PHYSICALLY-BASED B-SPLINE SURFACE SCULPTING

by

Colette J. Mullenhoff

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

The University of Utah

December 1998

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Colette J. Mullenhoff

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Elaine Cohen

Chris Johnson

Peter Shirley

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Colette J. Mullenhoff in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Elaine Cohen
Chair, Supervisory Committee

Approved for the Major Department

Robert Kessler
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

Conventional B-spline modeling software offers the designer shape interaction and manipulation through editing the associated control points, orders, and knot vector parameters. Such modeling is indirect in that the user manipulates algebraic rather than geometric parameters. Indirect modeling frequently proves to be difficult and tedious, especially for novice designers. Physically-based sculpting has the potential to provide direct shape interaction. In this context, physically-based methods incorporate physical laws into shape representation in order to manage shapes as material objects. To date, physically-based manipulation of B-spline represented shapes is not fully realized.

The goal of this research is to eliminate the need for the user to directly manipulate B-spline parameters by providing higher-level surface design tools based on physical techniques. The result is a means of surface interaction that excludes the need to thoroughly understand the underlying surface parameterization. Resulting shapes are deformed surfaces, sculpted by the designer, and therefore not defined by exact specifications such as radii, angles, or lengths.

A set of physics-based design tools for defining surface properties and forces are presented, along with tools for defining geometric constraints including the novel surface area constraint. Shape operators based on these tools have been designed and integrated into the Alpha_1 modeling system. Methods have also been realized to correlate between B-spline shape parameters and the physics-based sculpting framework. Several non-trivial surfaces were designed without directly editing B-spline parameters, demonstrating the success of this modeling scheme.

To my parents Donald and Pamela.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author would like to thank the chairperson of the supervisory committee, Prof. Elaine Cohen, for the advisement and for encouraging the author to do challenging research. Thanks to the other committee members, Prof. Chris Johnson and Prof. Peter Shirley, for their support and for sharing valuable knowledge. Thanks also to Prof. Frank Stenger and Prof. Peter Alfeld for helping the author understand and solve some difficult math problems involved with this research. Prof. Rich Riesenfeld deserves thanks for supporting the author in the Alpha_1 group and always keeping a smile on her face. The Alpha_1 research staff members Russ Fish, Mark Bloomenthal, and Dave Johnson also provided invaluable assistance. Thanks also to Hank Driskill and William Welch, who helped the author via email.

Special thanks go to the authors friends and graduate student colleagues. In particular, Patrick Tullmann deserves credit for providing much moral and technical support, for creating the thesis defense demo video, and so much more. Other great and helpful friends include Michelle Miller, Amy and Bruce Gooch, Tim Jacobs, Dean Brederson, Dave Pugmire, and many more. Thanks also to Tom Thompson, Bill Martin, the SCI research group members Steve Parker, Dave Weinstein, Peter-Pike Sloan, Leonid Zhukov, Yarden Livnat, Dave Beazley, and Ruth Klepfer for their intellectual involvement with problems the author brought to their attention.

This work was supported in part by DARPA (F33615-96-C-5621) and/or the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219). All opinions, findings, conclusions, or recommendations expressed in this document are mine and do not necessarily reflect the views of the sponsoring agencies.

CHAPTER 1

INTRODUCTION

Recently, physically-based modeling tools have been introduced with the goal of putting the laws of physics to work on behalf of the designer. Such tools incorporate physics-based properties, such as bending and stretching, into the shape representation. Using the hypothesis that humans are accustomed to, and have expectations of how physical objects behave, the computational methods adapt physical attributes into modeling operations.

The Alpha_1 research testbed modeling system [32], actively being developed at the University of Utah, is a modeling environment for computer graphics, visualization, engineering design, and computer-aided manufacturing. Models in the Alpha_1 system are created using the B-spline [10] surface representation. B-splines, noted for unified geometric shape representation, are a common and efficient means of representing complex shapes in many modeling environments.

The B-spline formulation for curves and surfaces is flexible, consisting of many low-level parameters such as polynomial degree, location of control points, and knot vector values; these parameters have local or global control over the B-spline. Sculpting B-spline represented shapes by modifying the accompanying parameters is often an indirect and tedious process. Physically-based sculpting methods potentially provide a more direct and less tedious design alternative.

Difficulties arise with direct sculpting of B-spline surfaces, due to artifacts from the parameterization of the surface. Figures 1.1 and 1.2 show that the same shape with different parameterization can have different results under the same modification. Typical B-spline modeling environments employing physical sculpting methods, automatically adjust the control points and ignore the other degrees of freedom. When these physically-based tools allow the designer to grab and pull

(a) (b)

Figure 1.1. An order three B-spline curve with uniform knot vector is modified by one control point to show resulting curve. (a) Order three curve with seven control points and knot vector $\{0, 0, 0, 1, 2, 3, 4, 5, 5, 5\}$. (b) The curve with its center point modified.

(a) (b)

Figure 1.2. An order three B-spline curve with the same shape as Figure 1.1(a), and different parameterization is modified in the same manner, yet achieves different results. (a) Order three curve with seven control points and knot vector $\{0, 0, 0, 2.1, 2.3, 2.6, 4.5, 5, 5, 5\}$. This curve has the same shape as Figure 1.1(a). (b) The curve with its center point modified the same as in Figure 1.1(b). These modified curves do not look the same.

on a surface point for example, the result entirely depends on the initial B-spline surface parameterization. If the desired shape is not achieved, the designer must usually modify the remaining degrees of freedom, and try again. A physically-based sculpting environment should not concern the user with surface parameterization issues.

To offer the designer physically-based B-spline parameter-independent surface sculpting tools, the system must correlate between physical properties and the surface representation. The resulting effects of sculpting a material surface are dependent upon the surface's properties. Does the designer want the surface to imitate spandex, putty, or other material? In addition, the system must process that information to appropriately adjust the necessary degrees of freedom of the surface while effects are being applied. Due to these complex issues, realizing a physically-based B-spline surface sculpting environment continues to be a difficult research problem.

This research presents physically-based design tools for direct B-spline shape manipulation with automatic control of the low-level surface representation parameters. The editing operators provided by the system are force effects, specifically *springs*, *gravity*, and *pressure*. Spring and gravity forces are defined by a direction and strength value. Pressure force is defined by a strength per unit area value. Spring forces have one end attached to a surface location and pull on the surface with a given direction and strength. Gravity attracts an entire unconstrained surface region in one direction with the desired force, while pressure pushes on a surface region with the defined force in the normal direction of the surface.

Suboperators in this research are *constraints* and *surface properties*, which are additional aids for the higher level force operators. Constraints restrict changes to the surface, while properties describe surface behavior under applied forces. There are four constraint types: *point*, *curve*, *surface normal*, and the novel *surface area*, as well as two surface properties: *stretch* and *bend*. One additional option allows for collision avoidance with combinations of static spheres during the sculpting process.

This research accomplishes the critical goal of integrating physically-based modeling techniques into a unified surface sculpting environment which frees the user from manipulating low-level B-spline parameters.

CHAPTER 2

BACKGROUND

This work is based on concepts from physics and previous research in computer-aided design and computer graphics. This section is an overview of the relevant work related to the application domain. Different approaches to shape behavior and shape modification are presented, as well as methods for applying and editing them.

2.1 Shape Description

The notion of removing surface representation parameters from the interactive modeling process is a topic of much interest. Research efforts on that topic have had varying levels of success. First, related work in geometric constraints is described, followed by surface properties and force effects.

2.1.1 Geometric Constraints

Constraint enforcement offers additional control of a shape during the modeling process. Typical examples of constraints include point, curve, and normal constraints (see Figure 2.1). A point constraint generally requires a surface to interpolate a specified point during surface manipulation. Similarly, a curve constraint is a curve that lies within the surface. And a normal constraint imposes the requirement that a surface normal direction remain fixed at a particular surface point.

Bartels and Beatty [2] introduced the notion of picking *any* point on a B-spline curve and changing its location. The resulting curve shape is computed by minimizing control point offset. In effect, curves are *constrained* to pass through locations as the user edits them.

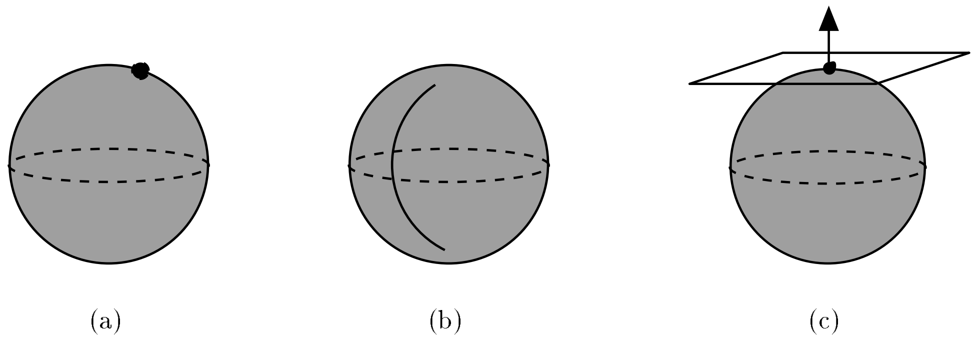


Figure 2.1. Examples of (a) point, (b) curve, and (c) normal constraints applied to spherical surfaces.

The interactive approach to free-form surface modeling presented in Welch and Witkin [33] involves "handles" for interactively manipulating an initial surface. One such "handle" allows for point constraints that can either be "control points," that appear as push-pins on the surface, or surface normal constraints. Surfaces are linearly constrained to contain the point, or surface normal at the point. Curves are another handle to control surface shape and can be attached to a surface, or can join two surfaces together. The surface is linearly constrained to optimally fit the curve.

Fowler's [12] research for tensor product surface manipulation is similar to that of Welch and Witkin [33]. Fowler's work offers direct manipulation of geometric properties to shape the surface. These geometric properties serve as the editable surface constraints and include properties that depend only on first-order derivatives, such as interpolating surface points and surface tangents.

Similarly, Greiner and Loos' [17] present point interpolation constraints and curve interpolation constraints using tensor-product B-splines. Their work allows the user to constrain any isoparametric curve; that is, a curve for which either u or v is constant. Fortunately, constraining a curve to remain fixed is an exact interpolation.

Gortler and Cohen [16] developed geometric modeling using wavelets, defining point constraints and point tangent constraints similar to those in Fowler [12] and Welch and Witkin [33].

Terzopoulos and Fleischer [28] present finite elements to demonstrate how real materials undergo inelastic deformation. A *yield condition* is imposed on a model to describe its elastic behavior. When the condition is exceeded through forces, then the model may behave inelastically and possibly fracture.

Free-form shape design of finite elements in Celniker's [4] work offers geometric constraints for edges of triangles composing the surfaces. These constraints include edges that are pinned at specified points, edges that remain fixed with changing normal vector direction, and edges that are constrained by both shape and normal vector direction.

Celniker and Gossard's [5] free-form modeling paradigm for finite elements enforces linear geometric constraints of point locations, point tangents and normals, curves on and along edges of a surface, and normals along edges or on curves within a surface.

Interactive sculpting techniques on tensor-product B-spline surfaces presented in Celniker and Welch [6] derive and enforce linear geometric constraints. These constraints include point, curve, and normal direction constraints, which are all linear functions of B-spline surface parameters. In addition to using the same methods as in Welch and Witkin [33] to optimally constrain modifiable curves on the surface, fixed curves are also constrained. Qin [25], and Terzopoulos and Qin [30] incorporate these same linear geometric constraints into the Dynamic NURBS representation and add weight constraints on the control mesh points.

Constraints prove to be an effective tool both for sculpting models and for animating real behaviors of objects. Constraining linear geometric properties of a surface, specifically surface points, curves, tangents and normals, seem to serve as an attractive set of useful constraints.

2.1.2 Surface Properties

Surface properties are important for a physically-based design environment. In order to predict how a constrained shape will react to applied forces, the model must be given guidelines on how to behave.

While Welch and Witkin's [33] interactive modeler maintains imposed constraints, the surface is calculated to be as smooth as possible. To compute what they consider smooth, fair, graceful shapes, *objective functions* are satisfied to regulate the bending and stretching of the surface. These *objective functions* are imposed over the entire surface, and thus do not allow for local control. A default objective function is set for surface smoothness.

Celniker and Welch [6] also attempt to create fair shapes. Fair, graceful shapes are achieved by minimizing how much a surface can stretch and bend, as in Welch and Witkin [33]. This process is described as minimizing the amount of energy stored in the surface. Similar methods for minimizing surface energy are also employed in other research [4, 5, 16, 17, 25, 30]. The deformation energy, or the energy functional in Celniker [4], Celniker and Gossard [5], Celniker and Welch [6], and similarly in Welch and Witkin [33] has the form:

$$\mathrm{E}_{\mathrm{deformation}} = \int_{\sigma} (\alpha \ \ \mathrm{stretch} + \beta \ \ \mathrm{bend})\, \mathrm{d}\sigma, \tag{2.1}$$

where α and β are the stretching and bending weights, and the stretch and bending terms in the equation are the shape's representation of them. Minimizing this equation will produce a surface that resists stretching and bending.

The sculpting framework in [12] constrains geometric properties based on first-order derivatives such as tension. Tension is the magnitude of the first derivative of a surface and serves as a stretching parameter. Fowler [12] offers both *uniform tension*, which uniformly scales both partial derivatives, and *directional tension*, which is in a user-defined direction in the tangent plane. Geometric properties can be applied and modified at any selected point on the surface.

The elastically deformable finite element models presented in [22, 28, 29] are also governed by surface energy. Surface properties can be defined, such as those of rubber, cloth, paper, and springy metal. The values for bending and stretching parameters are adjusted accordingly, and can be set independently for each surface point. This ability allows for local control of the surface; for example fractures and creases. The elastic parameters are calculated into the energy function of the surface, as in previous noted research.

Qin [25] and Qin and Terzopoulos' [30] dynamic NURBS (Non-Uniform Rational B-Splines), or D-NURBS, are animated as a function of time using equations of motion. With time, the behavior of the surface is also governed by physical surface parameters such as mass and damping distributions. Other research [6, 28, 29] also employs the equation of motion with mass and damping parameters. Using the applied forces described in the next section, the equation of motion is:

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{B}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}, \tag{2.2}$$

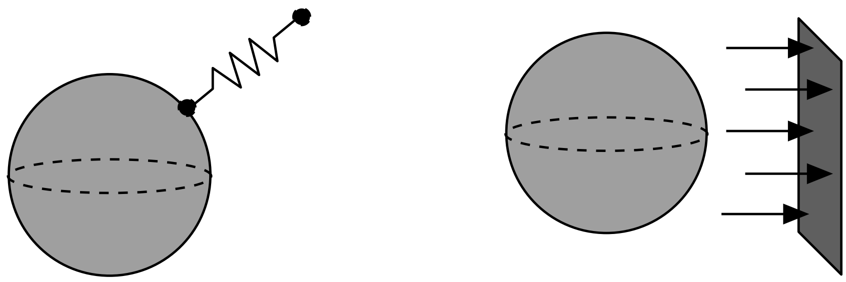
where $\mathbf{M}$ is the mass density, $\mathbf{B}$ is the damping term, $\mathbf{f}$ is the applied forces, and $\ddot{\mathbf{p}}$, $\dot{\mathbf{p}}$, and $\mathbf{p}$ are the acceleration, velocity, and location of the degrees of freedom of the surface. $\mathbf{K}$ represents stiffness, and is derived from the *thin-plate-under-tension* energy model [5, 29, 33] describing elastic potential energy, which is an extension of the energy functional in Equation 2.1:

$$\mathbf{K} = \iint (\alpha_{1,1}\mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2}\mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1}\mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2}\mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2}\mathbf{J}_{vv}^\top \mathbf{J}_{vv})\, du\, dv \tag{2.3}$$

where the $\alpha_{i,j}$ terms control the directional resistance to stretching, and the $\beta_{i,j}$ terms control the directional resistance to bending. $\mathbf{J}$ is the Jacobian matrix described in Appendix A and the subscripts on $\mathbf{J}$ represent partial derivatives.

