displacement that is applied to the skin surface point.

The actual distance measured is the distance from the undeformed surface point, s_i, to the closest point on the undeformed ($c = 0$) muscle surface, p_i^0. The distance is measured using the undeformed configurations of the muscle and the skin so that the falloff factor remains constant and predictable for a given surface control point.

Because this technique is used in computer animation, it is important that the output of the deformation algorithm be deterministic. That is, for a given muscle and piece of skin geometry, setting the muscle contraction parameter c to a particular value should always result in exactly the same deformed configuration of the skin surface control points. In order to achieve this, the distance that is input into the falloff function must be predictable. For example, the "most recent" distance could not be used. This is because using the most recent distance could result in a distance for a given value of c that would depend on the prior value of c. The prior value of c could in fact be any value in the interval $[0, 1]$. To solve this problem, the measurement from the initial, undeformed configuration is always used.

The parameters of the falloff function are two distances. The first, d_{full}, is the distance inside of which the muscle has full effect. The second, d_{none}, is the distance outside of which the muscle has no effect. These two distance parameters are constant across the entire muscle surface.

The falloff function takes as input a distance d. This distance is the distance from the skin control point to the closest position on the surface of the undeformed muscle, and was previously calculated to be $\|s_i - p_i^0\|$ in Figure 4.4.

$$\psi = \begin{cases} 0 & \text{if } d \geq d_{none} \\ 1 & \text{if } d \leq d_{full} \\ \lambda\left(\frac{d - d_{none}}{d_{full} - d_{none}}\right) & \text{else} \end{cases} \tag{4.1}$$

When the input distance d is less than or equal to d_{full}, the muscle has full effect and the multiplier is one. When the distance is greater than or equal to d_{none}, the muscle has no effect and the multiplier is zero. If d falls between these extremes, its location between d_{full} and d_{none} is normalized to the range $[0, 1]$. The normalization is done so that the result may be passed into $\lambda : [0, 1] \rightarrow [0, 1]$. The function λ

determines the shape of the falloff. That is to say, it controls how the falloff value changes between the maximum and minimum values.

Multiple λ functions are provided that may be selected by the user. Examples of linear, quadratic, cubic, and cosine [28] falloff functions are given in Figure 4.5.

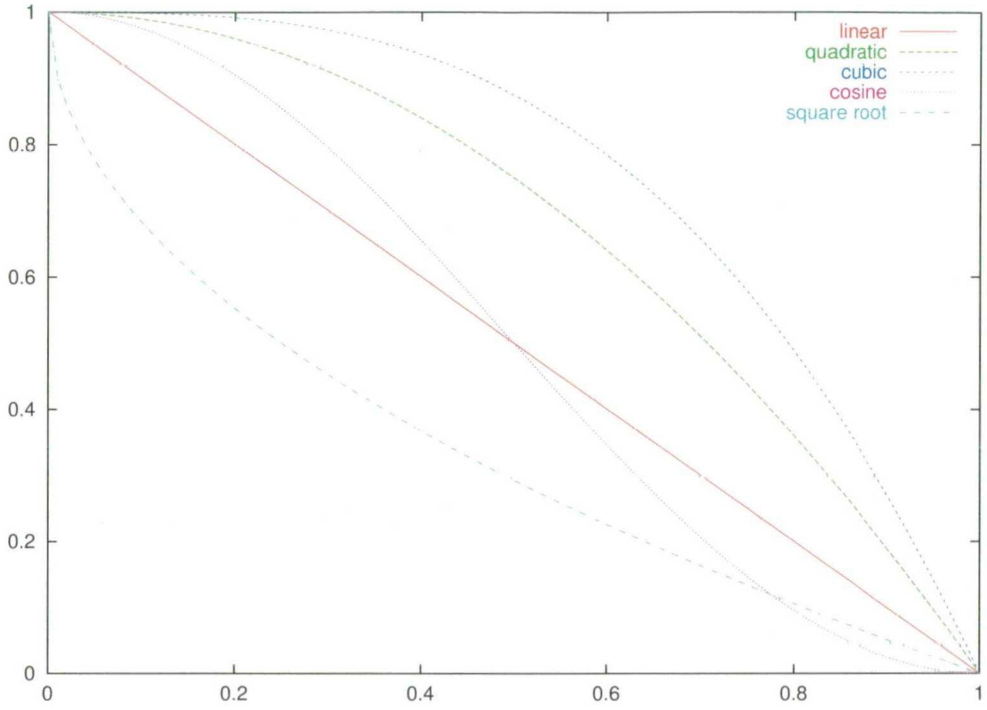

Figure 4.5: Falloff functions

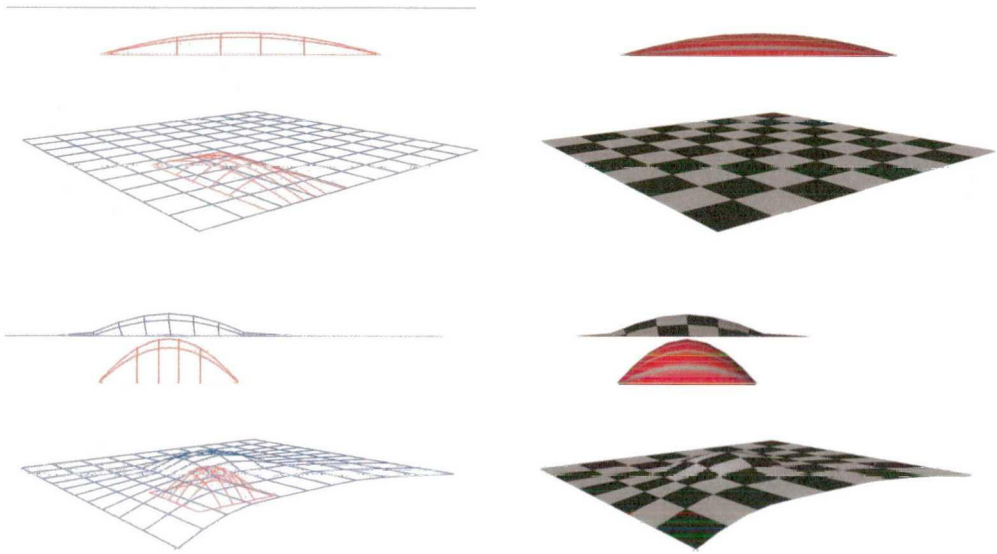

Figure 4.6: An example of a simple planar surface being influenced by a muscle. The muscle is shown in both a relaxed (first two rows) and a contracted (last two rows) state.

