ArtNova: Touch-Enabled 3D Model Design

Mark Foskey Miguel A. Otaduy Ming C. Lin
University of North Carolina at Chapel Hill
{*foskey,otaduy,lin*}*@cs.unc.edu*
http://www.cs.unc.edu/∼geom/ArtNova

Abstract

We present a system, ArtNova*, for 3D model design with a haptic interface.* ArtNova *offers the novel capability of interactively applying textures onto 3D surfaces directly by brush strokes, with the orientation of the texture determined by the stroke. Building upon the framework of* inTouch *[GEL00], it further provides an intuitive physically-based force response when deforming a model. This system also uses a* user-centric *viewing technique that seamlessly integrates the haptic and visual presentation, by taking into account the user's haptic manipulation in dynamically determining the new viewpoint locations. Our algorithm permits automatic placement of the user viewpoint to navigate about the object.* ArtNova *has been tested by several users and they were able to start modeling and painting with just a few minutes of training. Preliminary user feedback indicates promising potential for 3D texture painting and modeling.*

Keywords: Haptics, Modeling, 3D Painting, Textures

1 Introduction

Designing 3D digital models is an important part of the production process in VR, computer game design, entertainment and education. Model design involves both 3D geometry and surface appearance, and each component offers its own challenges. Designing 3D shapes is difficult when one's input tool has only two degrees of freedom; and painting a surface is further complicated by the fact that the screen is flat while the object being painted can be curved and non-convex. With a physical model one could simply shape it in three dimensions and then paint on its surface. We propose to make model design easier by emulating that kind of experience as faithfully as possible, while offering users more flexibility and power via digital media.

In physical sculpting, the sense of touch is essential. Force display is now making it possible to add some degree of haptic feedback to the 3D sculpting experience. As an early attempt to provide touch-enabled modeling features, a non-commercial plug-in to Alias|Wavefront's Power Animator software package was developed at SensAble Technologies [Mas98], but force update rates were too low to provide a realistic feel. Only recently have commercial haptic sculpting systems, such as $FreeForm^{\mathsf{TM}}$ [ST99], been introduced. They allow artists and designers to express their creativity with *3D-*$Touch^{\mathsf{TM}}$.

Figure 1. A turtle modeled and painted using *ArtNova*. Notice the patterns on its back and legs.

1.1 Main Results

Our goal is to develop a digital model design sytem that supports geometric modeling and texture painting with a direct 3D interface via force-feedback devices. Our system offers 3D texture painting on arbitrary polygonal meshes with the haptic stylus as an "electronic paintbrush". Unlike most of the existing haptic sculpting systems [MQW01, RE99, ST99], we use subdivision surfaces as the underlying geometric representation with a physically-based force model for deforming the models. The turtle in Figure 1 was created and texture-painted by *ArtNova*. Notice the shell pattern on the back, and the mottling on the legs and face.

In this paper, we present *ArtNova*, an integrated system for 3D texture painting and multiresolution modeling with haptic interface and user-centric viewing. Our system has the following characteristics:

- **Interactive 3D Texture Painting** – Given true 3D interaction via a force feedback device, we can interactively apply predefined textures directly onto the surfaces of the model without object registration problems, in addition to painting monochrome colors.

- **Dynamically Adjusted Viewing** – Viewpoint locations have a direct impact on the quality of the graphical display accompanying haptic editing. In addition to the typical 3D object grabbing capability, our system offers automatic repositioning of the object to place a selected point near the center of the field of view, without the user having to switch between haptic editing and camera repositioning. Also, it provides incremental viewpoint navigation based on the user's gestures to provide proper views of the regions under haptic manipulation.

- **Physically-based Force Response** – A spring-based force model is used to emulate the surface tension when the user is pulling or pushing to edit the subdivision meshes.

Several users have been able to use *ArtNova* to create interesting models with just a few minutes of training. In addition to painting with monochrome colors [GEL00], it has also been used to apply textures onto a 3D model's surface as well. The resulting meshes can be used directly for rendering and simulation without any format conversion. Preliminary user feedback suggests promising potentials of haptic interfaces for 3D painting and modeling.

1.2 Related Work

Haptic Interaction: Several real-time virtual environment systems have incorporated a haptic interface to enhance the user's ability to perform interaction tasks [HCT+97, MRF+96, MS94, RKK97]. Gibson [Gib95] and Avila and Sobierajski [AS96] have proposed algorithms for object manipulation including haptic interaction with volumetric objects and physically-realistic modeling of object interactions. Recently, SensAble Technologies developed the $FreeForm^{TM}$ modeling system to create and explore 3D forms using volumetric representations [ST99].