2.1.3 Effect Forces

Application of forces is the key to changing the shape of a model in many systems. The free-form modeling techniques in Celniker and Gossard [5] use what they term sculpting "loads" or "forces" to create shape modeling effects. Forces are a function of time over the surface shape. In Celniker [4] and Celniker and Welch [6], the types of sculpting forces presented include pressure, springs, and gravity for the effects of enlarge, attract, and flatten (see Figure 2.2 and Figure 2.3). Springs attract a



(a) (b)

Figure 2.2. Example effect forces applied to spherical surfaces. (a) Spring force. (b) Gravity force.

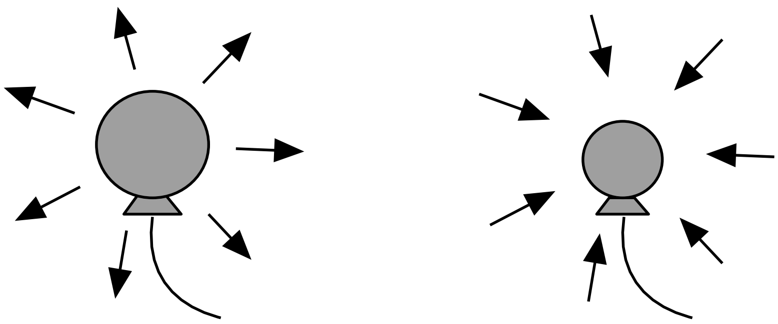


Figure 2.3. Pressure forces can "inflate" and "deflate" a shape.

surface point to a point in space, gravity pulls on the surface only in one direction, and pressure acts in the direction of the surface normal.

Qin [25] and Qin and Terzopoulos [30] support the forces of spring, repulsion, gravitational, and inflation to modify their D-NURBS. Similar forces are offered by Greiner and Loos [17], whose point to point, point to line, and normal direction to direction *attractors* and *repellers* are used to implement surface interaction and manipulation. These attractors and repellers are treated as external energies and are added to the energy of the surface.

The types of external force effects administered in Terzopoulos, Platt, Barr, and Fleischer [29] and Terzopoulos and Fleischer [28] are gravity, viscosity, and collisions. Terzopoulos, Platt, Barr, and Fleischer [29] also include spring forces.

Viscous fluid force is a constant force acting in the normal direction of the surface, much like pressure. The equation of motion is necessary to observe the elastic or inelastic effects of the fluids and collisions.

The popular modeling, animating, and rendering system Softimage 3D [27] offers a vast array of modeling tools. One of their tools, shrink wrapping, is relevant to this research. Imagine a balloon filled with air and with objects of various sizes and shapes. Shrink wrapping occurs when air is released from the balloon, and the balloon forms around the objects inside. In Softimage 3D, the "balloon" surface is represented as a polygon mesh or a NURB, while the objects inside the balloon cannot be represented with NURBS. Other deformation techniques in Softimage 3D can be applied in only limited ways to NURBS.

As mentioned previously, Welch and Witkin [33], Fowler [12], and Gortler and Cohen [16] present surface sculpting techniques that do not use forces. The designer interacts directly with geometric surface parameters such as surface points and surface tangents. Research in Forsey and Bartels [11] offers similar methods of shape modification by editing B-spline control points or provided edit points on the surface. This approach does not meet our goal of eliminating interaction with B-spline parameters.

2.1.3.1 Collisions

Collisions between a changing shape and static objects occur when using Softimage 3D [27] to perform the shrink wrap effect. Unfortunately, the explanation of how this system detects and deals with such collisions are not available in the manuals.

The elastically deformable models in Terzopoulos, Platt, Barr, and Fleischer [29] undergo collision dynamics by creating a potential around each object, which prevents the objects from interpenetrating. Objects deform as a function of time, and thus can be checked at each time step to see whether the energies of the objects are being violated.

Platt and Barr [22] implement *reaction constraints*, to prevent a moving surface from intersecting a polygon. These constraints work by limiting the force applied to the surface.

Thingvold's [31] *eplastics* simulates collisions between a moving surface and static objects; which can be B-spline surfaces or planes. Using a ray tracer, the moving surface is divided into triangular pieces, and then a ray is traced between the moving point locations, checking for intersections. If a collision is detected, the changing model either bounces away from the static object, or the objects stick together. Again, time is included, making for interesting animations of object interaction.

2.2 Sculpting Methods

To sculpt shapes using the force and collision methods described above, interaction methods must be implemented. Some of the previously outlined research discuss little or no designer interaction, while others are completely interactive. The following is a summary of how other research visualizes, applies, and modifies these sculpting options.

Greiner and Loos [17] discuss interactive surface modeling. Point attractors are used to attract a (u, v) surface point towards a point in space. These attractors are visualized as a sphere in space, connected with a line to another sphere on the surface. A point interpolation constraint is also viewed as a sphere. The other design tools, such as point to line and point to plane attractors, and normal and curve interpolation constraints, are not shown or described. The kind of input device used, how end locations, attract and repel intensities, and stretching and bending terms are modified, are also unclear.

The modeler for D-NURBS in Qin [25] and Qin and Terzopoulos [30] offers interactive sculpting of complex shapes. Interaction is offered through applying local and global shape constraints, and simulated forces to a surface, as well as creation of control polygons, and for adjusting control points and weights. Mass, damping, elasticity, and force specifications are edited in control panels. However, the methods for performing any of the described interactivity are not discussed.

Terzopoulos and Fleischer's [28] research for modeling inelastic deformations shows one interactive physically-based example: a model is deformed by a robot hand with sticky fingers. These fingers pull on the model and then let go, showing the deformation. It is unclear how the user manipulates the robot hand.

The interactive sculpting in [6] allows a designer to manipulate pressure force with a slider bar. The paper says it is possible to push, pull, and inflate surfaces but the interaction methods are not presented.

Interactive surface manipulation by Welch and Witkin [33] is performed by the user attaching point and curve handles to a surface, and then moving these handles. Surface points can be selected and moved to new locations, while curves may be inscribed onto the surface and manipulated. A user may also interactively select surface regions for refinement.

Interactive modeling by Gortler and Cohen [16] allows the designer to use the mouse to click on a surface point and drag it to a new location. Tangent constraints at a point on the surface are defined by orienting an arrow icon at that point.

Geometric constraint parameters and sculpting loads are individually attached to slider bars for interactive sculpting in Celniker and Gossard [5]. Further interaction is not discussed.

In Celniker's [4] work, interaction is achieved with input commands. For example, typing in the *pressure* command followed by a pressure magnitude will produce a slider bar for the user to manipulate. This method also allows for the modification of other command arguments, such as forces, loads, constraints, and material properties. In addition, a grid of points on the surface of a model is shown in a separate window. The designer can use the mouse to select points or groups of points in this window on which the loads should be applied. Points are color coded as to whether they are fixed or free to move. When a model becomes complicated, the ability to correlate between points on the surface and points in the second window becomes more difficult.

Fowler [12] addresses many issues of interaction. For experimental purposes, four panels are presented to control position, normal orientation, tension, and the twist vector. They are not intended as useful design tools, but as a test of capabilities. Another panel controls tension and twist. This panel contains two vectors to manipulate uniform or directional tension, which can also re-orient directional tension to apply a twisting effect. A Polhemus 3Space IsoTrak device [21] is offered for both 3D interaction and as a 2D tablet device. The user is also equipped with a hand-held locator for repositioning the surface. This device does not control tension. A VPL DataGlove [9] is another supplied interaction device, where the right hand is used as a sculpting tool. The DataGlove is a thin glove with attached optical fibers (or flex sensors) that measure finger joint angles. Tension is applied by the flex sensors of the four fingers, and the thumb applies orthogonal tension. Clenching the fist applies uniform tension to the surface. The keyboard controls modes of operation for the DataGloves.

Interactive shape sculpting can be implemented in many different ways. Methods range from typing commands to using virtual reality gear.

2.3 Related Work

In the early years of shape deformation techniques, Barr [1] presented operators for transforming geometric objects, such as stretching, bending, twisting, and tapering, which still serve as useful modeling tools in CAD systems today. To create more complex or arbitrary shapes, further modification techniques needed to be developed.

Sculpting surfaces according to Cobb [7] involves operations such as sweep, warp, bend, stretch, and twist. Her work is being used in B-spline based CAD/CAM systems today.

Sederberg and Parry [26] developed the free-form deformation or FFD as a sculpting method for solid models. This method involves attaching a parallelepiped lattice structure around the object to be deformed, and associating the shape to the lattice space. Moving regularly spaced points on the lattice results in a deformation of the model.

Coquillart [8] extended the free-form deformation techniques of [26] to provide user-defined lattice structures. Applying these lattices to an entire model, or to a smaller surface patch, and modifying the control points, allows for arbitrarily shaped deformations.

Thingvold [31] modeled and animated deformable B-spline surfaces by treating the control mesh as point masses connected by hinges and elastic springs. These masses, hinges, and springs served as the physically-based representation of the surface. His work, also an extension of the Alpha_1 system, was one of the first attempts to incorporate elastic and plastic modeling.

2.4 B-splines

The B-spline [10] curve and surface representations are being used in this thesis research. B-splines have a rich history of applications for sculpting shapes [6, 11, 17, 25, 30, 33]. The following describes the B-spline curve and tensor-product surface formulation.

A piecewise polynomial B-spline curve with $n+1$ control points $\{p_0, ..., p_n\}$ is defined by

$$c(u) = \sum_{i=0}^{n} p_i B_{i,k}(u)$$

where u is the parametric variable and $B_{i,k}(u)$ is the ith B-spline of order k. The B-spline basis functions with knot vector sequence $t_0 \leq t_1 \leq ... \leq t_{n+k}$, are defined recursively as

$$B_{i,1}(u) = \begin{cases} 1 & \text{for } t_i \leq u < t_{i+1} \\ 0 & otherwise \end{cases}$$

and for $k > 1$ as

$$B_{i,k}(u) = \begin{cases} \frac{(u-t_i)}{t_{i+k-1}-t_i} B_{i,k-1}(u) + \frac{(t_{i+k}-u)}{t_{i+k}-t_{i+1}} B_{i+1,k-1}(u) & \text{for } t_i < t_{i+k} \\ 0 & otherwise \end{cases}$$

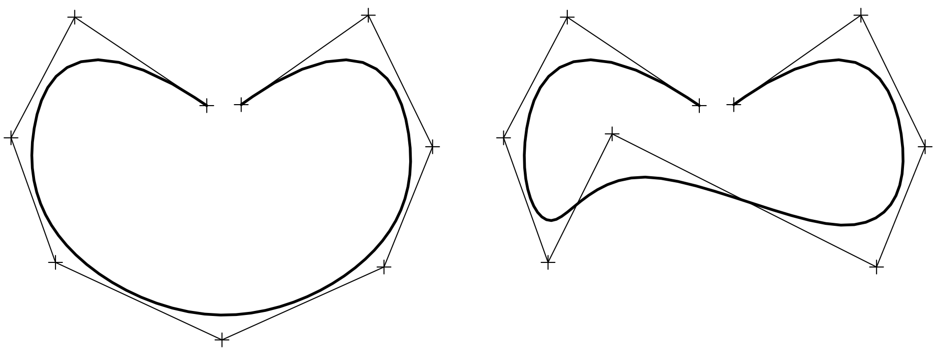
A tensor-product B-spline surface is defined similarly as a generalization of the B-spline curve, with an $(m+1)(n+1)$ control mesh $\{p_{i,j}\}$, and parametric variables u and v, by

$$s(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} p_{i,j} B_{i,k_u}(u) B_{j,k_v}(v).$$

B_{i,k_u} are the B-splines of order k_u on the nondecreasing knot vector $\{u_0, ..., u_{m+k_u}\}$ in the u direction, and B_{j,k_v} are the B-splines of order k_v on the nondecreasing knot vector $\{v_0, ..., u_{n+k_v}\}$ in the v direction.

2.4.1 Refinement

The piecewise quality of tensor-product B-splines allows for the desirable property of local curve and surface control. In other words, manipulating a control point only affects a local region of the curve or surface surrounding that point, and does not necessarily alter the entire surface. Figure 2.4(a) shows an order four B-spline curve with its associated control points and polygon. Displacing a particular control point results in the figure shown in 2.4(b), demonstrating the local control of B-splines. When the local control is not fine enough, more degrees of freedom are needed. When presented with a new knot vector sequence [7], the *refinement* process transforms existing control points and produces new control points, while leaving the curve or surface unchanged. Figure 2.5(a) shows the



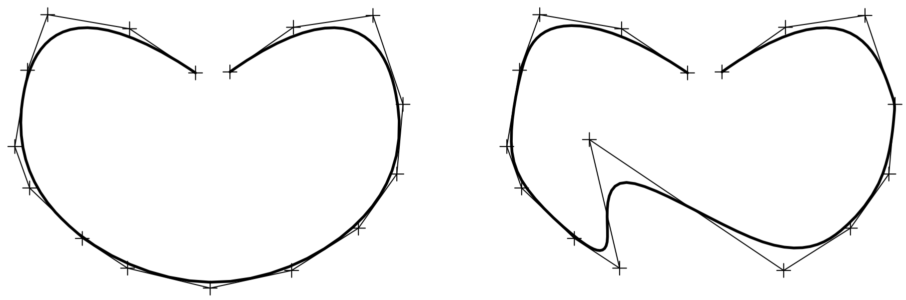
(a) (b)

Figure 2.4. Example of editing one control point of a B-spline curve, showing local control. (a) The original order four B-spline curve. (b) One control point of the curve is modified for a new shape.

refined curve of Figure 2.4(a). These additional degrees of freedom allow for finer tuning of the shape, as demonstrated in Figure 2.5(b).

Forsey and Bartels [11] present a method of local refinement called hierarchical B-spline refinement. A hierarchy of rectangular B-spline surface refinements called *overlays*, are manually applied before manipulating the surface. The surface is then treated as a subdivided surface. Subdivision is the process of treating the surface as if it were divided into smaller surface pieces that can be manipulated independently. An extension of their work would be the ability for the surface to automatically refine as it's being sculpted.

Surfaces in [33] are automatically refined based on constraint error. The refinement method represents the surface as sum of B-spline surfaces at varying levels of detail. *Eplastics* [31] also automatically refines a B-spline surface. When a surface intersects a static object, one new knot value is added at that approximate location. There is a limit to the number of refinements possible for the same intersection. Even though neither of these systems automatically refine surfaces based on surface properties, these works are the only sculpting environments found that implement any automatic B-spline refinement, and thus are very important to the research at hand.



(a) (b)

Figure 2.5. Example of editing a refined B-spline curve by one control point, showing even finer local control than the unrefined curve in Figure 2.4. (a) The refined B-spline curve of Figure 2.4(a). (b) The refined curve with one control point modified, showing finer control.

Robert Hooke, the pioneer of elasticity theory, created Hooke's Law, which is summarized nicely by Fung [13]. The following is a relevant portion of Fung's interpretation:

> Neighboring points remain as neighbors under any loading condition. No cracks or holes may open up in the interior of the body under the action of external load. A material satisfying this hypothesis is said to be a *continuum*. The study of the deformation or motion of a continuum is called the *continuum mechanics*.

Applying this framework implies that the physically-based deformable models in this thesis will always be in continuum and not produce rips or tears, regardless of the user-applied forces and constraints.