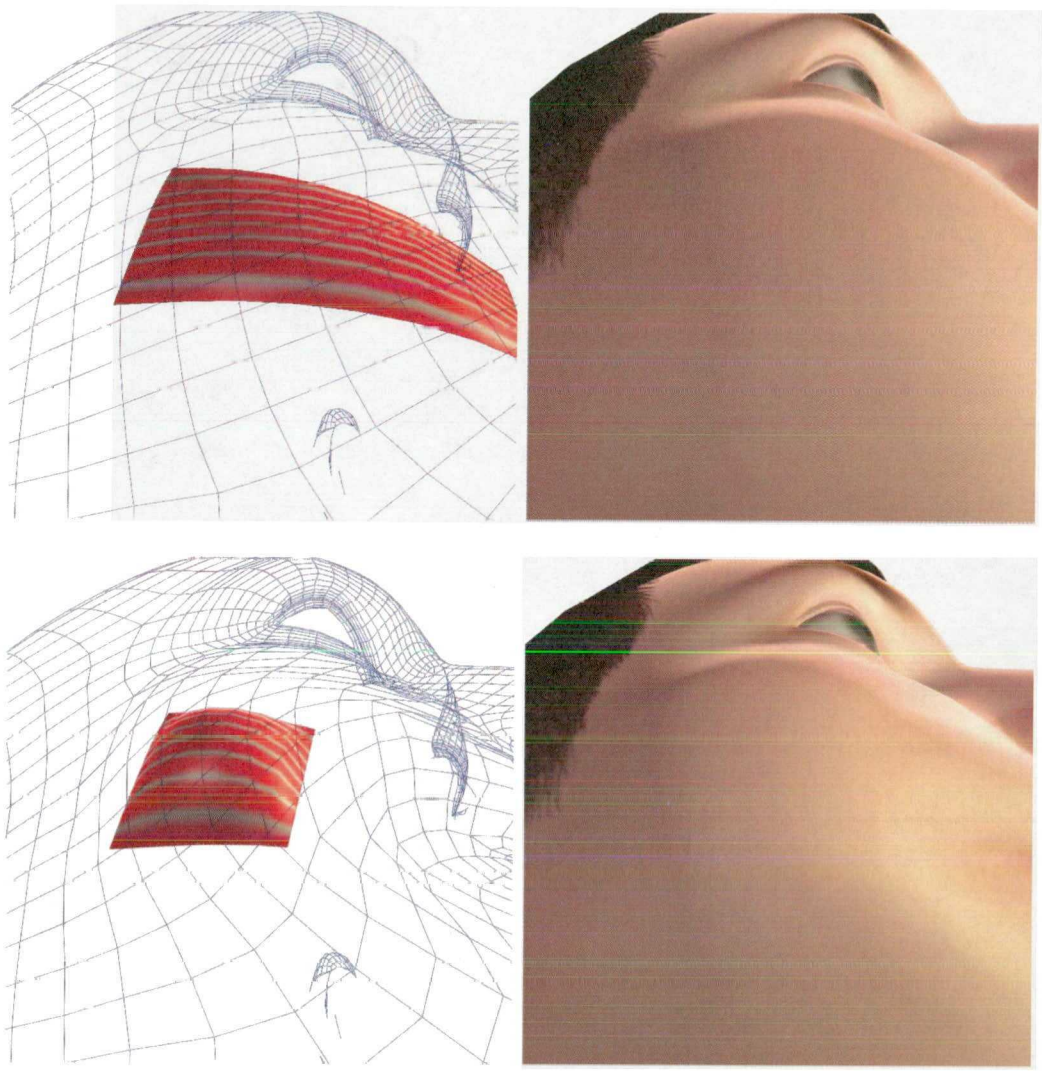

Figure 4.7: An example of a real model being deformed by a muscle. The muscle is both bulging and stretching the skin.

CHAPTER 5

RESULTS

The entire system has been implemented and tested in a production environment for a documentary concerning dinosaurs. The system was used to automatically create muscles for existing dinosaur models and to create skin deformation during the animation sequences.

The system has also been tested on a human head model. This model was created with FaceGen Modeler by Singular Inversions. Muscles were created in locations that would allow various facial expressions to be created, such as in the forehead, around the mouth, and in the cheeks. Some animation frames using this model are shown in Figure 5.4.

Various synthetic test environments were also used to test the muscle system, including flat planes (Figure 4.6), cylinders, and spheres.

The muscle generation process is quite simple from the point of view of the user. It merely involves sketching two curves on the surface and providing the desired thickness values for the muscles. After these steps have been taken, muscle generation only requires a single mouse click. If the placement of the muscle does not match the user's expectations, adjusting the curves and regenerating the muscle is a simple process requiring only a few minutes of work.

The simplicity of muscle generation allows this task to be performed very quickly. The muscles in the dinosaur characters were set up in approximately one hour per character. Each character was initially created with three pairs of muscles. Each pair consists of a muscle and its mirror across the sagittal plane of the body. The dinosaurs each have similar muscles in their jaws and around their nostrils. Other muscles were placed in various locations as needed, such as in the neck and on the back.