Geometric Modeling: There is an abundant wealth of literature on geometric modeling, interactive model editing, and deformation methods applied to curves and surfaces. Geometric formulations for model editing can be classified as pure-geometric representations such as NURBS [PT97], free-form deformation (FFD) [SP86], or physically-based modeling techniques such as D-NURBS [QT96]. With a similar mathematical framework to hierarchical editing [FB88], subdivision methods allow modeling of arbitrary topology surfaces [SZ98], while supporting multiresolution editing [DKT98, HDD+94, KS99, SZMS98, ZSS97]. There are also other sculpting techniques based on volumetric modeling methods [GH91, MQW01, RE99, PF01]. The haptic modeling system *inTouch* uses subdivision surfaces as its underlying geometric representation [GEL00], but it lacks a physically-based force model to generate a realistic feedback sensation.

3D Texture Painting: By using standard graphics hardware to map the brush from screen space to texture space, Hanrahan et al. allowed the user to paint directly onto the model instead of into texture space [HH90]. This approach has been applied to scanned surfaces using 3D input devices, such as data gloves and a Polhemus tracker [ABL95]. However, the painting style of both systems can be awkward, due either to the difficulty in rotating an object for proper viewing during painting, or to the deviation in paint location introduced by the registration process.

General texture mapping approaches, such as [BVIG91, MYV93], are powerful but require user input to generate the map. The approaches used in the Chameleon system [IC01] are more appropriate for casual, quick painting and not for highly detailed models. There are also commercial 3D painting systems [Hem00, COR00]. Most of them often use awkward and non-intuitive mechanisms for mapping 2D textures onto 3D objects, or require that a texture for the model be provided. None offers the natural painting style desired by artists and designers.

Johnson et al. introduced a method for painting a texture map directly onto a trimmed NURBS model using a haptic interface [JTK+99]. Its simplicity and intuitive interface support a natural painting style. However, its parameterization technique is limited to NURBS and does not apply to polygonal meshes, which are more commonly encountered in computer graphics and animation.

Our approach for haptic painting is similar to that presented in [GEL00] which can only paint colors. Our texture painting bears some resemblance to lapped textures [PFH00]. That work determines the orientation of the texture for overlapping patches based on user-specified vector fields, whereas ours applies textures *interactively* to a local region by brush strokes, with the orientation of the texture determined directly by the stroke.

Camera Placement: Camera control is a fundamental problem for 3D graphics applications. Several techniques on user interfaces for camera control have been proposed, including orbiting techniques mapping 2D motion into 3D interaction [CMS88, PBG92, Wer94, ZF99], use of image plane constraints [GW92, PFC+97], and direct camera manipulation using a 6DOF input device [WO90]. Our approach differs from many of the existing techniques that use 2D input devices to directly manipulate the viewpoint. Our main focus in ArtNova is to achieve automatic placement of viewpoint via *implicit control* based on the user's manipulation of the haptic device. Our camera positioning techniques are also useful for touch-enabled exploration of predefined scenes and massive models [OL01].

1.3 Organization

The rest of the paper is organized as follows. Section 2 gives an overview of our system and its user interface. We present our new texture painting algorithm in Section 3. Multiresolution modeling based on subdivision surfaces with a spring-based force computation is presented in Section 4. Section 5 describes a novel feature for dynamically adjusting the viewpoint, as the objects are manipulated. We briefly highlight the implementation of our prototype system with a haptic interface and demonstrate its features via the actual artistic creations of several users in Section 6.

2 Overview

In this section we give an overview of the system architecture and the user interface of our system.

2.1 System Architecture

The overall system consists of a haptic server and a graphical client application, connected using VRPN [VRPN], a library for distributed virtual reality applications. As with *inTouch*, a copy of the model is retained on both the haptic server

and graphical client, and calculations that deform the model are duplicated on both applications, so that the changes in position of numerous vertices need not be passed over the network. An overview of the system architecture is given in Figure 2.

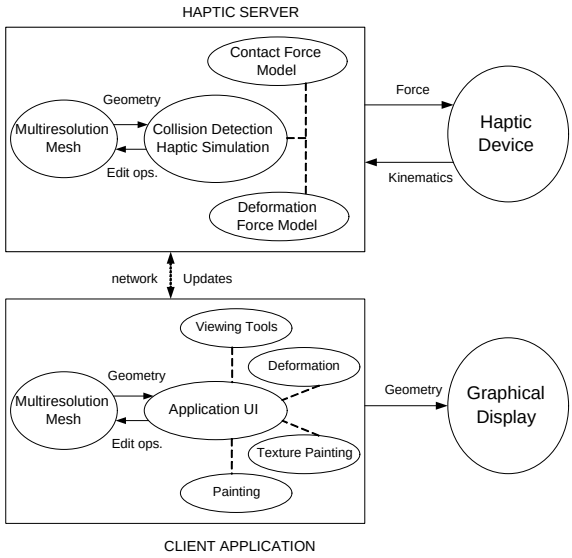

Figure 2. System Architecture.