CHAPTER 3

MODELING ENVIRONMENT

The main goal of this research is to provide physically-based design tools for direct B-spline shape sculpting that do not burden the user with details of the surface representation. To eliminate B-spline parameter manipulation and the resulting artifacts during modeling, new design methods must be offered, that can create the same shapes as standard B-spline editing techniques. Physically-based tools might make the modeling process easier for some designers. They have the possibility to decrease the learning curve for creating models, since these tools promote the familiar activity of direct shape interaction. This chapter describes the resulting modeling environment that has been integrated into the Alpha_1 framework as a set of physically-motivated design tools.

3.1 Overview

The sculpting environment described in this chapter and seen in Figure 3.1 is written in C++, extended with a Tcl/Tk/[incr Tk] interface. [incr Tk] is a Tk extension that allows for new widgets to be created, using normal Tk widgets as their components. The following interface provides tools for visualizing, applying, and modifying surface properties, constraints, obstacles, and force effects. To create this sculpting environment, 8500 additional lines of C++ code and 2300 lines of Tcl/Tk/[incr Tk] interface code was written.

To keep interaction simple and consistent with Alpha_1, interaction is achieved using a mouse and keyboard. The mouse is used for selecting pulldown menus, buttons, moving sliders, etc, and, in conjunction with keyboard modifiers, interacting with the models; the keyboard is used to type exact values for various parameters. From left to right, the buttons in Figure 3.2 are: spring force, gravity force, inflation

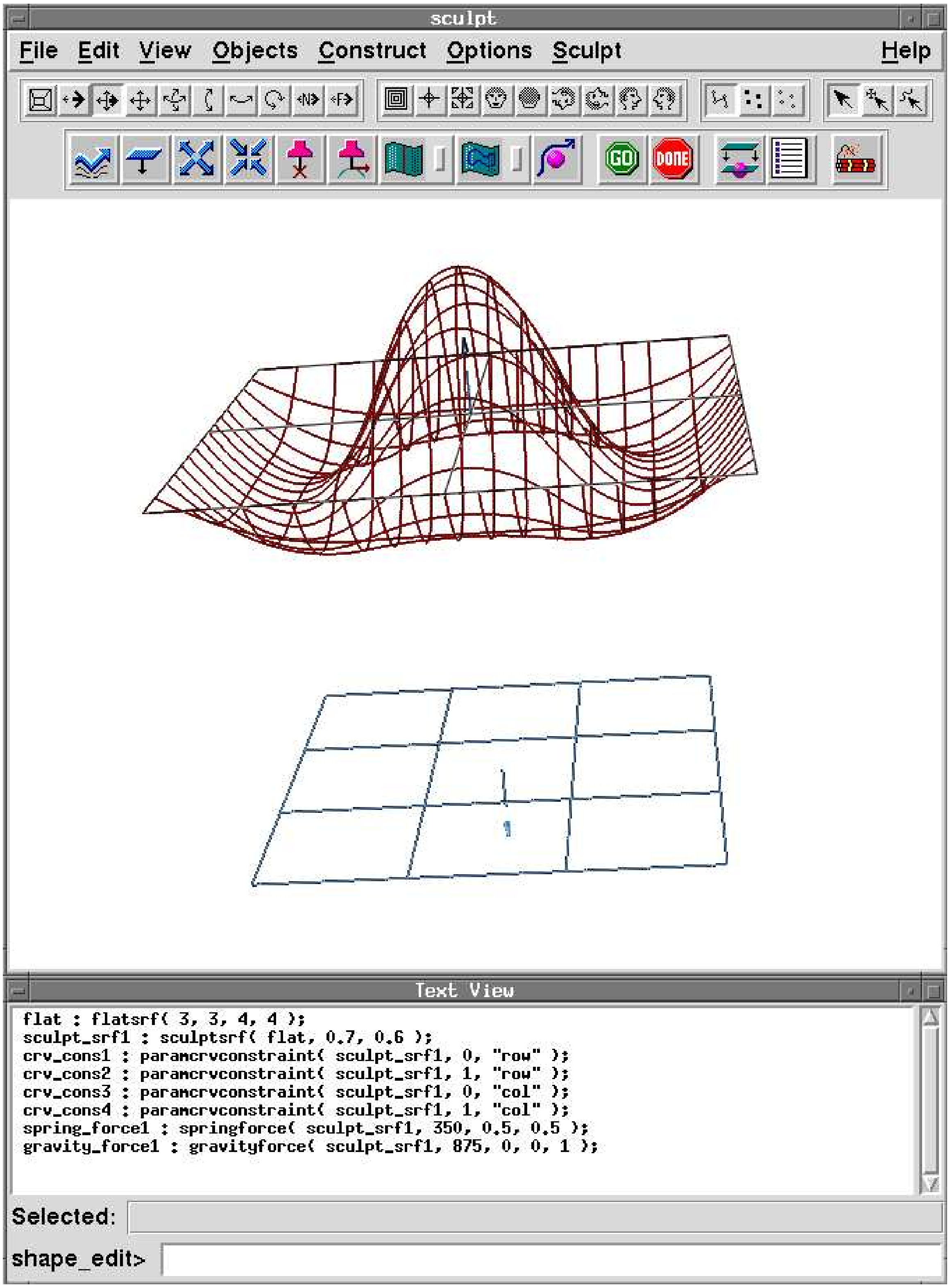


Figure 3.1. The sculpting environment created for this research.

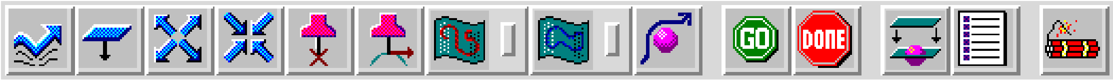


Figure 3.2. The toolbar in this modeling environment that provides sculpting capabilities.

force, deflation force, point constraint, normal constraint, curve constraint pulldown toggle, surface area pulldown toggle, obstacle, begin sculpting, done sculpting, obstacle avoidance toggle, sculpt properties, and remove sculpt operator.

3.2 Surface Properties

The user is provided with *surface properties* that are applied to the shape undergoing modification. As mentioned before, the importance of surface properties in the context of this research, is to offer the designer a way to characterize the material behavior of the surface [4, 5, 6, 12, 16, 17, 22, 25, 28, 29, 30, 33], such as whether it is to act like rubber, steel, or cloth. The user's prior knowledge of how the surface will react under certain conditions facilitates the modeling process. For example, pulling on a small area of a flexible surface could produce a pointy bump, or a rounded bump, depending on the material properties. Most widely used descriptions of surface properties involve parameters governing stretching (tension) and bending (rigidity), which control the *fairness* or smoothness of the surface.

In this environment, a surface to be sculpted is given default bend and stretch property values, which may be customized to achieve the desired material effect. Figure 3.3 shows the surface property dialog containing slider bars for easily changing these bend and stretch values. The property dialog appears when the user presses the **Properties** button or selects it from the pulldown menu. Even though surface properties are not visualized on the surface, they are always present.

Tension and rigidity are uniform over the entire surface being sculpted, as non-uniform material qualities would require a more complex interface and interaction.

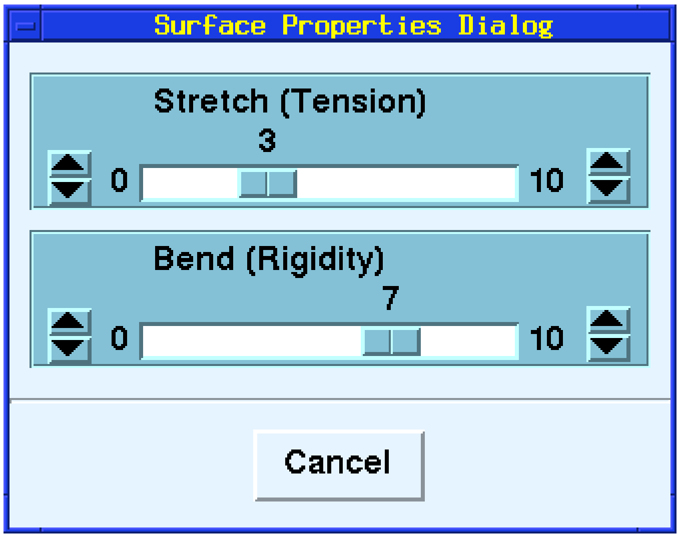


Figure 3.3. The surface properties dialog showing editable sliders for modifying the stretch and bend parameters of a surface being sculpted.

3.3 Constraints

Using constraints as an aid for sculpting models is a routine practice [4, 5, 6, 12, 16, 17, 25, 28, 30, 33]. Constraints are a vital component in the modeling framework in that they restrict surface mobility during sculpting. The constraints supported in previous works are the motivation for the point, curve, and surface normal constraints offered here, which accordingly, will not allow a chosen point, curve, or surface normal direction to change. Not even applied forces can modify a constraint. This section also introduces the novel surface area constraint.

3.3.1 Point Constraints

In this sculpting environment, a point constraint is applied by selecting the **point constraint** button or menu pulldown. The point constraint is visualized as a small sphere (see Figure 3.4) on the surface with the sphere center at the (u, v) surface location of the constraint. The sphere size is computed proportionally to the size of the surface, to ensure that the constraint is always visible. The position of a selected point constraint can be modified using the mouse.

If the user desires additional information or details about a point constraint, he

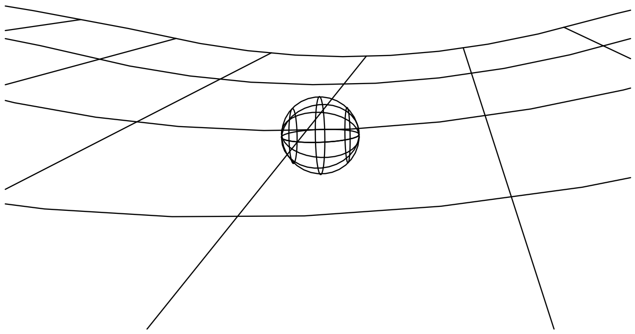


Figure 3.4. Point constraint visualized as a sphere on the surface to be sculpted.

can press the **Properties** button, as for surface properties, or use the appropriate selection in the pulldown menu. The details provided in this dialog (see Figure 3.5) include the XYZ world coordinate and UV surface parameter location of the constraint. This information cannot be changed in the dialog. As the constraint is interactively modified, the dialog information will update accordingly.

3.3.2 Normal Constraints

A normal constraint is applied in a manner similar to the point constraint application, that is, by selecting the **normal constraint** button or pulldown. A normal constraint is visualized as a plane tangent to the surface (see Figure 3.6)

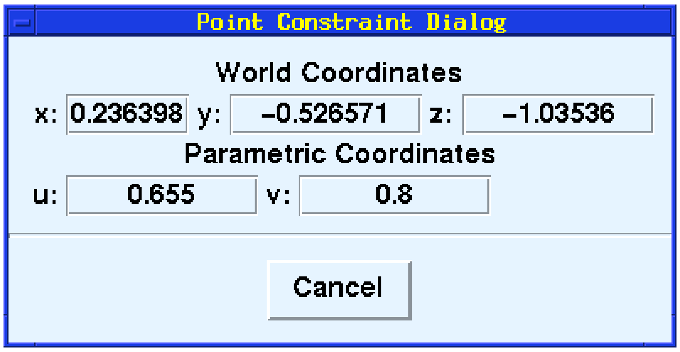


Figure 3.5. A point constraint properties dialog, showing the location of the constraint in both world and parametric coordinates.

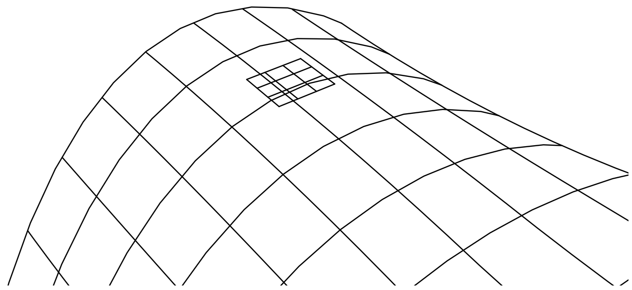


Figure 3.6. Normal constraint visualized as a plane on the surface.

at the (u, v) constraint location. The normal vector is normal to this plane. The position of a normal constraint can be modified on the surface by moving it in the same manner as for a point constraint.

In addition to giving the world and parametric locations of the normal constraint on the surface, the properties dialog box for a normal constraint provides the direction of the normal and the isoparametric tangents along the U and V axis, as in Figure 3.7. This dialog will update as the constraint is interactively modified.

3.3.3 Curve Constraints

Curve constraints include parameter curve constraints as well as freehand curves. An isoparametric curve constraint uses a curve in the surface with constant u or v value. Figure 3.8 shows both isoparametric curve constraints and a freehand drawn curve constraint on a surface. To utilize a curve constraint, first the user must select the type of curve he wants to constrain. A pulldown toggle is provided in the constraints section of the toolbar, so the chosen constraint is viewed in the toolbar (Figure 3.9(a)). The same choices are also available in the pulldown menu (Figure 3.9(b)). For either one of the isoparametric curve constraints, selecting a location on the surface will create the constraint, and thus eliminate the user's need to know a particular u or v parametric value. Dragging the cursor over the surface produces an arbitrary curve constraint.

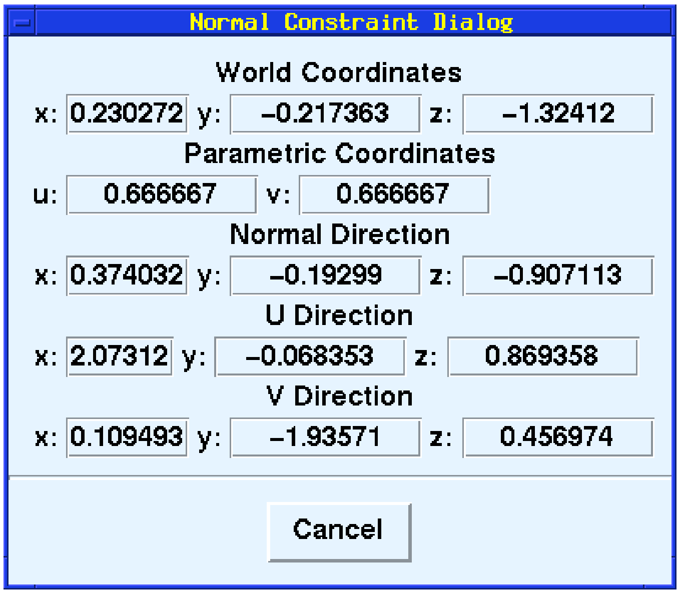


Figure 3.7. A normal constraint properties dialog, giving the location of the constraint in both world and parametric coordinates, the direction of the normal, and the directions of the vectors tangent to the surface.

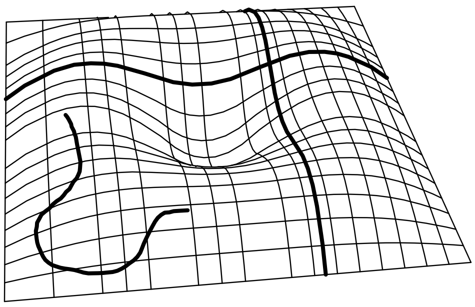
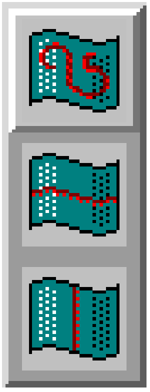
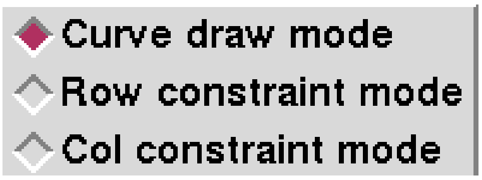


Figure 3.8. Two parameter curve constraints and a freehand drawn curve constraint applied to a surface prior to sculpting.



(a)



(b)

Figure 3.9. Curve constraint types may be chosen by using a pulldown toggle in the toolbar or from radiobuttons in the menubar. (a) Pulldown button toggles for the curve constraint mode choices of freehand drawn, row, or column. (b) Associated radiobutton menu choices for curve constraints.