Most of the skin movements of the dinosaur models were set up using free-form deformations. This setup was done by a student with rigging[3] experience who participated in the production. Currently, he is employed as a professional character setup technical director at Rhythm & Hues Studios in Los Angeles, California. FFDs allow for great flexibility, but can be time-consuming to set up. For example, the nostrils on the dinosaur models were set up such that they could be controlled either by the muscle system *or* by FFDs. The initial FFD setup required a half hour per nostril. Thus, setting up the nostrils alone required an hour, during which all muscle-based setup could be done. When asked how long it would take to simulate the skin deformation created by the cheek muscles, the response from the rigger was "I am not sure that Maya could effectively and reliably create those results straight out of the box." Here the muscle system has even greater benefit, because the deformations it performs may not even be achievable with the techniques used for the rest of the character setup. At the very least, a significant time investment would be required in order to research another way of creating this motion.

[3]Rigging refers to the process of setting up a character model for animation. This is done by providing controls to do things such as move the leg (using forward or inverse kinematics) or blink the eyes

Therefore, the muscle system benefited the production in two major ways. First, it allowed skin deformations to be setup in far less time than required with FFDs. Second, it was able to produce pleasing muscle-based skin movement that could not be reproduced with other techniques.

Some of the dinosaur muscles were set up to be manually controlled by the animator, and some were set up to contract automatically based on the dinosaur's movement. For example, the muscles controlling the nostrils were set up for manual control. In contrast, the muscles in the cheeks were set up to contract based on the movement of the jaw. In this case, skin deformation was automatically produced by opening and closing the mouth, without requiring any additional work on the part of the animator. In fact, some of the animation had already been completed before the muscles were added. Although the muscles were added later, the end effect was still the same skin deformation, without requiring any extra work by the animator.

Figure 5.5 shows a series of images from a completed animation using the same model and muscle setup as Figure 5.1. The animator did not need to directly manipulate the contractions of the cheek muscles because they were set up to be automatically driven by the movement of the jaw. For the purposes of these images, all keyframes other than those for the jaw were removed and the camera was moved in closer to the head so that the results of the cheek muscles may be more carefully scrutinized.

The dinosaur animations are part of a documentary, and realism is a major factor. Therefore, some manual tweaking of the system's runtime parameters (see Table 5.1) was necessary. This iterative design process was made far more efficient by the real-time speed of the deformation algorithm. The result of modifying one of the parameters could be visualized immediately, allowing a more explorative approach to

be taken. Slower muscle simulation systems utilizing mass-spring or finite element models would have taken much more time to solve, increasing the time between iterations of the design.

c	contraction amount ($c \in [0, 1]$)
b	bulge factor
d_{full}	the distance inside of which the muscle has full effect on the skin
d_{none}	the distance outside of which the muscle has no effect on the skin
λ	falloff function

Table 5.1: System variables adjustable at run-time

The runtime parameters listed in Table 5.1 are those provided directly by this system. Because the system was implemented as a plugin for Maya, various other runtime parameters are available as a part of the Maya infrastructure.

The surface deformation algorithm is implemented as a Maya deformer[4]. Maya deformers provide an overall "envelope" parameter that globally scales the effect of the deformer. When finer control is desired, a weight may be applied individually to any control point that is affected by the deformer. This weight value is multiplied by the deformation and can scale it up or down. Assigning a weight of zero to a control point results in the muscle not affecting it at all.

Additionally, the set of points that the deformer affects may be edited. However, if the desired behavior is to prevent a muscle from affecting a particular control point, it is more efficient to remove it from the deformer's point set than to set the weight to zero. A weight of zero causes the deformation loop to be performed on the point,

[4]A "deformer" in Maya is a tool that modifies the shape of some piece of geometry by manipulating its control points

the output of which is multiplied by zero. All of the computation time spent by the deformation algorithm on that point is therefore wasted, because the result will always be zero. Removing the point from the deformer's set prevents it from being considered in the deformation loop. The computation time that was wasted when setting the weight to zero is therefore avoided. As a result, the time required to perform surface deformation is reduced.

This sort of set editing was done when setting up the muscles for the human head example (Figure 5.4). In order to keep the muscles in the upper lip from affecting the lower lip, the control points in the lower lip were manually removed from the upper lip muscles' deformation sets. Likewise, the control points in the upper lip were manually removed from the lower lip muscles' deformation sets. This manual tweaking was necessary because using only the distance parameters d_{full} and d_{none} was not sufficient. When at rest, the head model's mouth was closed. Therefore, the control points on the surfaces of the upper and lower lips were immediately adjacent to each other. In some cases, where the lips were touching, the upper lip control points and lower lip control points were in the exact same position. Therefore, there was no distance from the muscle that could include only the upper lip's control points and none of the lower lip's control points.

This additional and somewhat tedious task caused the human head model to require longer to set up than any of the dinosaur characters. Approximately two hours was spent creating the muscles, adjusting the deformation sets, and creating more natural controls for the muscles. The "more natural"' controls consisted of moving the parameters for muscle contraction together in a common location, grouping them, and giving them names such as "brow left" and "smile".

In all cases, empirical observation shows that the generated muscles do conform to the shape of the skin surface above them. This can be seen in Figures 3.5, 4.6, 5.1, and 5.4. Thus, the system is successfully able to create new muscles based on existing surface geometry, which satisfies the primary goal of the muscle generation algorithm. Satisfying this goal allows for the use of a top-down character design process.