2.2 User Interface

ArtNova allows the user to edit the geometry and the surface appearance of a model by sculpting and painting with a haptic interface. The user sees the model being edited, the tool being used, and a menu that can be operated using either the haptic tool or a mouse. Each type of model manipulation is indicated by a different tool. For example, the user moves the object with a mechanical claw, paints with a paintbrush, and deforms with a suction cup.

As an alternative to the claw tool for moving the object, the user's viewpoint can be changed using the viewing techniques described in Sec. 5. An *automatic repositioning* feature lets the user move the last touched point on the model to the center of the viewing area using a single keystroke, and there is a "flying mode" controlled by the haptic device.

For painting there are a continuous color picker, sliders for brush width and falloff of paint opacity, and choice of textures for texture painting. The width of the stroke can also be changed by adjusting the pressure applied when painting.

A basic undo feature is provided for deformations and painting, and there are provisions for saving models and screen shots. A snapshot of the user interface is shown in Figure 3.

3 Texture Painting

In addition to modeling, *ArtNova* allows the user to paint textures and colors onto the model using a virtual paintbrush. Arbitrary polygonal models can be painted, and each stroke has a configurable *falloff*, fading into the background

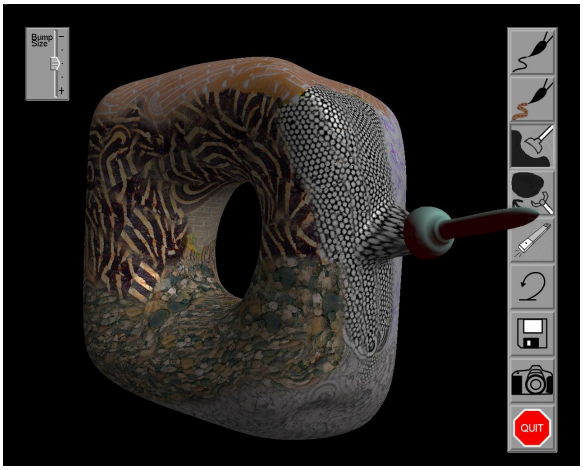

Figure 3. The graphical user interface. The user is performing a deformation on a painted toroidal base mesh.

near the boundaries of the stroke. The detailed algorithm for painting monochrome colors was described in [GEL00]. We briefly summarize it here before describing our novel texture painting method.

3.1 Painting Algorithm

Each stroke of the brush is decomposed into a sequence of *stroke segments*, which are represented as 3-space vectors linking the positions of the tool tip at successive frames. The brush radius is computed at the stroke endpoints (based on the force exerted by the user) and linearly interpolated across the length of the vector. This radius determines a volume of influence in 3D. Triangles are painted starting with the one containing the tail of the stroke segment. When one triangle has been painted, any neighbors that also extend into the volume of influence are then painted.

We paint each triangle by rasterizing the corresponding triangle in texture space. As the triangle is rasterized, for each texel p_t we determine the corresponding point p_w in world space on the surface of the model. We then calculate D as

$$D = \frac{\|p_w - q\|}{R_s(q)},$$

where q is the point on the stroke segment nearest p_w, and $R_s(q)$ is the stroke radius at q. Once we have D, we compute the color of p_t by blending the color being painted with the background according to a falloff function depending on D.

Because we use a straight vector to represent a portion of a stroke that actually follows a curved surface, there can be artifacts if the surface curvature relative to the length of the stroke segment is appreciable. Typically, however, the stroke segments are quite short and this problem does not arise.

3.2 Texture-Mapped Paint

Texture-mapped paint builds upon our algorithm for painting monochrome colors. For clarity we will refer to the

texture map into which colors are drawn as the *target texture map*, and the predefined texture array as the *source texture map*. For each texel being rasterized in the target texture map, we compute the corresponding point p_w in world space, and its nearest neighbor q on the stroke segment (see Figure 4). We then compute two coordinates s and t that will be used to look up colors in the source texture map. The s coordinate represents length along the stroke, and t represents signed distance from the stroke. We compute s and t as follows: Let the current stroke segment be represented by a vector $\vec{v}$ with tail p_v. Then, for s, we maintain the total length of all previous segments of a given stroke, and add the distance $\|q - p_v\|$. The sign of t is equal to the sign of

$$((p_w - p_v) \times \vec{v}) \cdot \vec{n}$$

where $\vec{n}$ is the stored normal vector of the triangle containing p_w. The magnitude of t is just $\|p_w - q\|$.