The constant u or v parameter value of a selected isoparametric curve constraint is modified with mouse movement. Currently, freehand curve constraints cannot be directly edited. Another area of research is involved with the issues of editing freehand drawn curves embedded in surfaces.

3.3.3.1 Surface Areas

Surface areas are a novel type of constraint. A surface area constraint does not allow the entire area on the outside of a closed curve to move. Only the portion of the surface inside the constraint is malleable.

There are two ways to apply a surface area constraint. The user can either draw a freehand area, or create a circular area. These two choices are available through a pulldown toggle and the menu, as with curve constraint choices. A freehand surface area constraint is created in the same way as an arbitrary curve constraint. However, if the drawn curve is not closed, the system automatically closes it and so creates the surface area. The circular surface is created by selecting a point on the surface, which becomes the circle center, and then dragging the mouse to a point on the desired boundary. When the user is done, the circular boundary will show up on the surface as a closed surface area.

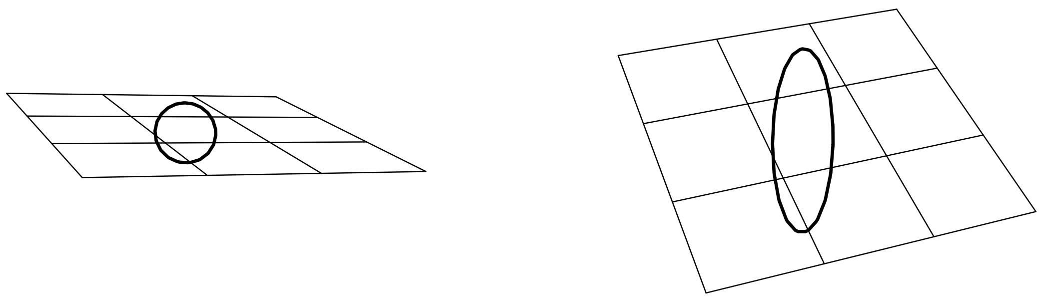
Figure 3.10(a) shows a circular surface area constraint created on a flat surface and viewed at a particular angle. When viewing the surface at a different angle, as in Figure 3.10(b), the surface area constraint looks different. This interface can create various surface area constraints depending on the orientation and shape of the surface.

As with freehand curve constraints, there are no methods implemented currently for modifying surface area constraints once drawn on the surface. However, undesirable surface area constraints can be removed quickly and redrawn.

3.4 Obstacle Avoidance

Changing the shape of an object while considering other objects in the scene has been given the terms collision dynamics [29], reaction constraints [22], and shrink wrapping [27]. This idea leads to surfaces that stick to and mold around, bounce off, or are repelled by other objects. In this environment, the surface being sculpted avoids static obstacles. Currently, spheres are the only type of obstacle allowed.

A sphere obstacle is created by specifying a center point and a radius. A sphere can be relocated using the mouse, and the radius can be changed using keyboard



(a) (b)

Figure 3.10. A circle surface area applied to a surface before sculpting. (a) A circle surface area is applied to a surface. (b) The same surface with its applied circle surface area looks different at another angle.

input. Multiple, independent obstacles may exist in the scene, as in Figure 3.11. The interface provides a toggle button (see Figure 3.12) for choosing to either ignore or recognize obstacles.

3.5 Effect Forces

To create new shapes, the designer needs to be equipped with techniques for modifying the surface. Direct manipulation of surface points and curves [33], or geometric property constraints [12, 16] can accomplish simple effects. Other effects, such as enlarging an entire model or applying other global surface modifications could require significant effort. Editing individual control points, as in [11] is not an option as the goal is to eliminate interaction with B-spline surface parameters.

This research presents an effects generator. This tool enables the designer to apply different types of force effects to the surface. The types of effects provided are spring, gravity, and pressure forces. These capabilities are important in order

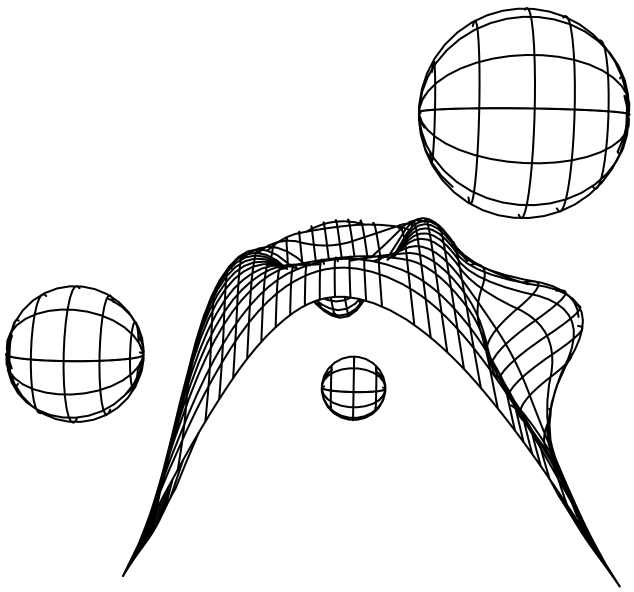


Figure 3.11. A surface to be sculpted, with three sphere obstacles set up in the scene for the surface to avoid.

(a) (b)

Figure 3.12. Collision avoidance toggle choices from the toolbar. (a) Collision avoidance is off. (b) Collision avoidance is on.

to modify the shape of an object using physically-based techniques. Each effect is a unique tool for surface modification. Spring forces are local, while gravity and pressure forces are global.

3.5.1 Spring Forces

The idea behind a spring is to attach the two ends to desired locations of objects, and then let go of the spring. Depending on the properties of the spring and the objects to which it is attached, the spring could change the shape of the objects or move them. A *spring force* pulls a surface point in a specific direction. Spring forces are created with the **Spring force** button or pulldown, and are visualized as an arrow attached to the surface at a particular (u, v) location. The point to which a spring force is attached can be modified by moving its symbol around on the surface like a point or normal constraint. Spring forces can be independent, point to a common world-space location, or share a direction vector (see Figure 3.13).

3.5.2 Gravity Forces

This research presents gravity as the concept of an object being attracted in a single direction. In this environment, a *gravity force* is only effective in achieving shape modification when some part of the object is constrained. After creating this force with the **Gravity force** button or pulldown, the gravity force tool appears as a plane with a normal vector attached at the center, as in Figure 3.14. Following mouse motions, this tool rotates around the sculpting surface.

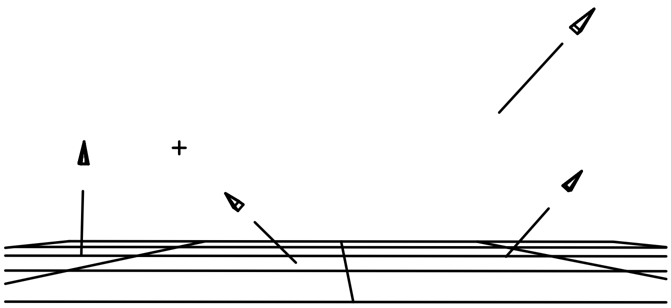


Figure 3.13. Three spring forces applied to a surface. The spring force on the left is pointing in the normal direction to the surface. The middle spring force is pointing to a world-space location. The spring force on the right is pointing in the same direction as the above vector.

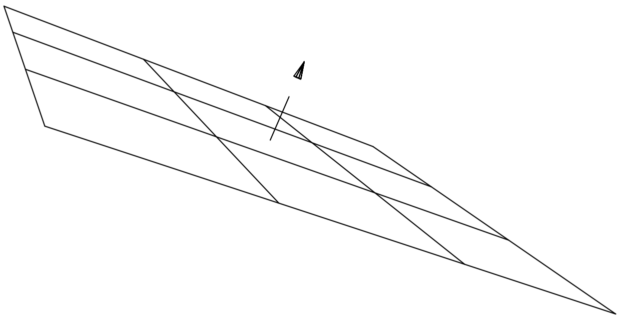
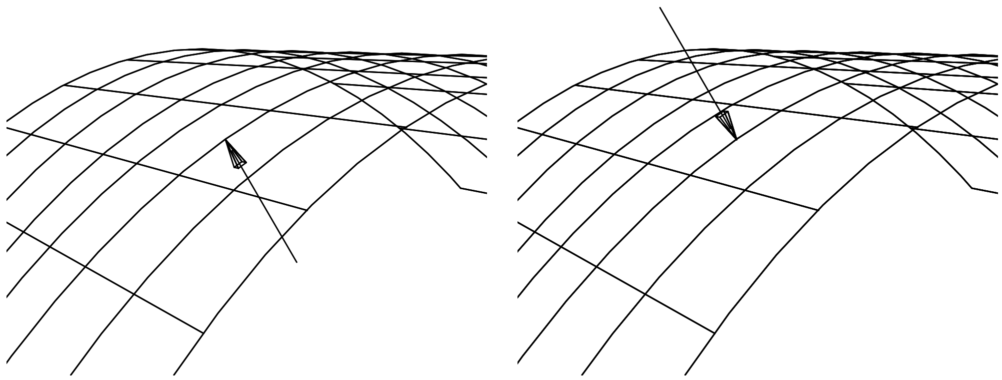


Figure 3.14. A gravity force tool visualized as a plane with an arrow in the center. The arrow points in the direction of the gravity force.

3.5.3 Pressure Forces

Pressure is applied force per unit area, and two different force effects may occur when applying *pressure* to a shape; increasing pressure, which is an inflation effect, and decreasing pressure, which is a deflation effect. These effects are applied to a surface by selecting either of the **Inflation force** or **Deflation force** buttons or pulldowns, and are best described as forces acting in the directions normal to the surface, such as a balloon being filled or emptied of air. Pressure force tools are visualized simply as an uneditable arrow pointing at the surface at a default surface location, as shown in Figure 3.15.



(a) (b)

Figure 3.15. The two pressure force tools of inflate and deflate are arrows pointing at the surface, showing the direction pressure is applied. (a) Inflation force tool is an arrow pointing at the surface. (b) Deflation force tool is an arrow pointing at the surface.

Since forces have magnitude, every force effect is given a default value that may be modified using a slider (see Figure 3.16). Like surface properties and constraints, this information is accessed through the **Properties** button or pulldown. A dialog box for showing force details, like the details for a point or tangent constraint, can be viewed by selecting the **Details** button. Figure 3.17 is an example of such details for a spring force. The XYZ world coordinates and UV surface parameter locations indicate the spring force attachment point. The Direction field shows the direction of the force. As with point and normal constraints, this information may not be modified. A details dialog does not exist for the pressure forces since the directional forces may vary over the surface.

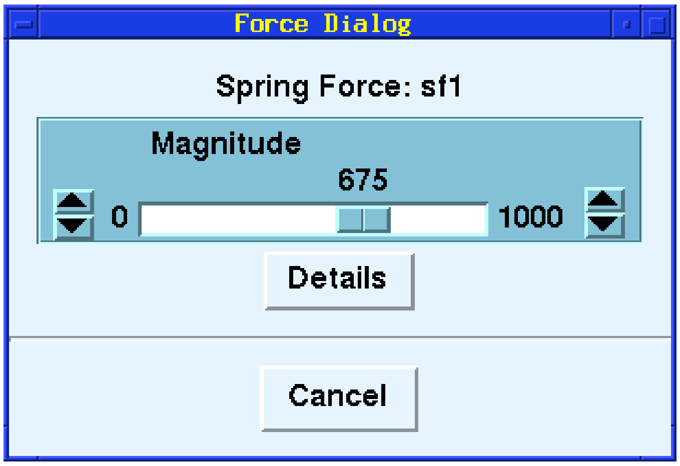


Figure 3.16. Force magnitude dialog box for a spring force. Force dialogs give the type and name of the force, and a slider for manipulating force magnitude. The Details button is used to get additional information about the force.

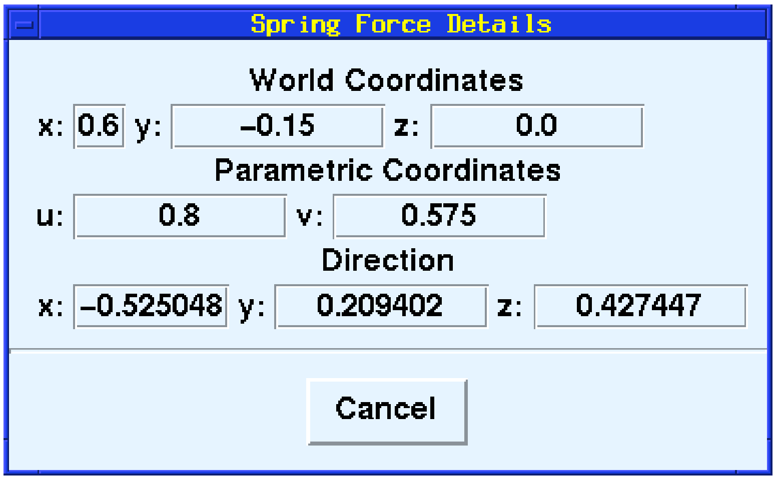


Figure 3.17. Details dialog for a spring force gives both world and parametric coordinates of the springs attachment point on the applied surface, and the direction of the force.

CHAPTER 4

IMPLEMENTATION

Manipulating B-spline shape parameters based on the physical conditions imposed by the user is not a simple task. Since this new interface operates at a higher level than concerning the user with low-level B-spline shape parameters, those details are being taken care of automatically. This chapter discusses the methodology used to create and modify tensor-product B-spline surfaces under physically-based conditions.

4.1 Tensor-Product B-spline Surfaces

The deformable models described in this research are tensor-product B-spline surfaces. The tensor-product B-spline surface are a supported shape representation in the Alpha_1 modeling system. As given in Section 2.4, a tensor-produce B-spline surface is:

$$s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} B_{i,k_u}(u) B_{j,k_v}(v), \tag{4.1}$$

with parametric values u and v, $(m+1)(n+1)$ control points $\{p_{i,j}\}$, and B-spline basis functions $B_{i,k_u}(u)$ and $B_{j,k_v}(v)$.

4.1.1 Surface Properties

Surface properties define the physical qualities of a surface, and describe how it should react under applied forces. The supported properties in this work include tension and rigidity also known as stretch and bend.

A useful equation to describe the elasticity of NURBS surfaces is derived in D-NURBS research [25, 30]. This equation allows for uniform or varied surface properties over the model, and can easily be applied to the simpler tensor-product B-spline surface. As given in Section 2.1.2, the stiffness equation is defined as:

$$\mathbf{K} = \iint (\alpha_{1,1}\mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2}\mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1}\mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2}\mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2}\mathbf{J}_{vv}^\top \mathbf{J}_{vv})\, du\, dv \quad (4.2)$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are functions controlling local stretching and bending of the surface. The Jacobian matrix $\mathbf{J}$ is derived for B-spline surfaces instead of NURBS surfaces (see Appendix A). $\mathbf{K}$ is an $N \times N$ matrix, where $N = (m+1)(n+1)$.