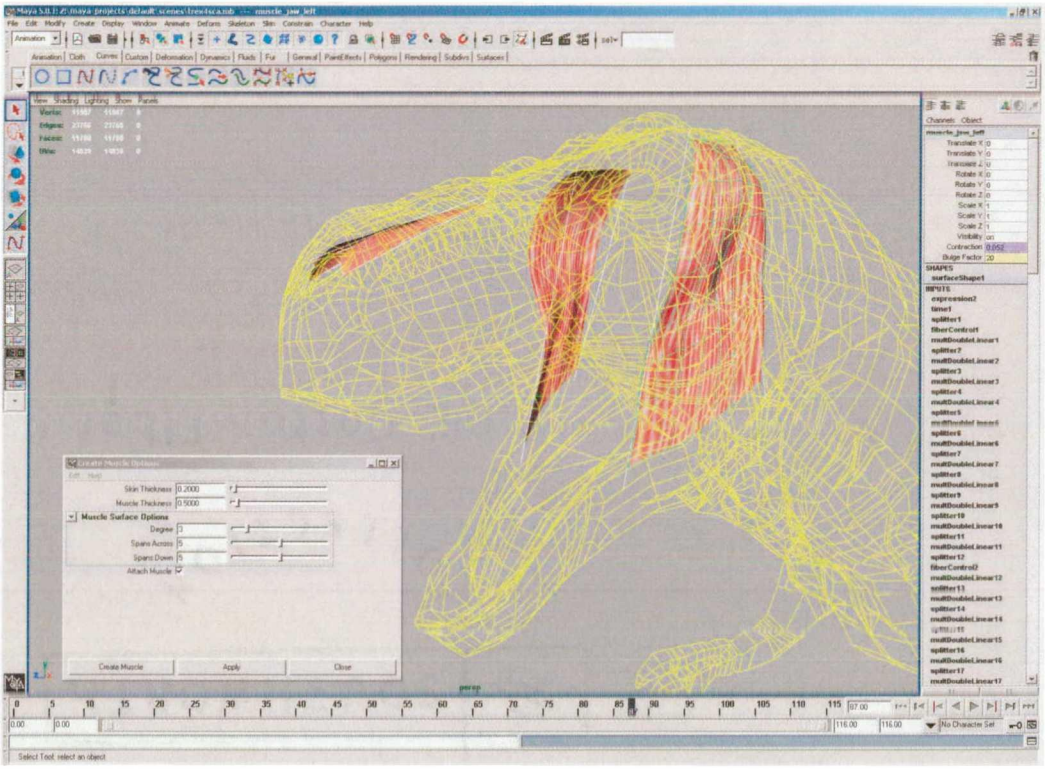

Figure 5.1: A snapshot of the Maya interface with the muscle creation window open.

Figure 5.1 shows the Maya interface with the muscle creation window open in the lower left-hand corner. The tyrannosaurus rex model in the image is made up of

about 12,000 polygons and 12,000 vertices. Even with its rough geometry, generating muscles that fit the surface and preserved the modeler's intent was a simple task, requiring only a few minutes to achieve the desired muscle shape and placement.

The results of the skin deformation algorithm have also been satisfying. The resulting skin deformations were approved for use in the documentary for which they were created. Although some accuracy has been traded off for speed, the results were nonetheless pleasing to the eye. Figures 5.4 and 5.5 show the results of the skin deformation on human and dinosaur models.

In the case of the dinosaur models, this skin deformation algorithm was able to succeed where other approaches did not. A system involving the use of springs, masses, and dampers was also attempted, but was never able to achieve pleasing results. In fact, the rough edges of the dinosaur models (see Figure 4.3) caused unpredictable and generally unsatisfactory results. In some cases, this was able to be resolved by using a very small time step size in the forward integration algorithm that solved the spring-mass system, but this resulted in unacceptably long computation time.

This system can be implemented into the animation pipeline in a variety of ways. Most obviously, muscles can be attached to characters and used to directly manipulate their skin. The system can also be combined with other techniques when setting up a character for animation.

Another way to utilize the system is in the creation of morph targets. The morph target approach calls for manually specifying various configurations for the character and interpolating between them. The drawback is that specifying each desired configuration requires considerable time. The system could be used to facilitate this

process. If the target configuration is intended to show the result of muscle influences, muscles could be attached to the target model and used to create the desired effect. If the physical accuracy of the muscle system is inadequate for the situation, it could instead be used to generate an initial approximation of muscle effects, which could then be fine-tuned. Such a use of the system would still provide a time benefit over creating the target by hand.

5.1 Performance

Performance was a design constraint of the skin deformation algorithm. Hands-on use of the system when connected to the existing dinosaur animation setup resulted in speed that "felt" interactive. In order to quantify the performance, some tests were run on the Tyrannosaurus Rex model shown in Figures 5.1 and 5.5.

In order to perform these tests, as many features as possible were disabled that were not directly involved in the muscle algorithm. This was done in order to isolate, as much as possible, the performance of the muscle system itself from any other computationally expensive elements in the model.

Although the full dinosaur model consists of approximately 12,000 control points, less than 1,000 were in the deformer sets of the two cheek muscles. In this case, removing points from the deformer sets was done to save processing time (by looping over several hundred vertices rather than 12,000) rather than to fine-tune the effects of the muscles, as was done for the human head model.

After setting up the test environment, a simple animation of opening and closing the jaw of the dinosaur played back at approximately 15fps on a 3GHz Pentium4.

The complete dinosaur, fully setup for animation, comprises a very complex 3D model. In order to obtain performance numbers that more accurately represented the muscle system in isolation, a synthetic test environment was created using a simple plane setup as in Figure 4.6. A simple animation loop was created that contracted the muscle from $c = 0$ to $c = 0.6$ and back over the course of 60 frames. For each trial, the only change was in the number of control vertices in the plane. d_{full} and d_{none} were specified such that all control vertices were in the area of influence of the muscle, and no control points were removed from the muscle's deformation set.