For small N, Gaussian quadrature [3] with Gauss-Legendre [23] calculated weights and abscissas was chosen to solve the integral of Equation 4.2. Gaussian quadrature is an appropriate choice for numerically integrating because of the polynomial nature of B-spline surfaces. More often N is large, and Gaussian quadrature is too time costly of a method. So, a direct method is used to solve for large $\mathbf{K}$'s instead. The computation of $\mathbf{K}$ can be separated into a sum of integrals as:

$$\begin{aligned} \mathbf{K} \;=\; & \alpha_{1,1}\iint \mathbf{J}_u^\top \mathbf{J}_u\, du\, dv + \alpha_{2,2}\iint \mathbf{J}_v^\top \mathbf{J}_v\, du\, dv + & (4.3)\\ & \beta_{1,1}\iint \mathbf{J}_{uu}^\top \mathbf{J}_{uu}\, du\, dv + \beta_{1,2}\iint \mathbf{J}_{uv}^\top \mathbf{J}_{uv}\, du\, dv + \beta_{2,2}\iint \mathbf{J}_{vv}^\top \mathbf{J}_{vv}\, du\, dv & (4.4) \end{aligned}$$

To exemplify this direct integration method, one element in the matrix produced by the integral of

$$\alpha_{1,1}\iint \mathbf{J}_u^\top \mathbf{J}_u\, du\, dv \quad (4.5)$$

looks like

$$\alpha_{1,1}\iint B'_{i,k_u}(u)B_{j,k_v}(v)B'_{r,k_u}(u)B_{s,k_v}(v)\, du\, dv, \quad (4.6)$$

which is the same as

$$\alpha_{1,1} \int B'_{i,k_u}(u) B'_{r,k_u}(u)\, du \int B_{j,k_v}(v) B_{s,k_v}(v)\, dv. \tag{4.7}$$

With a large uniform knot vector, the method takes advantage of the fact that all B-spline basis functions are translations of a single function, with the exception of the end cases. From Equation 4.7, the answer to the integral

$$\int B_{j,k_v}(v) B_{s,k_v}(v)\, dv \tag{4.8}$$

is one of a few possible choices that were computed exactly and put in a small look-up table. Similar look-up tables were computed for the first and second derivative functions.

4.2 Forces

Force effects are used to act upon and sculpt a surface's shape. As explained previously, effects in this environment include spring, gravity, inflation, and deflation forces.

The equation of motion from Section 2.1.2, contains a force element $\mathbf{f}$. This force is defined for NURBS [25, 30], and similarly in [6] for B-spline surfaces, and in [4] and [5] for finite-elements as

$$\mathbf{f} = \int\int \mathbf{J}^\top \mathbf{f}(u,v)\, du\, dv, \tag{4.9}$$

where $\mathbf{f}(u,v)$ is the force distribution vector. For the same reasons as with the stiffness matrix $\mathbf{K}$, Gaussian quadrature [3] is used to integrate the Equation 4.9.

With all user-applied surface properties and forces added to the model, the desired shape is the equilibrium shape. Equilibrium is achieved when user-specified forces balance with internal shape forces (i.e., surface properties) [4, 5, 6, 14], as in:

$$\mathbf{Kp} = \mathbf{f}.$$

The degrees of freedom in the vector $\mathbf{p}$ represent the final shape. Animating the surface is not included in this research, therefore the mass and damping effects from the equation of motion [6, 25, 28, 29, 30] are not needed.

4.3 Constraints

Point, curve, surface normal, and surface area constraints are represented as linear geometric functions of the B-spline representation's degrees of freedom; linear constraints can be can be solved quickly.

Consider the set of linear equations for equilibrium

$$\mathbf{K}\mathbf{p} = \mathbf{f},$$

where no constraints have been imposed on the system. When constraints are added, the equation becomes more complex.

Linear constraints are expressed as

$$\mathbf{A}\mathbf{p} = \mathbf{b},$$

where the N-vector $\mathbf{p}$ represents the degrees of freedom, $\mathbf{A}$ is an $M \times N$ with $M < N$, matrix of coefficients whose M rows each represent a linear constraint on $\mathbf{p}$, and the M-vector $\mathbf{b}$ represents the corresponding constraint values on $\mathbf{p}$. For example, consider the following case:

$$\begin{array}{cccc} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} & = & \begin{bmatrix} 5 \\ 5 \end{bmatrix} \\ \mathbf{A} & \mathbf{p} & = & \mathbf{b}, \end{array} \tag{4.10}$$

where ($M = 2$) constraints ($x_1 = 5$ and $y_1 = 5$) are expressed. Enforcing the constraints on the system is done by reducing Equation 4.10 to an unconstrained system of equations with minimal degrees of freedom. With the solution $\mathbf{y_0}$ given, the constraints can be written as a *general linear equation* as

$$\mathbf{p} = \mathbf{G}\mathbf{y} + \mathbf{y_0}, \tag{4.11}$$

where $\mathbf{y}$ is a Z-vector representing the reduced set of unconstrained degrees of freedom ($Z = N - M$), and the columns of the $N \times Z$ matrix $\mathbf{G}$ span the null space of $\mathbf{A}$. $\mathbf{G}$ and $\mathbf{y_0}$ are computed using the robust linear least-squares solver of singular value decomposition (SVD) [15].

Substituting Equation 4.11 into the system equilibrium equation $\mathbf{K}\mathbf{p} = \mathbf{f}$, yields:

$$\mathbf{K}(\mathbf{G}\mathbf{y} + \mathbf{y_0}) = \mathbf{f}$$

$$\mathbf{K}\mathbf{G}\mathbf{y} + \mathbf{K}\mathbf{y_0} = \mathbf{f}$$

$$\mathbf{K}\mathbf{G}\mathbf{y} = \mathbf{f} - \mathbf{K}\mathbf{y_0}$$

$$\mathbf{G}^\top\mathbf{K}\mathbf{G}\,\mathbf{y} = \mathbf{G}^\top\mathbf{f} - \mathbf{G}^\top\mathbf{K}\mathbf{y_0},$$

which is re-written using simpler terms [25, 30] as

$$\mathbf{K_y}\mathbf{y} = \mathbf{f_y} + \mathbf{g_y}.$$

This equation is the final equilibrium equation supporting the proposed physically-based modeling environment. A solution for $\mathbf{y}$ is calculated using the conjugate gradient method [15], and then inserted back into the *general linear equation* to solve for $\mathbf{p}$. The conjugate gradient method is used because it is an iterative method, ideal for solving large, sparse, symmetric, positive definite linear systems. Keep in mind that each linear constraint is calculated in a different way; therefore the number of rows in the constraint matrix $\mathbf{A}$ and vector $\mathbf{b}$ will vary, as of course will their content.

4.3.1 Point Constraints

Fixing a point on the surface at parametric location (u^0, v^0) requires the following constraint equation

$$s(u^0, v^0) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} B_{i,k_u}(u^0) B_{j,k_v}(v^0),$$

which adds one constraint row into the **A** constraint matrix.

4.3.2 Normal Constraints

The surface normal at a particular (u^0, v^0) parametric location on a tensor-product B-spline surface $s(u, v)$ is calculated as the cross product of any two surface tangent vectors at that point. Thus constraining these two tangent vectors is equivalent to constraining the surface normal at a given (u^0, v^0). In this work, the two tangent vectors t_u and t_v are calculated as the tangents to the surface in the u and v directions. Thus, the surface normal n is

$$n = |t_u \times t_v|.$$

The two equations to constrain are the following tangent functions defining $t_u(u^0, v^0)$ and $t_v(u^0, v^0)$:

$$t_u(u^0, v^0) = \frac{\partial s(u^0, v^0)}{\partial u} = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} \frac{dB_{i,k_u}(u^0)}{du} B_{j,k_v}(v^0)$$

$$t_v(u_0, v_0) = \frac{\partial s(u^0, v^0)}{\partial v} = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} B_{i,k_u}(u^0) \frac{dB_{j,k_v}(v^0)}{dv}.$$

These two functions add two constrained rows into the **A** matrix, one for t_u and one for t_v.

4.3.3 Parameter Curve Constraints

Constraining a fixed parameter curve requires slightly more computation than a point constraint, but the equations are more simple. The constraint function to compute for fixing a parameter curve depends on whether the curve is in the u or v direction of the B-spline surface. A parameter curve in the u (row) direction has a particular constant v^0 parametric value, while a curve in the v (column) direction has a constant u^0 value (see Figure 3.8). The constraint equations for column $c(u^0)$ and row $c(v^0)$ parametric curve constraints are given by

$$
\begin{aligned}
c(u^0) &= \sum_{j=0}^{m} p_{i,j} B_j(u_0), \quad \text{for each } i = 0..n \\
c(v^0) &= \sum_{i=0}^{n} p_{i,j} B_j(v_0), \quad \text{for each } j = 0..m
\end{aligned}
$$

where $m + 1 =$ the number of control points in the u direction, and $n + 1 =$ the number of control points in the v direction. A row constraint adds $(m+1)$ constraint rows to the **A** matrix, while a column constraint adds $(n + 1)$.

4.3.4 Arbitrary Curve Constraints

A freehand drawn curve constraint on a surface is more complex than any of the point, normal, or parametric curve constraints. This section provides the necessary equations for constraining a curve, but for the derivation or a more detailed explanation, please see [6].

When an arbitrary curve constraint is drawn on the surface, the curve is stored as a list of surface (u, v) locations, so let $t(s) = [u(s) \;\; v(s)]$ be the parametric curve in the surface. To shorten further notation, let

$$N_{i,j}(t(s)) = B_{i,k_u}(u(s)) B_{j,k_v}(v(s)).$$

The 3D shape of the curve in the surface is given by

$$c(s) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} N_{i,j}(t(s)).$$

The linear equation to solve for constraining a curve is

$$C_{k,l} = \int_{curve} \left(\sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j} N_{i,j}(t(s)) \right) N_{k,l}(t(s)) ds,$$

where $C_{k,l}$ denotes each of the rows of matrix **A** imposing a constraint on the $(k,l)^{th}$ control point of the surface. A curve constraint adds (m+1)(n+1) constraints to the system.

4.3.5 Surface Areas

Even though a surface area is created much like an arbitrary curve, the constraint methods are different. If the surface area is first constrained like a curve, (m+1)(n+1) constraints are added to the system, yet the region outside of the surface area still needs to be fixed. Doing so will add even more constraints to the system. As a result, a method has been developed here for a surface area to contribute no more than (m+1)(n+1) constraints.

The idea behind fixing a surface area is to constrain all of the control points influencing the outer region. This system constrains an outer surface area using steps:

- Get the bounding box of the surface area, (u_{min}, v_{min}) and (u_{max}, v_{max}) (Figure 4.1).

- Compute the control point ranges for which the basis functions at the bounding box's parametric values are non-zero: $[cp_{min}(u_{min}) \ldots cp_{max}(u_{min})]$, $[cp_{min}(v_{min}) \ldots cp_{max}(v_{min})]$, $[cp_{min}(u_{max}) \ldots cp_{max}(u_{max})]$, and $[cp_{min}(v_{max}) \ldots cp_{max}(v_{max})]$.

- Constrain all the control points in the v parametric direction for which the indices in the u direction are in the range $[0..cp_{max}(u_{min})]$ (Figure 4.2).

- Constrain all the control points in the v parametric direction for which the indices in the u direction are in the range $[cp_{min}(u_{max})..m]$, where $m+1 =$ the number of control points in the u direction (Figure 4.2).

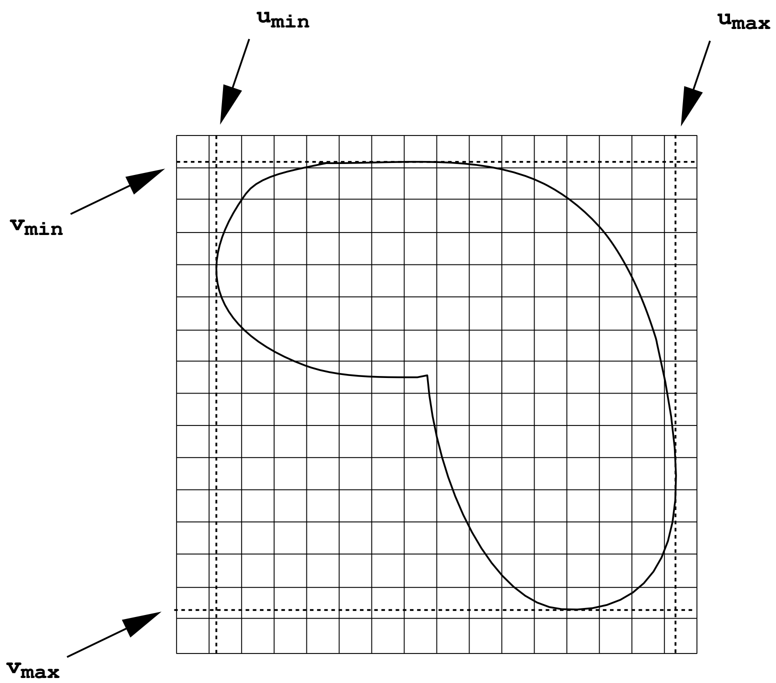


Figure 4.1. A surface area constraint drawn on a flat surface. The bounding box of the surface area is also shown.

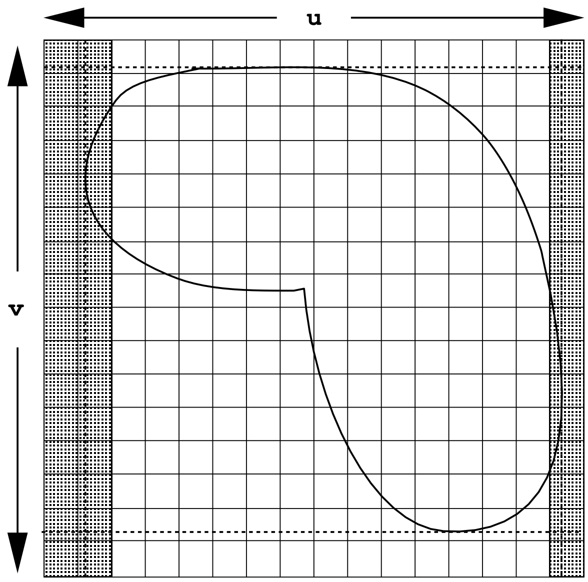


Figure 4.2. The support for the greyed out regions are constrained because they represent “outside” the surface area.

- Constrain all the control points in the u parametric direction for which the indices in the v direction are in the range $[0..cp_{max}(v_{min})]$ that do not overlap the control points from the previous steps (Figure 4.3).

- Constrain all the control points in the u parametric direction for which the indices in the v direction are in the range $[cp_{max}(v_{max})..n]$ that do not overlap the control points from the previous steps and where $n + 1 =$ the number of control points in the v direction (Figure 4.3).

- For the remaining control points p_i the following steps are taken (Figure 4.4):

  - Map p_i to its parametric surface point (u, v).
  - If (u, v) is on or outside the surface area, then constrain p_i.
  - Else (i.e., for (u, v) inside the surface area):

    * Get the control point ranges for which the basis functions at (u, v) are non-zero.

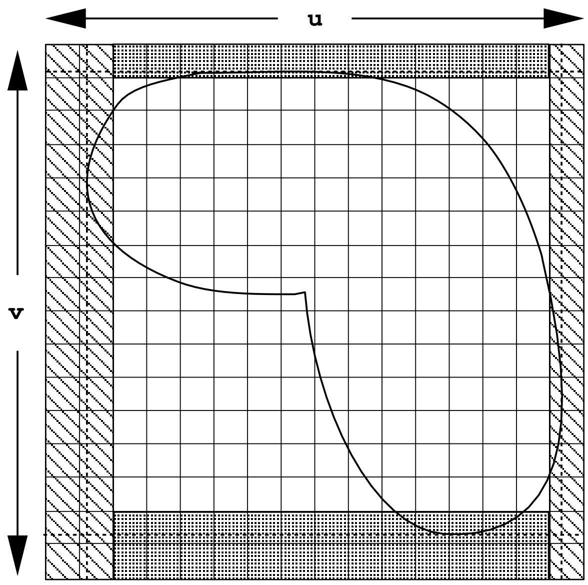


Figure 4.3. Not overlapping previously constrained regions, the support for the greyed out regions are constrained. These regions also represent "outside" the surface area.

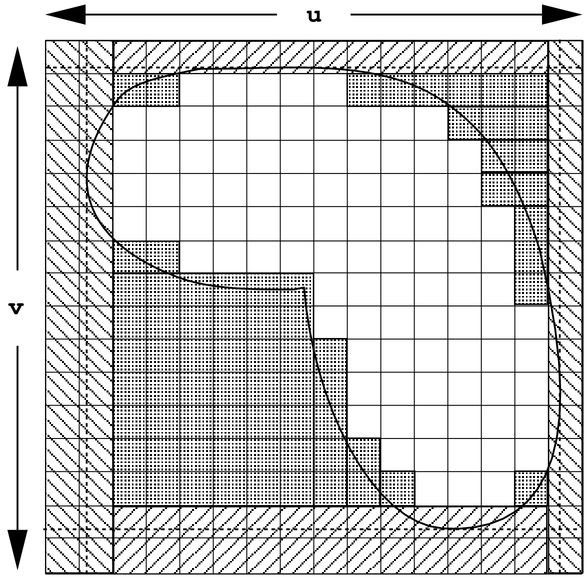


Figure 4.4. With all of the support constrained outside of the surface area's bounding box, still more "outside" regions are constrained.

- ∗ Map the four bounding control points to their parametric surface points (nodal).
- ∗ If any of the four surface points are on or outside the surface area, then constrain p_i.

4.4 Refinement

Refining B-spline surfaces offers finer control, but the problem arises of how a surface should be refined to give desired results. Some B-spline sculpting research does not focus on the issue of B-spline refinement [6, 17, 25, 30], others let the user define how the surface is refined [7, 11], and other research supports automatic refinement [31, 33].

Editing bending and stretching properties alone can change the surface shape. In other systems, when finer control is needed or desired, the user is responsible for adding more degrees of freedom until they have enough control to reflect the desired properties. In this work, the idea is for a surface, when given specified stretch and bend property values, to be automatically refined such that sculpting the surface will give results reflecting those properties.

As discussed in Section 2.4.1, the refinement process in the Alpha_1 system accepts refined surface knot vectors as input, where new vector values are added and no existing knot values may be modified. Returned from refinement methods are a new surface with the same shape as the unrefined version, but with more control points and containing the refined knot vectors. The new control points are automatically generated in the underlying refinement procedures. With these refinement guidelines in mind, refining the surface to have the proper degrees of freedom for desired surface behavior is dependent upon the additional knot vector values being inserted into the current representation.

With constant $\alpha_{i,j} = tension(stretch)$ and $\beta_{i,j} = rigidity(bend)$ functions over the surface being sculpted, the surface should behave the same at all geometric locations. Therefore, a surface with uniform parameterization will presumably have consistent behavior. Since the process for refining a surface only offers control over its new knot vector sequences, this research aims to refine these knot vectors, and thus the surface, to achieve uniformity.

Requiring the input surface knot vectors to be uniform allows for uniform refinement based on a computed heuristic. Consider that a highly refined surface is needed if the user wants lots of flexibility, while the opposite applies for more rigid surface properties. Therefore, more flexibility encourages a small heuristic, and less flexibility means a large heuristic. Considering knot vectors with parametric range between zero and one, the equation

$$heur = \left(\frac{\alpha^2+\beta^2}{{\alpha_{max}}^2+{\beta_{max}}^2}\right)^{k-1}, \quad 0.01 < heur < 1, \tag{4.12}$$

produces a heuristic value between zero and one, where α_{max} and β_{max} are the maximum values for stretching and bending parameters, and k is the order of the surface in the parametric direction being refined. Since a heuristic of zero is too small, and will cause multiple knots to be formed, the heuristic in Equation 4.12 is bounded between 0.01 and 1. If the parametric range of a given knot vector is not between 0 and 1, *heur* must be scaled by the size of the range.

One more consideration must be taken when calculating the heuristic in Equation 4.12, which are the dimensions of the surface in the u and v directions. For instance, if the surface in the u direction is twice the size of the surface in the v direction, the knots could be twice as far apart in the u direction of the surface. For this reason, the *heur* for each knot vector must be scaled down by (smaller dimension)/(larger dimension) to achieve more uniformly spaced knots.

4.5 Collisions

Collision detection or avoidance is a way of letting changing objects in a scene interact with each other. In this research, the changing object is the surface being sculpted. Any other objects for the surface to notice are various sized sphere obstacles that the user can locate in the scene.

Since the presented physically-based system does not solve for the equilibrium of the surface over time as in [25, 28, 29, 30, 31], the system is unable to detect when or where a collision is taking place. Therefore, collision detection occurs between the sculpting surface and obstacles in the scene when the surface *reaches* its computed shape. If a collision occurs, the surface then alters its shape to avoid intersecting with the obstacles.

Described here is a method for avoiding collisions that reduces the forces on the system until obstacles and the surface no longer intersect.

```
percent := 10
Do
   intersection := FALSE
   For each obstacle
      if the surface and obstacle intersect && percent <= 100
         decrease forces by percent%
         calculate surface sculpted with new forces
         percent := percent + 10
```

```
        intersection := TRUE
While intersection == TRUE || percent <= 100
```

As a result of the above algorithm, the surface will no longer change shape if the forces are all zero, even if the surface is still intersecting obstacles.

4.6 Optimizations

The implementation of the above physically-based methods lends itself to some interesting optimizations. Using the various design tools impacts what needs to be recomputed, and therefore large parts of the computation can be reused. The major optimization techniques are noted here.

With surface properties, forces, constraints, and optional obstacles applied to a surface, the sculpted surface is calculated according to the previous sections, which are summarized in the following steps:

1. Refine the surface according to its material properties to get a new surface. The new surface is used in the remaining steps.
2. Create the following for the surface:
   - force vector $\mathbf{f}$,
   - stiffness matrix $\mathbf{K}$,
   - constraint matrix $\mathbf{A}$, its associated vector $\mathbf{b}$, the nullspace $\mathbf{G}$ of $\mathbf{A}$, and the solution $\mathbf{y_0}$ to $\mathbf{A}\mathbf{y_0} = \mathbf{b}$.
3. Calculate $\mathbf{y}$ for $\mathbf{G}^\top\mathbf{K}\mathbf{G}\,\mathbf{y} = \mathbf{G}^\top\mathbf{f} - \mathbf{G}^\top\mathbf{K}\mathbf{y_0}$, or ($\mathbf{K_y}\mathbf{y} = \mathbf{f_y} + \mathbf{g_y}$.).
4. Compute the result $\mathbf{p}$ from $\mathbf{p} = \mathbf{G}\mathbf{y} + \mathbf{y_0}$.

With solution $\mathbf{p}$, the new sculpted surface is computed and visualized on the screen for the user. If the user changes surface properties, then the refinement needs to be re-calculated. Therefore, steps 1-4 are followed, recomputing all vectors and matrices for a new answer. Unfortunately, no computations can be reused when surface properties are modified. Luckily, changing forces, constraints, and obstacles always lead to the optimization of reusing previous computations.

Editing constraints for a surface being sculpted does not change material properties or forces, therefore refining the surface is not necessary, and the $\mathbf{K}$ stiffness matrix and $\mathbf{f}$ force vector remain intact. As a result, only the vectors and matrices for constraints $(\mathbf{A}, \mathbf{b}, \mathbf{G}, \mathbf{y_0})$ need recalculating, followed by steps 3 and 4.

The most optimization occurs when forces and obstacles are edited. As with constraints, material properties do not change, therefore the surface is not refined, and the $\mathbf{K}$ stiffness matrix is the same as before. In addition, constraints are also left unchanged, and thus computing $\mathbf{A}$, $\mathbf{b}$, $\mathbf{G}$, and $\mathbf{y_0}$ is unnecessary. For further optimizations, the matrices $\mathbf{G}^\top$ and $\mathbf{K_y}$, and vector $\mathbf{g_y}$ are also previously stored to avoid additional computations. As a result, only minimal calculations are required in step 3, followed by a complete step 4.

CHAPTER 5

RESULTS

This sections describes the results of various test cases that demonstrate the geometric constraints and the physically-based design capabilities described in the previous chapters. A complex surface is created using a variety of tools, the difference between force effects are shown, surface property effects are differentiated, surface area constraints are demonstrated, and evaluation of the implementation is given.

5.1 Sculpting

Figure 5.1 shows a simple, flat rectangle. Three point constraints, a freehand curve constraint, a parametric curve constraint, and a gravity force effect are applied to this surface. When the user is ready for sculpting to occur, the surface in Figure 5.2 results. Nearly instantaneous updates are obtained when adjusting the force location, magnitude, or direction, or editing any of the constraints. The time for computing surface property manipulation is discussed below. Figure 5.3 shows the same surface as 5.2, but at a different angle to visualize obscured results.

5.2 Force Effects

Each force effects provides a unique tool for modifying the surface. These forces offer either localized or global control over shape changes. Figure 5.4 shows how gravity has global control over changing the surface, while springs are a local deformation. The spring is pulling the center of a flat surface in one direction, while gravity is pulling the entire surface in the opposite direction. Pressure is also a global shape modifier. Figure 5.5 demonstrates the gravity effect on the complex surface of Figure 5.2, and Figure 5.6 shows the same surface under an

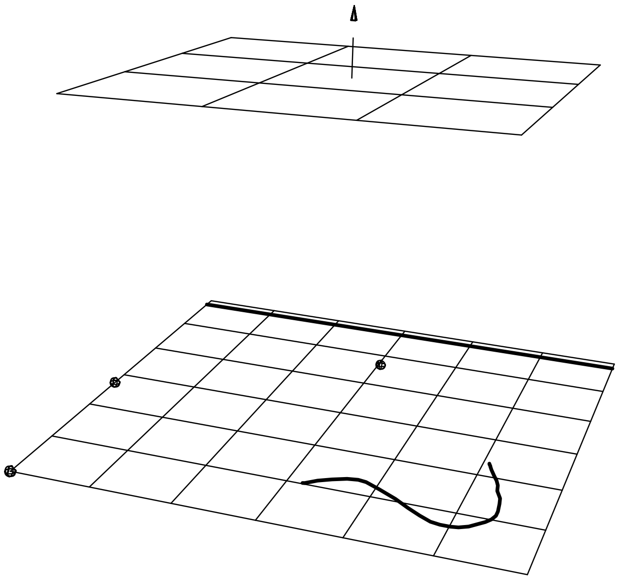


Figure 5.1. Three point constraints, a freehand curve constrain, a parametric curve constraint, and a gravity force are applied to a flat surface before sculpting.

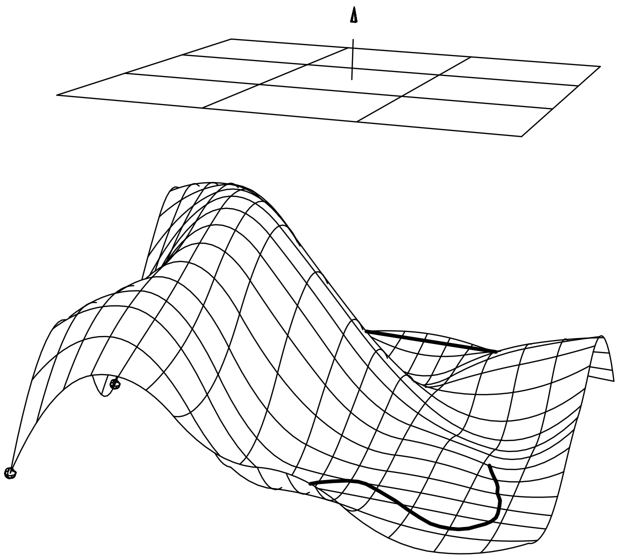


Figure 5.2. The flat surface in Figure 5.1 is sculpted to give a surface that is pulled in the direction of gravity and maintains its constraints.

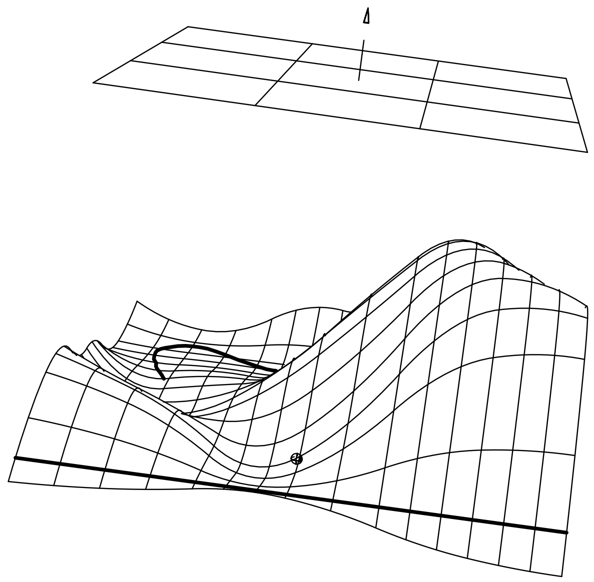


Figure 5.3. Another perspective of the sculpted surface in Figure 5.2.

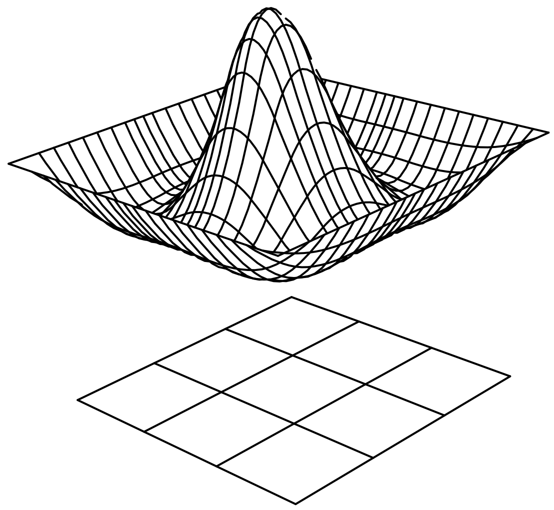


Figure 5.4. A spring force pulls up on a flat surface showing local control, and a gravity force pulls in the opposite direction showing global control.

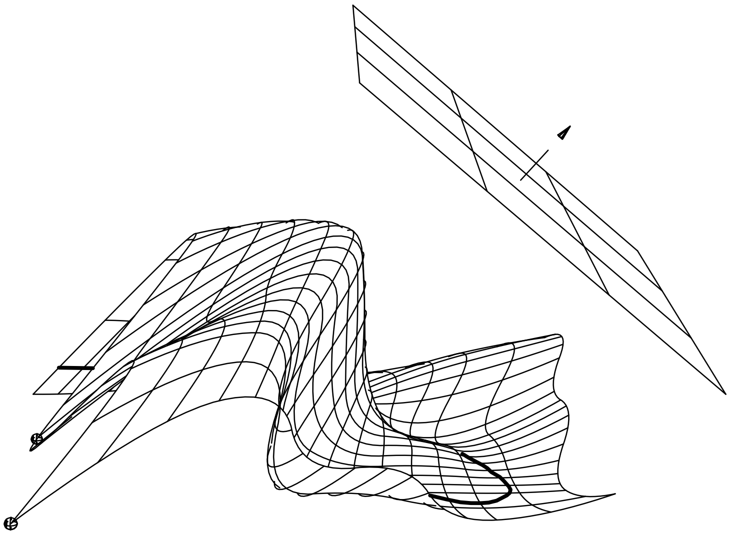


Figure 5.5. A gravity force sculpts the wavy surface in Figure 5.2 to achieve a new surface shape.

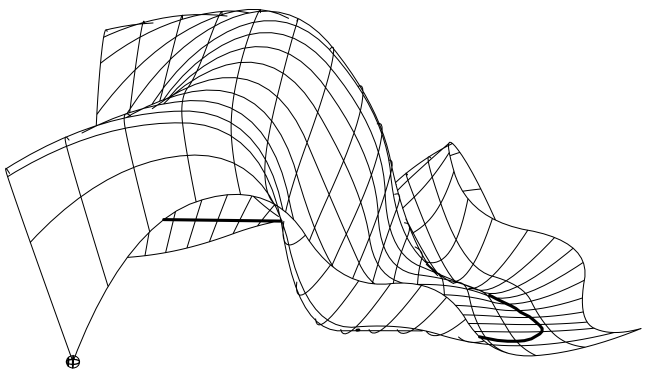


Figure 5.6. An inflation pressure force sculpts the surface in Figure 5.2 to result in more varied results.

inflation pressure force. The pressure force effect is fixed to act in the direction of the surface normal, while gravity is free to pull in any direction. These two examples also show how a sculpted surface can be used as the surface to sculpt again. Compositing sculpting operation is an effective means to create a surface full of features and detail.