Figure 5.2 shows the results of the trial in terms of frames per second. This number was obtained by configuring Maya to playback the animation at the maximum framerate possible, and observing the fps value.

Observing this plot shows that the frames per second achieved is essentially inversely proportional to the number of control points being deformed. In other words, $r \approx a \cdot \frac{1}{n}$, where r is the rate in frames per second, n is the number of control points, and a is some constant.

In Chapter 4, the claim was made that the surface deformation algorithm runs in linear time with respect to the number of control points being deformed. This statement refers to the amount of time required to compute the deformation for all skin control points. Figure 5.2 shows the *rate* at which the deformation can be calculated, not the time required. However, converting between the rate (r) and the time for one iteration of the algorithm (t) is trivial.

$$t = \frac{1}{r} \tag{5.1}$$

This formula suggests that the amount of time required for the surface deformation algorithm is indeed linear with respect to the number of control points being deformed.

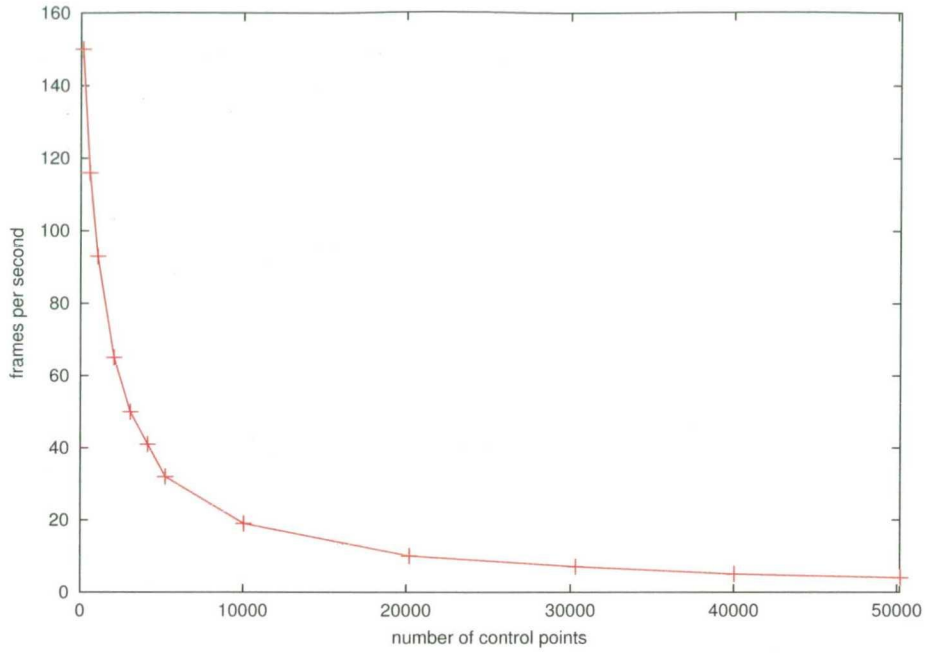

Figure 5.2: A plot of the performance of the surface deformation algorithm. The x-axis shows the number of control points being deformed, and the y-axis shows the number of frames per second computable by the algorithm.

If $r = \frac{1}{t} \approx a \cdot \frac{1}{n}$, then $n \approx a \cdot t$, a linear relationship. Figure 5.3 replots the data from Figure 5.2 using Equation 5.1 to provide the values along the y-axis. The resulting graph proves the linear relationship between the number of control points being deformed and the time taken by the surface deformation algorithm.

Synthetic benchmarking with the plane provides a good benchmark of the surface deformation algorithm running in isolation. This is useful for examining the raw performance of they system by itself. However, the numbers obtained with the dinosaur model serve as a good indicator of performance in an actual animation environment.

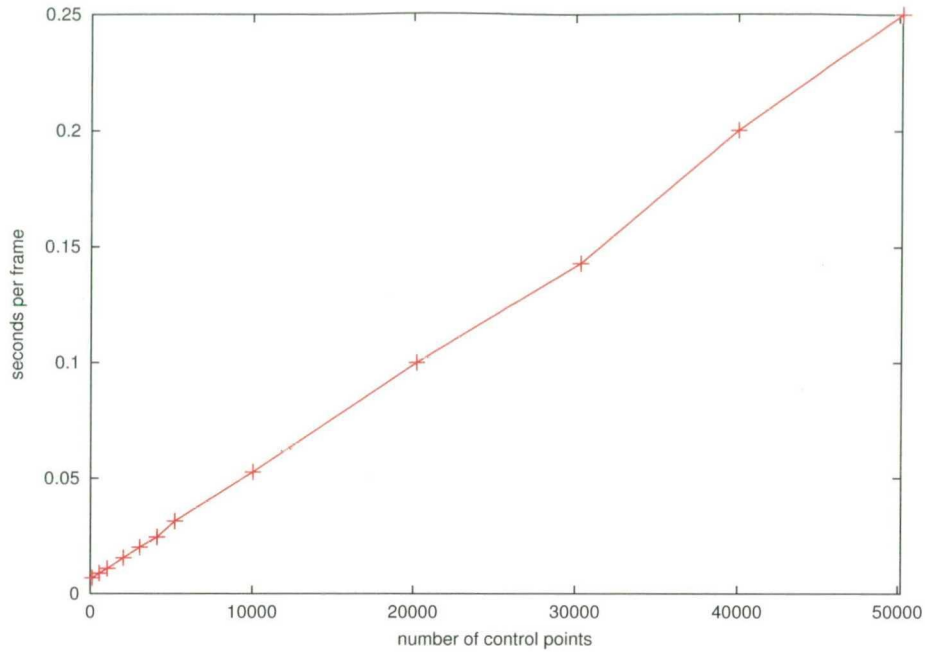

Figure 5.3: A plot of the performance of the surface deformation algorithm. The x-axis shows the number of control points being deformed, and the y-axis shows the time taken to execute the surface deformation algorithm once.