5.3 Surface Properties and Automatic Refinement

The bend and stretch surface properties, aided with automatic surface refinement are provided in this system as tools for gaining additional control over the shape of a surface being sculpted. Figure 5.7 shows a flat 3x3 mesh surface sculpted with minimum stretch and maximum bend without automatic refinement. The surface is constrained by two isoparametric curves deformed by a single spring force. Figure 5.8 shows the same surface, but using automatic refinement. Notice the extra detail near the parametric curve constraints (the edges), and that the peak has a sharper bend in the refined image.

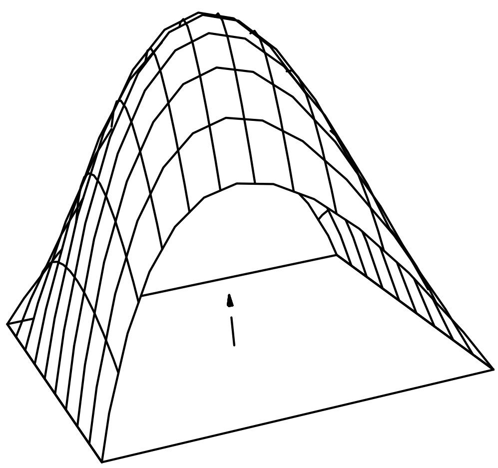


Figure 5.7. A flat 3x3 mesh surface sculpted with minimum stretch and maximum bend without automatic refinement. The surface is constrained by two isoparametric curves, and deformed by a spring force.

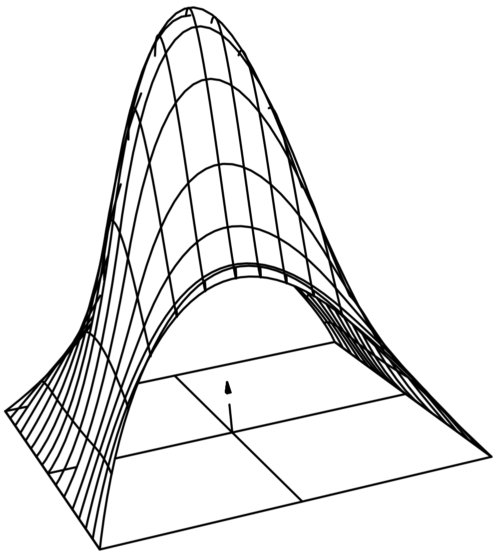


Figure 5.8. A flat 3x3 mesh surface automatically refined to 6x6 and sculpted with minimum stretch and maximum bend. As in the previous figure, the surface is constrained by two isoparametric curves, and deformed by a spring force.

While bending and stretching properties provide some control over the surface, a surface will not sculpt in a predictable way if sufficient constraints are not supplied. For example, a surface with only one or two point constraints will collapse to unexpected results. This behavior is due to the nature of the energy minimization algorithm governing the surface properties. So the single point constraint case collapses to a line segment.

5.4 Surface Area Constraints

Surface area constraints allow the designer to focus on sculpting a region of the surface without indirectly effecting other areas of the surface. The previous sculpting methods can be performed on a surface area. Figure 5.9 is a flat rectangle with a freehand surface area constraint and three spring forces all pointing towards the same point in space. The result of this sculpting setup is shown in Figure 5.10.

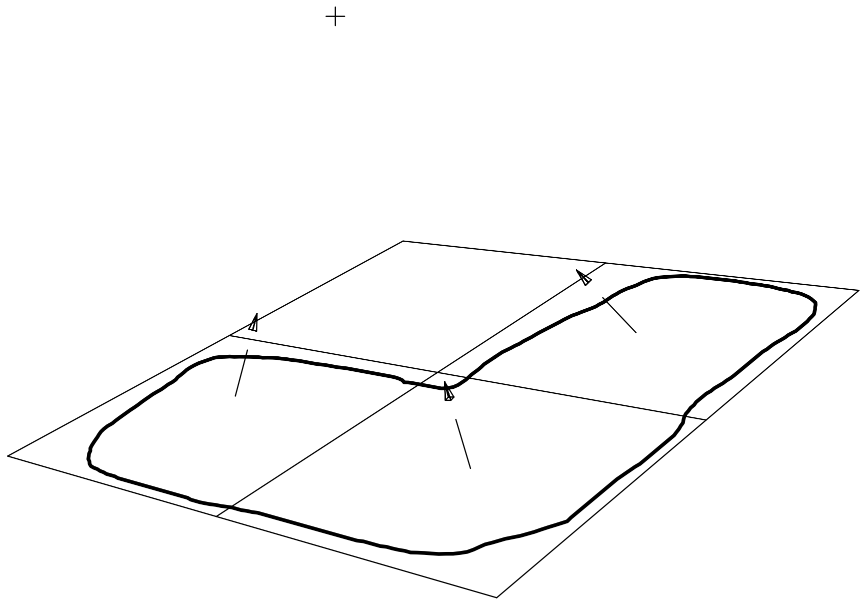


Figure 5.9. A freehand surface area and three spring forces are applied to a flat surface. All of the springs are directed at the same point in space.

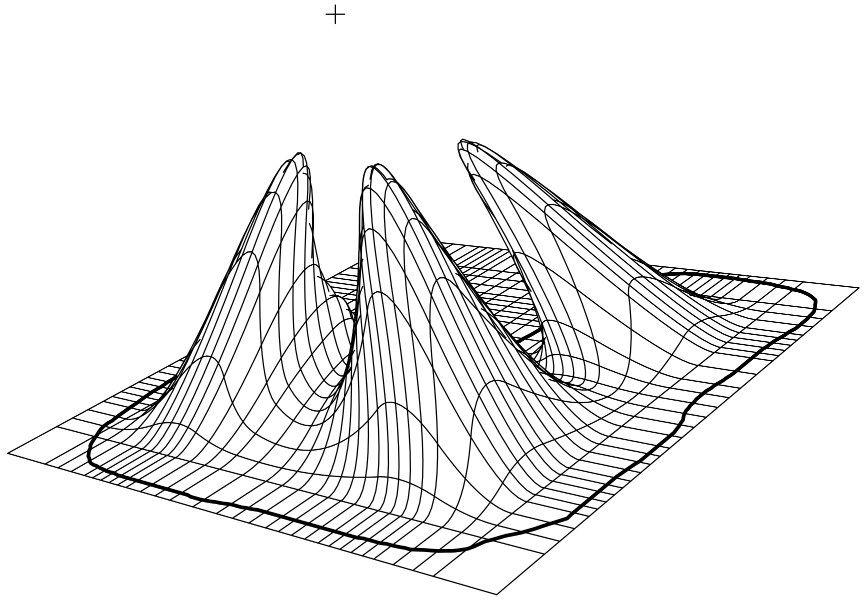


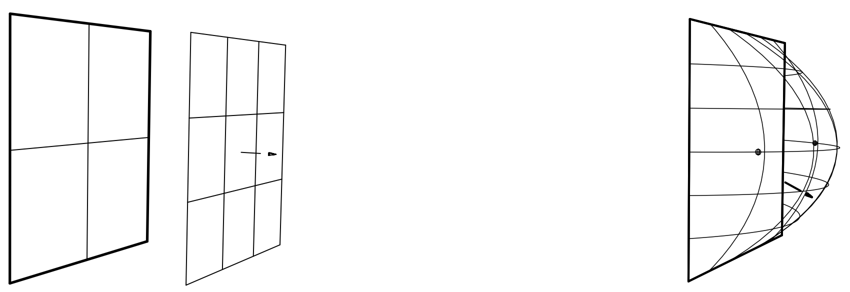
Figure 5.10. The surface in Figure 5.9 is sculpted according to the applied design tools of three spring forces, constrained by a freehand drawn surface area.

5.5 Example: Sculpting a Face

To show the usefulness of the sculpting environment, and thus this research, we describe the process of sculpting a flat surface into the shape of a face. Figure 5.11(a) shows the flat surface to be sculpted on the left. The surface boundaries are constrained with four parametric curve constraints, which are bolded in the picture. A gravity force is applied to the surface for sculpting. The result of this sculpting operation is the curved surface shown in Figure 5.11(b).

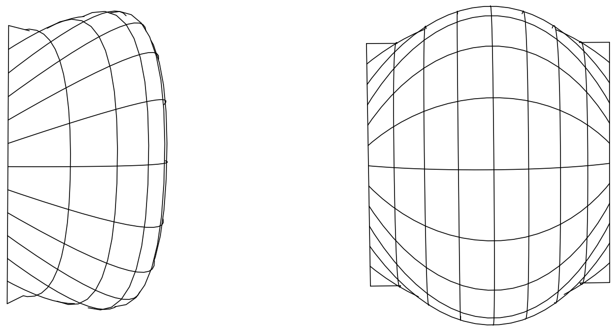
The next step requires sculpting the resulting shape. Two point constraints are applied to the left and right sides of Figure 5.11(b), as well as four parametric curve constraints along the surface's boundaries. For sculpting this shape, an inflation force is applied. Figure 5.12 shows two angles of the resulting sculpted surface. Notice the two sides with point constraints are not bulging like the top and bottom areas.

Now that we have the general shape of a face, details of a face need to be added. Figure 5.13 shows the application of a surface area in the front center of the surface. This surface area was applied at a skewed angle using the circle surface area tool, and thus having an oval shaped area. Two forces are applied to sculpt this area, a



(a) (b)

Figure 5.11. (a) A flat surface constrained by its boundaries is sculpted with a gravity force. (b) The newly sculpted surface is constrained at its boundaries and with two points on either side. A pressure force is then applied.



(a) (b)

Figure 5.12. The facial shape resulting from the composite gravity and pressure sculpting operations in Figure 5.11. (a) Side view. (b) Front view.

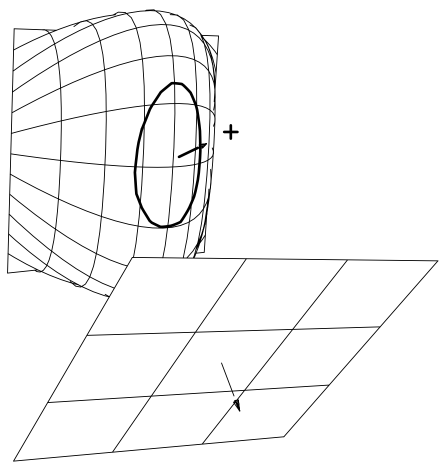
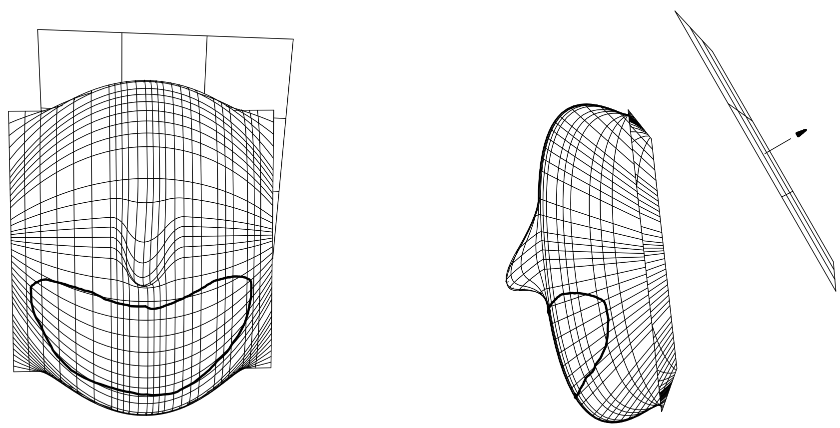


Figure 5.13. A surface area constraint is applied to the surface. A small spring force points up from the center of the surface area, and a large gravity force will pull diagonally down in front.

large gravity force for pulling the area diagonally down in front, and a spring force to give a smaller section of the surface a little lift. The resulting sculpted nose is shown in Figure 5.14.

As seen in the sculpted surface of Figure 5.14, a mouth is the next feature this face will receive. This figure shows a freehand drawn surface area, with a gravity force applied to the other side of the surface. Two views are given to demonstrate the shape of the nose, as well as the direction of the gravity force. The face, supplied with a nose and mouth is given in Figure 5.15.

This face is now due to have an eye or two. As shown in Figure 5.15, a surface area is drawn around the region to receive an eye feature. For sculpting, a gravity force is applied to the back. The result in Figure 5.16 is given another surface area for an eye, as well as a gravity force to sculpt this last eye. The final face, containing a mouth, a nose and two eyes, is shown in Figure 5.17 and 5.18.



(a) (b)

Figure 5.14. A nose was created as the result of the sculpting in Figure 5.13. (a) To create a mouth, a freehand drawn surface area constraint is applied to the surface, and a gravity force points towards the back side of the face. (b) A side view of the new nose, the drawn surface area constraint, and the gravity force.

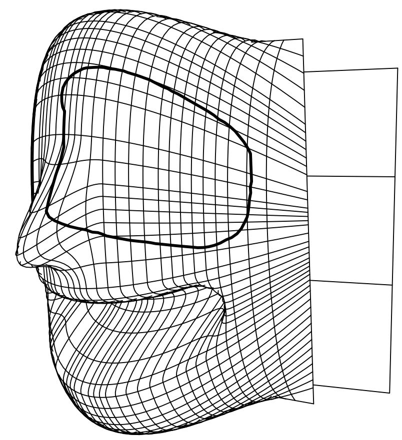


Figure 5.15. The face, which includes a nose and a newly sculpted mouth, is given a surface area constraint and a gravity force for creating an eye.

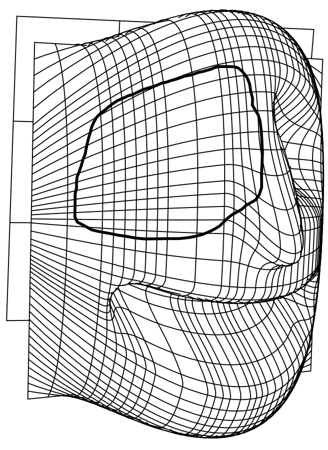
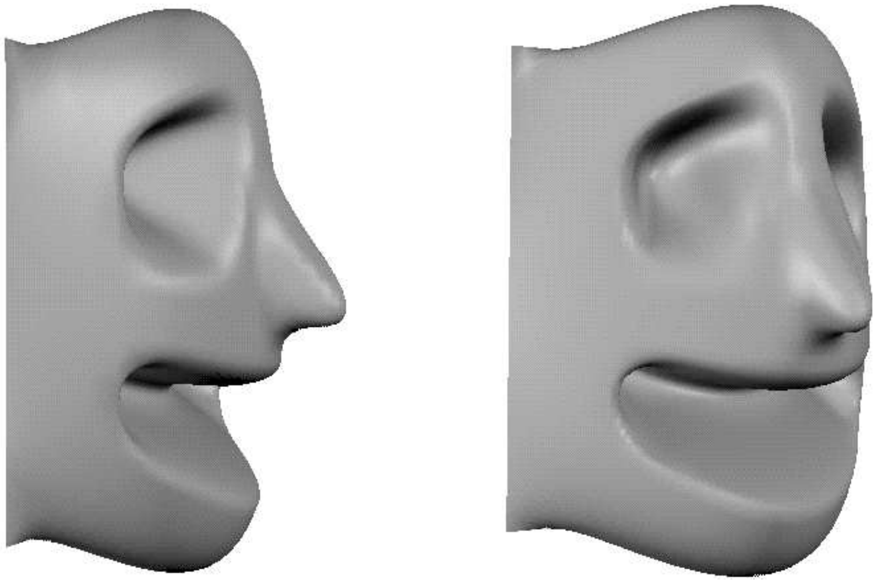


Figure 5.16. A face with one eye, a nose, and a mouth is the result of sculpting in Figure 5.15. One more surface area and gravity force is applied to achieve the second eye.