In such an environment, the muscle system is just one of many animation techniques being concurrently applied to the model.

The performance measurements in this section demonstrate that the performance constraint placed on the surface deformation algorithm has been met. Not only does the algorithm run in linear time, but it is fast enough that the results of changing a parameter of the system may be visualized immediately.

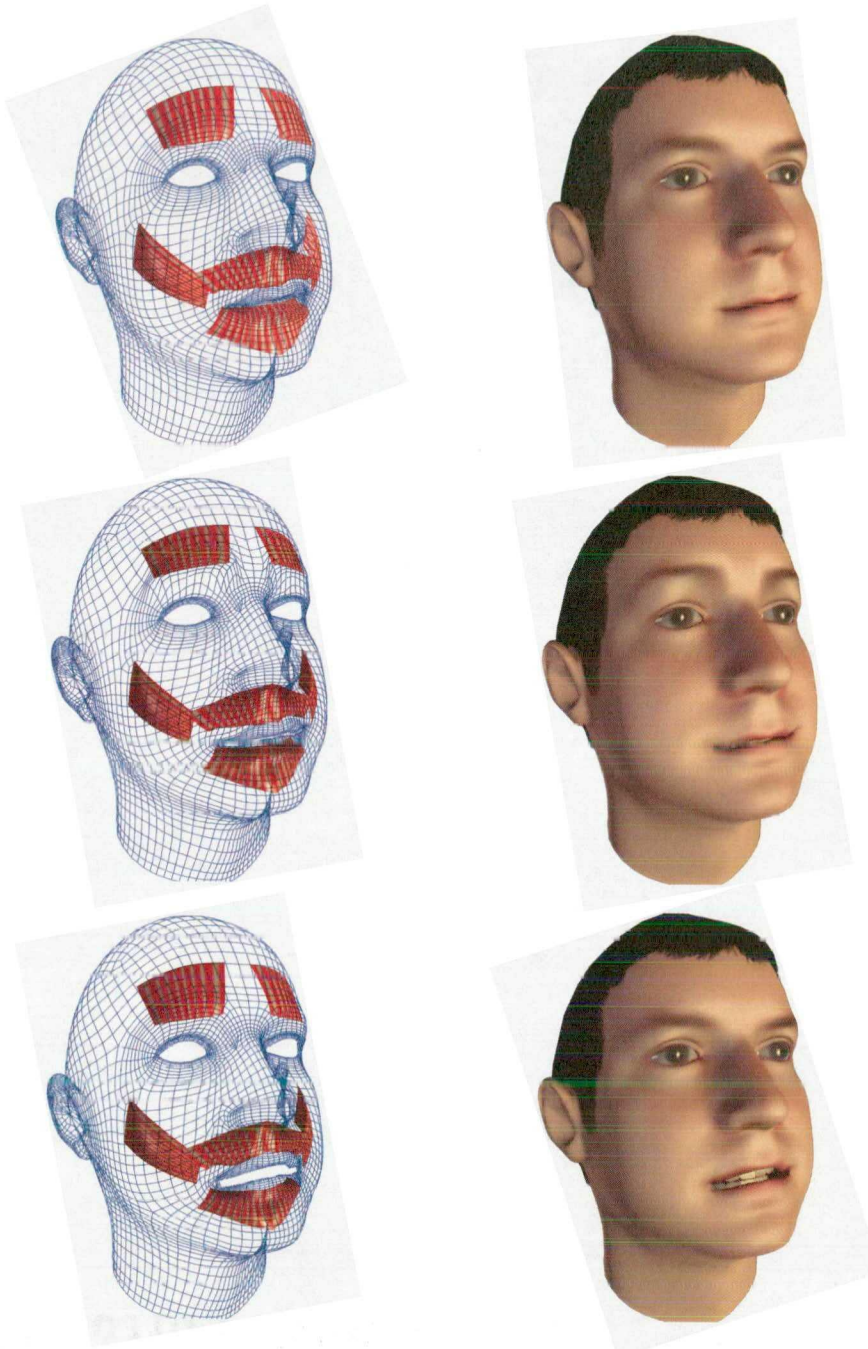

Figure 5.4: Frames from an animation sequence of a human head. Multiple muscles surround the mouth to approximate a sphincter muscle.

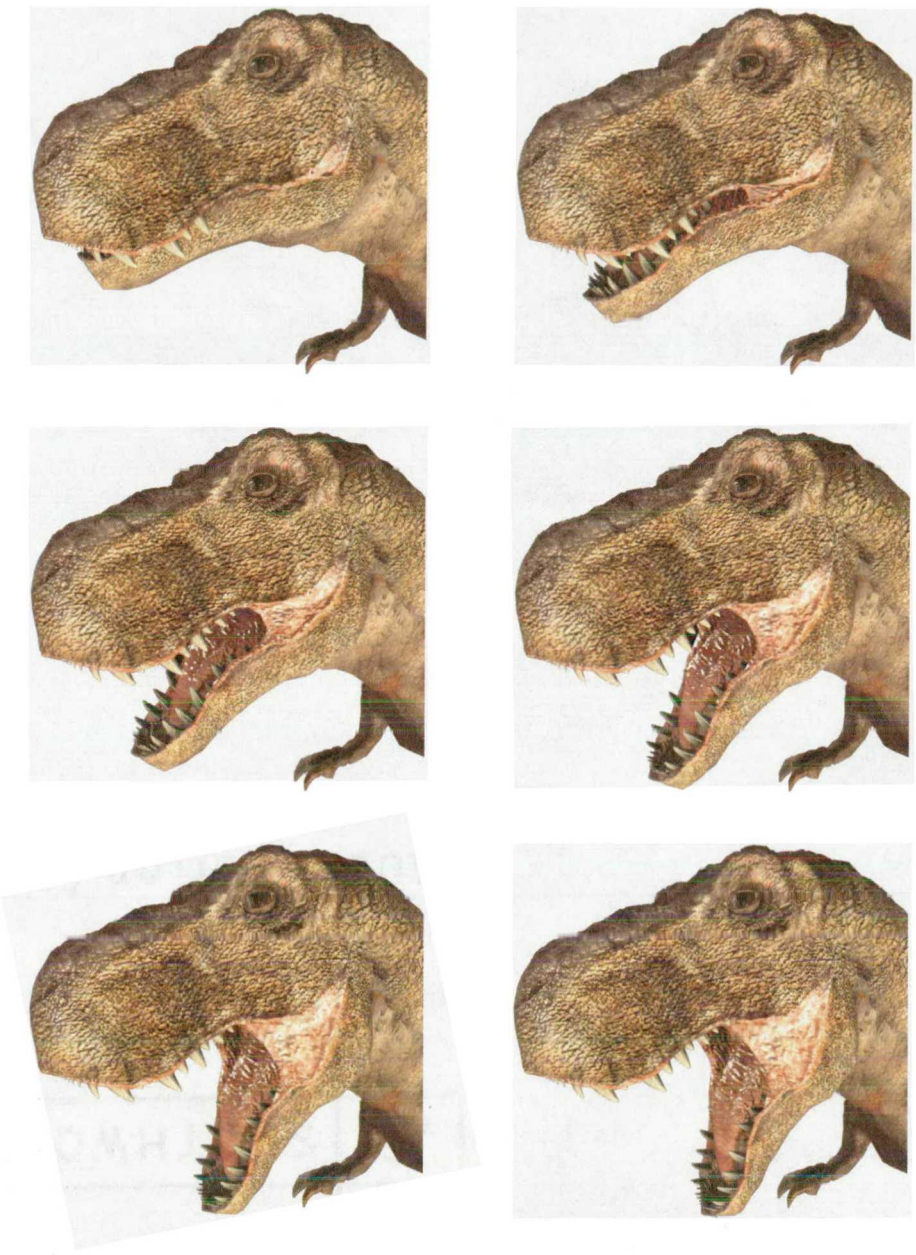

Figure 5.5: Frames of a tyrannosaurus rex from a fully animated sequence. The animator had direct control over the jaw, and the muscle movement in the cheek region was driven by the jaw joint.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This document has presented a system for simulating skin deformation by modeling the underlying musculature. By modeling reality rather than using ad-hoc methods, a greater degree of realism is obtained.

The system is able to automatically generate muscles based on existing surface geometry. This ability allows characters to be constructed in a top-down fashion, in which the character models are created first and the muscles are added later. Each automatically generated muscle is represented by a first-order continuous parametric surface. The muscle surface lies within an area defined by user input. Furthermore, the shape of the muscle will automatically conform to the topology of the skin surface.

The muscles are able to be contracted in order to influence the skin. The model simulates isotonic contraction, in which the muscle thickness increases as the length decreases. The increase in thickness results in a bulge, the amount of which can be customized by the artist in order to satisfy a variety of visual requirements. Both the contraction and the bulge factor can be keyframed by the artist in order to produce animation. Isometric contraction, in which the muscle increases in thickness without changing in length, may also be approximated by modifying the bulge of the muscle independently of the length.

Bones are not explicitly represented in this model, but their effect on the overlying muscles is simulated. While contracting, a muscle does not penetrate deeper into the character's body than its initial configuration. This preserves the influence of the bones as the muscles slide across them.

As muscles contract and bulge, they influence the skin above them. In this system, the skin deformation is done in a manner which trades off pure physical accuracy in order to gain speed and applicability to a large variety of geometry. Especially important is the ability to work with geometry that does not exhibit first-order continuity. The speed achieved by the algorithm allows the deformation to be performed at interactive speeds, even when the muscle is affecting tens of thousands of control points.

The effect of the muscles on the skin decreases as the distance from the muscle increases. This is modeled by specifying the distances at which falloff begins and ends, and a function that controls the shape of the falloff. These distances and the falloff function are fully under the control of the animator, and may even be keyframed in order to maximize the flexibility of the system.

The system was created in order to fulfill the needs of a production. By using the muscle system, the production was able to generate visually pleasing skin motion in much less time than would be required by other methods, such as free-form deformations and morph targets.

In accordance with the pipeline already in use by the production, the system was developed as a plugin for the Maya 3D animation system. Muscles are implemented using Maya's existing NURBS functionality, and the skin deformation is implemented as a Maya deformer. A clean integration into the existing Maya environment resulted in a short learning curve for the artists.

The surface deformation algorithm used by this model emphasizes stretching of the skin as well as bulging. As a result, the system is particularly applicable to facial musculature. Facial muscles typically insert directly into the epidermis. Therefore, when they are contracted, they pull the skin toward the muscle origin. The skin deformation algorithm also pulls the skin control points toward the muscle insertion. Muscles that do not do much stretching of the skin, such as the biceps, can be simulated by increasing the bulge of the muscle while only making small changes to the muscle's length.

In comparison to other muscle-based techniques, this approach offers increased speed, the ability to handle first-order discontinuous surface geometry, and automatic generation of muscles based on existing surface geometry. Given these advantages, the system would be a good candidate to use when:

- Speed is important. Such a situation can arise when resources are limited, whether the resources are computing power or time.

- Muscles need to be added after the surface geometry has been modeled. One example of this is when surface data is obtained from the real world, such as with a laser rangefinder. Another example is when the appearance of the character is designed by an artist, and must look a certain way.

- When the surface geometry is not smooth or could otherwise not be handled by other algorithms. The dinosaur models shown in this document are examples of such geometry.