(a) (b)

Figure 5.17. Two side views of the face that was sculpted from a flat surface. (a) The profile mug shot. (b) Another perspective.

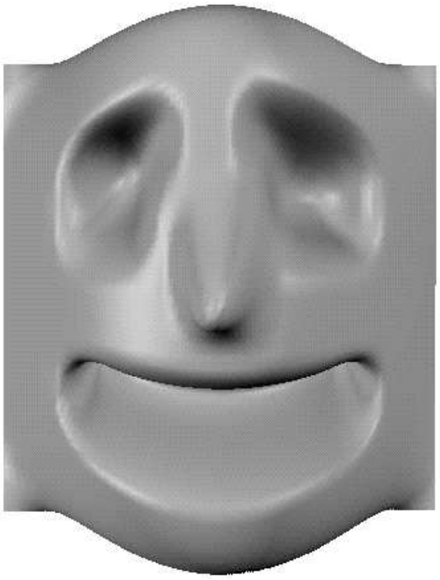


Figure 5.18. The official mug shot of the sculpted face.

5.6 Performance Evaluation

As discussed in Section 4.6, there are three performance critical algorithms in this system, specifically, solving for the forces, constraints, and stiffness of the surface. As noted before, forces are calculated with:

$$\mathbf{f} = \iint \mathbf{J}^\top \mathbf{f}(u,v)\, du\, dv,$$

linear constraints with:

$$\mathbf{Ap} = \mathbf{b},$$

which reduces to (see Section 4.3)

$$\mathbf{p} = \mathbf{Gy} + \mathbf{y_0},$$

for the unconstrained system, and stiffness, which governs stretching and bending as:

$$\mathbf{K} = \iint (\alpha_{1,1}\mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2}\mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1}\mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2}\mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2}\mathbf{J}_{vv}^\top \mathbf{J}_{vv})\, du\, dv.$$

These algorithms are used to solve the system:

$$\mathbf{G}^\top \mathbf{KG}\, \mathbf{y} = \mathbf{G}^\top \mathbf{f} - \mathbf{G}^\top \mathbf{K} \mathbf{y_0}.$$

The orders of the previous algorithms can be expressed in terms of the number of constraints, number of forces, orders of the surface, or the size of the control mesh. Since the size of the mesh determines the size of large integral equations for solving the system, the following results are reported in terms of the size of the $m \times n$ control point mesh. In the following analysis, the order of the SVD and conjugate gradient iterative solvers does not depend directly on the size of the control point mesh, but instead on the number of constraints. For a $r \times r$ matrix, with condition number κ, the conjugate gradient method takes on the order of $r^2\sqrt{\kappa}$, which is the number of iterations to converge. In the case of a refined surface with several hundred constraints, computing the SVD of the matrix **A** takes only a few seconds, and the conjugate gradient solver takes far less than a second.

Changing a force in this system requires computations from the least complex algorithm. The effect of editing a single pressure force requires on the order of $mn^2 + nm^2$ operations, while the effect of editing a spring or gravity force takes on the order of $m^2 + n^2$ operations. If any force is changed, computations of all forces acting on the surface are recalculated.

The effect from changing a constraint takes more time to compute than the effect from changing a force. Recomputing the equations to satisfy constraint changes with a curve constraint involved, takes on the order of m^2n^2 operations, while without a curve constraint, only takes on the order of mn operations. If any constraint is changed, the constraint matrix **A** is recomputed.

Changing the bending and stretching properties of the surface results in computations requiring minimal time. When a property is changed, the refinement heuristic is computed, the surface is automatically refined, and all of the above calculations take place. Solving the matrix **K** for new surface properties takes on the order of m^2n^2 operations.

In addition to time complexity, space complexity is also a factor when evaluating the performance of a system. In terms of space, K is an $mn \times mn$ matrix, using on the order of n^2m^2 bytes of memory. Therefore, if the size of the surface is large, for example 100×100, it takes 100,000,000 bytes of space. The bounds on automatically refining the surface is determined by the size of K.

CHAPTER 6

CONCLUSION

This thesis presented physically-based design tools for direct B-spline shape sculpting. Incorporating physically-based means of manipulating a shape solves the problem of tediously editing B-spline shape parameters. The physically-based tools provided include spring forces, for local control over the surface, as well as gravity and pressure forces for global control. A set of tools are offered for constraining specified points, curves, and normals on a surface. Surface area constraints were introduced as a new constraint mechanism to restrict movement outside of the specified area. Surface properties, specifically bend and stretch are also provided as additional aids for shape design. Static sphere obstacles can also be introduced to the system, for the surface being modeled to avoid.

To make use of all these shape sculpting tools, a modeling environment was incorporated into the existing Alpha_1 system. This environment offers buttons, sliders, menus, using the mouse and keyboard for interactive application and editing of the discussed force effects, constraints, surface properties, and obstacles. An attractive feature of this modeling system is that once a surface is sculpted it can then used as the basis for additional sculpting.

In typical physically-based systems, if the detail of the surface is not fine enough to achieve the desired results, the designer must manually refine the surface. This research presents a means of automatically refining the surface. Given stretch and bend values, surface parameter ranges, surface orders, and surface dimensions, a heuristic value for both parametric directions is computed as a guide for refinement.

While there are limitations in the current implementation, this system successfully demonstrates a set of useful physically-based design tools for interactive surface sculpting.

6.1 Future Work

The work presented in this thesis demonstrates an environment for physically-based sculpting, paving the way for additional features to be investigated. This section describes a number of features that would serve to enhance the current state of this research.

6.1.1 Extend Collision Capabilities

Although collision avoidance was implemented, additional collision capabilities were not explored. Currently, the obstacles to avoid are spheres. Additional types of objects, such as planes, tubing, or a user-defined arbitrary shape, might introduce more interesting sculpting results. As another enhancement, collision avoidance techniques could be developed to allow the object being designed, to stick to and wrap around the obstacles based on the surface's material properties. Therefore, forces over the entire surface would need not be reduced, and would only change in the regions of intersection. Future research for detecting and avoiding self-intersections could also be added.

6.1.2 Advanced Interface and Interactivity

The current implementation for surface areas and arbitrary curve constraints does not allow for their editing once drawn on the surface. One interactive alternative might map the curve's control polygon onto the surface. As the control polygon is edited, the curve shape will follow and remain on the surface along with the control polygon. Methods would need to be researched to develop such a control polygon to surface mapping.

Physically-based interaction would benefit from having more direct interaction with the surface being sculpted. The addition of 3D surface manipulation, using for example a data glove in an immersive or perhaps haptic environment, would offer more the sense of actual sculpting. 3D manipulation could also allow for simpler positioning of obstacles in the scene, since current editing is done with the mouse in a non-intuitive 2D framework.

Speed is an issue when dealing with design tools. The designer usually expects a quick reaction time when a command is executed. Even though many optimization techniques were implemented for more user-friendly design tools, real-time interaction was not a goal of this research. As a result, the current environment would benefit from further optimizations, such as coding sparse matrix computations into the Alpha_1 system.

6.1.3 Reduced Memory Use

Since the size of **K** grows with the square of the size of the surface mesh, **K** should be the first component of this system to optimize when considering memory usage. Taking advantage of the symmetry of **K** in terms of saving memory, or creating other methods that reduce the system to smaller, independently computed systems could be researched, for example, subdividing the surface into patches.

6.1.4 Smarter Refinement

Refining the surface for enough degrees of freedom to attain the desired effect during sculpting is a difficult problem. A more intelligent refinement method that focuses on adding detail to the areas being sculpted might be advantageous, instead of uniformly refining over the entire surface. This method would also allow for multiple surface areas drawn on the surface to have different material properties. Refining only portions of the surface could help speed up the modeling process in some design cases, due to the smaller size of the physical system to solve.

6.1.5 NURBS Extension

The tensor-product B-spline surface representation offers extensive physically-based functionality, as presented in this thesis. Although, this surface representation is a subset of NURBS (**N**on-Uniform **R**ational **B**-**S**pline) surfaces. Extending the physically-based design tools to work with NURBS surfaces would offer additional surface options for sculpting, such as spheres and surfaces of revolution. With NURBS, comes the weighting factor of the control points, which would require integration into the physics formulation as well as the automatic refinement process. Adding another degree of freedom might slow computations by expanding the size of the systems to be solved, and thus additional optimization methods would need to be considered.

6.2 Summary

This research realizes a physically-based B-spline sculpting environment with automatic control of low-level parameters. This environment combines force effects tools of springs, gravity, and pressure, linear constraints of points, curves, and normals, surface properties of stretch and bend, and sphere obstacle avoidance, into a single modeling framework. The nontrivial surfaces created in the Results section of this thesis, most noticeable is the sculpting of a face, were designed without directly editing B-spline parameters, demonstrating the validity of this interactive modeling approach. The author hopes this alternative to B-spline modeling interaction will become a staple in future generation modeling tools.

APPENDIX

JACOBIAN MATRIX DERIVATION

The Jacobian matrix is derived for B-spline surfaces. Given the parametric values u and v, the B-spline surface function $s(u,v)$ calculates the coordinates $x = s(u,v)_x$, $y = s(u,v)_y$, and $z = s(u,v)_z$.

Letting the vector

$$\mathbf{s} = \left[\begin{array}{ccc} s(u,v)_x & s(u,v)_y & s(u,v)_z \end{array} \right],$$

and if the $3(m+1)(n+1)$ generalized coordinate vector is

$$\mathbf{p} = \left[\begin{array}{ccc} p_{0,0}^{\top} & \cdots & p_{m,n}^{\top} \end{array} \right]^{\top} \text{where each } p_{i,j}^{\top} = \left[\begin{array}{ccc} x_{i,j} & y_{i,j} & z_{i,j} \end{array} \right],$$

then the Jacobian matrix of s with respect to the vector $\mathbf{p}$ is the matrix whose $(i,j)th$ entry is the result of $\partial s(i)/\partial \mathbf{p}(j)$, where

$$N_{i,j}(u,v) = \frac{\partial s_x}{\partial (p_{i,j})_x} = \frac{\partial s_y}{\partial (p_{i,j})_y} = \frac{\partial s_z}{\partial (p_{i,j})_z} = B_{i,k_u}(u) B_{j,k_v}(v).$$

The $1 \times (m+1)(n+1)$ Jacobian matrix $\mathbf{J} = \mathbf{J}(u,v)$, used to calculate the stiffness matrix $\mathbf{K}$, can now be written as

$$\mathbf{J} = \left[\begin{array}{ccc} N_{0,0}(u,v) & N_{0,1}(u,v) \cdots & N_{m,n}(u,v) \end{array} \right].$$

REFERENCES

[1] Barr, A. H. Global and local deformations of solid primitives. In *Computer Graphics (SIGGRAPH'84 proceedings)* (July 1984), vol. 18, pp. 21–30.

[2] Bartels, R., and Beatty, J. A technique for the direct manipulation of spline curves. In *Graphics Interface '89* (June 1989), pp. 33–89.

[3] Burden, R., and Faires, J. *Numerical Analysis.* PWS-Kent Publishing Company, 1993.

[4] Celniker, G. *ShapeWrite: Finite Element Based Free-Form Shape Design.* PhD thesis, Massachusetts Institute of Technology, 1990.

[5] Celniker, G., and Gossard, D. Deformable curve and surface finite-elements for free-form shape design. In *Computer Graphics (SIGGRAPH '91 proceedings)* (July 1991), vol. 25, pp. 257–266.

[6] Celniker, G., and Welch, W. Linear constraints for deformable b-spline surfaces. In *1992 Symposium on Interactive 3D Graphics* (1992), pp. 165–170.

[7] Cobb, E. *Design of Sculptures Surfaces Using B-splines.* PhD thesis, University of Utah, 1984.

[8] Coquillart, S. Extended free-form deformations: A sculpturing tool for 3d geometric modeling. In *Computer Graphics (SIGGRAPH '90 proceedings)* (August 1990), vol. 24, pp. 187–196.

[9] DataGlove is a trademark of Sun Microsystems, for more information, see the Sun web site: http://www.sun.com/.

[10] de Boor, C. *A Practical Guide to Splines.* Springer-Verlag, 1978.

[11] Forsey, D., and Bartels, R. Hierarchical b-spline refinement. In *Computer Graphics (SIGGRAPH '88 proceedings)* (August 1988), vol. 22, pp. 205–212.

[12] Fowler, B. Geometric manipulation of tensor product surfaces. In *1992 Symposium on Interactive 3D Graphics* (1992), pp. 101–108.

[13] Fung, Y. *Foundations of Solid Mechanics.* Prentice-Hall Inc., 1977.

[14] Giancoli, D. *Physics for Scientists and Engineers.* Prentice Hall, 1988.

[15] Golub, G., and Loan, C. V. *Matrix Computations.* The John Hopkins

University Press, 1996.

[16] Gortler, S., and Cohen, M. Hierarchical and variational geometric modeling with wavelets. In *1995 Symposium on Interactive 3D Graphics* (April 1995), pp. 35–42.

[17] Greiner, G., and Loos, J. Data dependent thin plate energy and its use in interactive surface modeling. In *Eurographics '96* (1996), vol. 15, pp. C–176–C–185.

[18] Hanrahan, P., and Haeberli, P. Direct wysiwyg painting and texturing on 3d shapes. In *Computer Graphics (SIGGRAPH '90 proceedings)* (August 1990), vol. 24, pp. 215–223.

[19] Herzen, B., Barr, A., and Zatz, H. Geometric collisions for time-dependent parametric surfaces. In *Computer Graphics (SIGGRAPH '90 proceedings)* (August 1990), vol. 24, pp. 39–48.

[20] Hsu, W., Hughes, J., and Kaufman, H. Direct manipulation of free-form deformations. In *Computer Graphics (SIGGRAPH '92 proceedings)* (July 1992), vol. 26, pp. 177–184.

[21] 3Space IsoTrak is a trademark of Polhemus Navigation Sciences, for more information, see the Polhemus web site: http://www.polhemus.com/.

[22] Platt, J., and Barr, A. Constraint methods for flexible models. In *Computer Graphics (SIGGRAPH '88 proceedings)* (August 1988), vol. 22, pp. 279–288.

[23] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. *Numerical Recipes in C.* Cambridge University Press, 1992.

[24] Promayon, E., Baconnier, P., and Puech, C. Physically-based deformations constrained in displacements and volume. In *Eurographics '96* (1996), vol. 15, pp. C–155–C–162.

[25] Qin, H. *D-NURBS: Dynamic Non-Uniform Rational B-Splines.* PhD thesis, University of Toronto, 1995.

[26] Sederberg, T., and Parry, S. Free-form deformations of solid geometric models. In *Computer Graphics (SIGGRAPH '86 proceedings)* (August 1986), vol. 20, pp. 151–160.

[27] Softimage Inc. *Softimage 3D Reference Guide: A to H, Interface*, 1996.

[28] Terzopoulos, D., and Fleischer, K. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics (SIGGRAPH '88 proceedings)* (August 1988), vol. 22, pp. 269–278.

[29] Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. Elastically

deformable models. In *Computer Graphics (SIGGRAPH '87 proceedings)* (July 1987), vol. 21, pp. 205–214.

[30] Terzopoulos, D., and Qin, H. Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics 13*, 2 (April 1994), 103–136.

[31] Thingvold, J. Elastic and plastic surfaces for modeling and animation. Master's thesis, University of Utah, 1990.

[32] University of Utah. *Alpha_1 User's Manual*, 1992.

[33] Welch, W., and Witkin, A. Variational surface modeling. In *Computer Graphics (SIGGRAPH '92 proceedings)* (July 1992), vol. 26, pp. 157–166.