When compared to techniques for skin deformation that are not anatomically based, such as free-form deformations and morph targeting, a major advantage of

this approach is the time saved in setting up the character. These other methods allow considerable flexibility in deforming a character, but are very time-consuming to set up. Furthermore, some skin stretching results may be impossible to create with FFDs. Although any skin movement can be created with morph targets, each desired skin configuration must be manually modeled, which is highly time-intensive. Fortunately, if morph targeting is desired, this system can be used in the production of the targets, which improves the speed of that approach.

A number of avenues for future research are available. In order to further automate the muscle creation process, a system could be developed that would take a character's skeleton as input and automatically create muscles for it. The locations for these muscles could be based on an analysis of the skeleton's joint structure and the skin that is bound to the skeleton. Muscle and skin depth could be automatically determined based on the distance of the skin from the skeleton. Such technology could greatly reduce the amount of time required to prepare a character model for animation.

Similarly, a generic facial muscle layout could be applied to a head model by identifying various feature points. Important features would likely include the mouth, nose, ears, and eyes. Once these features have been identified, their locations could be used to determine the positions of facial muscles. Identifying these feature points could either be done by the user, or possibly automated in some way.

The skin deformation algorithm performs the same algorithm independently on many different inputs. This SIMD (single instruction multiple data) behavior makes the algorithm ideally suited for parallelization. Recently there have been a number of techniques proposed which harness the power of the processors on consumer graphics

hardware. These processors are built to perform SIMD tasks. It is likely that performance could be increased by offloading this task from the main processor to the processor on the video hardware.

A number of the parameters in the muscle system are constant across the entire muscle. Some examples of such parameters are τ_m, the thickness of the muscle, and b, the bulge factor. Allowing these results to vary across the muscle surface could increase the flexibility of the system. This could be done by allowing the user to specify the values at some locations, and using these given values to interpolate the rest of the values.

BIBLIOGRAPHY

[1] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 612–619. ACM Press, 2002.

[2] Jules Bloomenthal. Medial-based vertex deformation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 147–151. ACM Press, 2002.

[3] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252. ACM Press, 1989.

[4] David T. Chen and David Zeltzer. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 89–98. ACM Press, 1992.

[5] Paul Ekman and Wallace V. Friesen. *Facial Action Coding System.* Consulting Psychologists Press, 1978.

[6] Joe Fordham. Middle earth strikes back. *Cinefex*, 92:71–142, 2003.

[7] Kolja Kähler, Jörg Haber, and Hans-Peter Seidel. Geometry-based muscle modeling for facial animation. In *Proceedings Graphics Interface*, pages 37–46, 2001.

[8] Rolf M. Koch, Markus H. Gross, Friedrich R. Carls, Daniel F. von Büren, George Fankhauser, and Yoav I. H. Parish. Simulating facial surgery using finite element models. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 421–428. ACM Press, 1996.

[9] K. Komatsu. Human skin model capable of natural shape variation. *The Visual Computer*, 3(5):265–271, March 1988.

[10] Paul G. Kry, Doug L. James, and Dinesh K. Pai. Eigenskin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–159. ACM Press, 2002.

[11] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic modeling for facial animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 55–62. ACM Press, 1995.

[12] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172. ACM Press/Addison-Wesley Publishing Co., 2000.

[13] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 181–188. ACM Press, 1996.

[14] Nadia Magnenat-Thalmann and Daniel Thalmann. Human body deformations using joint-dependent local operators and finite-element theory. pages 243–262, 1991.

[15] Overview of the mpeg-4 standard. Technical report, ISO/IEC, March 2002.

[16] L. Nedel and D. Thalmann. Modeling and deformation of the human body using an anatomically-based approach. In *Proceedings of the Computer Animation*, page 34. IEEE Computer Society, 1998.

[17] Frederick I. Parke. Computer generated animation of faces. In *Proceedings of the ACM annual conference*, pages 451–457. ACM Press, 1972.

[18] Stephen M. Platt and Norman I. Badler. Animating facial expressions. In *Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 245–252. ACM Press, 1981.

[19] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. *ACM Trans. Graph.*, 22(3):578–586, 2003.

[20] Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 163–172. ACM Press/Addison-Wesley Publishing Co., 1997.

[21] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160. ACM Press, 1986.

[22] Karan Singh and Eugene Fiume. Wires: a geometric deformation technique. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 405–414. ACM Press, 1998.

[23] Demetri Terzopoulos and Keith Waters. Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer Animation*, 1:73–80, December 1990.

[24] Daniel Thalmann, Jianhua Shen, and Eric Chauvineau. Fast realistic human body deformations for animation and vr applications. In *Proceedings of the 1996 Conference on Computer Graphics International*, page 166. IEEE Computer Society, 1996.

[25] R. Turner and D. Thalmann. The elastic surface layer model for animated character construction, 1993.

[26] Lawson Wade and Richard E. Parent. Fast, fully-automated generation of control skeletons for use in animation. In *Proceedings of the 2000 Conference on Computer Animation*, page 164. IEEE Computer Society, 2000.

[27] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 129–138. ACM Press, 2002.

[28] Keith Waters. A muscle model for animating three-dimensional facial expression. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 17–24. ACM Press, 1987.

[29] Jane Wilhelms and Allen Van Gelder. Anatomically based modeling. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 173–180. ACM Press/Addison-Wesley Publishing Co., 1997.

[30] Eugene Wolff. *Anatomy for Artists*. H. K. Lewis & Co., London, 4th edition, 1968. Illustrated by George Charlton.

[31] Jin xiang Chai, Jing Xiao, and Jessica Hodgins. Vision-based control of 3d facial animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 193–206. Eurographics Association, 2003.

[32] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 247–253. ACM Press, 2003.