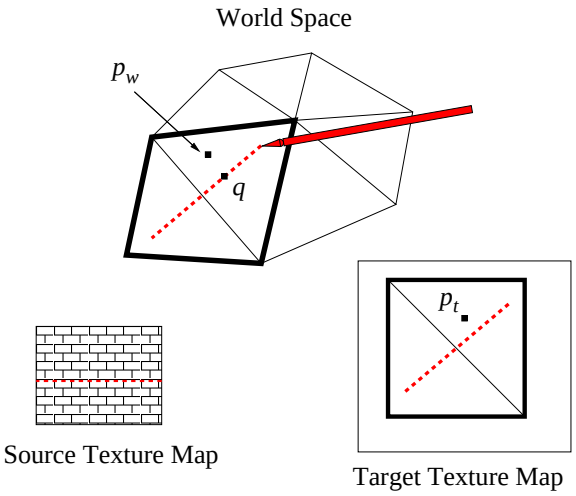

Figure 4. Texture painting.

Before s and t are used to look up a color, they are scaled by a user-adjustable factor so that the texture features will be the desired size on the model. Texture lookup is performed modulo the dimensions of the texture patch, so it is important that the texture be periodic. Note that, although the texture will repeat along a stroke, the user can break up the periodicity by starting new strokes and altering stroke orientation.

At the boundary between stroke segments, the point q on the stroke vector nearest to p_w may be an endpoint. If q is the tail end of the stroke segment, texture is not applied for that point. If q is the front, then t is still taken as the distance from p_w to q, with sign given as above, while s is fixed. This has the effect of replicating pixels from the trailing stroke to fill gaps between roughly rectangular strips. Figure 5 shows an example of a textured cow model.

3.3 Accelerated Texture Painting

The original algorithms for color and texture painting rasterized the color data into the texture memory in software. We have developed an improved algorithm that relies on modern graphics hardware to copy the texture. For each stroke segment, we first accumulate a list of triangles that extend into the volume of influence of the segment. Then the region of the texture parameter domain covered by those triangles is rendered with a fixed z coordinate in an orthogonal projection. If there are multiple patches in the target texture, the relevant triangles from each patch are rendered into a different region of the frame buffer. These triangles are textured, using coordinates generated dynamically that index into the source texture memory. The resulting image is then copied into the target texture memory with alpha blending. To produce the appropriate falloff function, the source texture has been prepared with the alpha values determined by the falloff. We can paint simple colors by choosing a monochromatic source texture.

Figure 5. Cuts of beef indicated by different textures using *ArtNova*.

3.4 Texture Coordinates

The above algorithms assume that texture coordinates for all triangles in the model have been determined. If they are not provided, we use a simple algorithm for generating texture coordinates. Since the user is painting on the model, not the texture map, we need not concern ourselves with contiguity of the image in texture memory. Also, there is relatively little concern with distortion, for the same reason. If the choice of texture coordinates distorts the image, the distorted version of the image will appear in the texture memory, not on the model. The algorithm we currently use groups triangles into pairs, which are mapped to squares in texture space. The areas of the squares are chosen to approximately reflect the initial areas of the triangles.

3.5 Comparison with other approaches.

One sophisticated program for painting on 3D models is DeepPaint [Hem00], a commercial product distributed by Right Hemisphere. Its interaction is fundamentally two dimensional, but the geometric model for applying a texture to the surface can be compared as an issue on its own. Deep

Paint offers two modes for texture painting. In one, the texture is copied directly to the texture memory, so that distortions in the parameterization are reflected in the appearance of texturing. For instance, the scale of the texture may be finer near the "north pole" of a sphere. In the other mode, the texture is applied in screen space, and then remapped as it is transferred to texture memory. This is the same approach that Chameleon [IC01] uses to apply solid color. This approach yields results that are independent of the parameterization of the object being painted, but it creates a distinctive "stretching" of the texture near the silhouette of the figure, which becomes noticeable when object is rotated to a different position. Because our approach determines temporary texture coordinates locally in the neighborhood of a portion of a single stroke, the overall scale of the texture is not dependent on where it is placed, making the painting more predictable for the user.

4 Multiresolution Mesh Editing

Our model editor is based on *inTouch*'s geometric framework [GEL00], which is strongly influenced by Zorin et al. [ZSS97]. For completeness, we briefly describe the framework here and then describe the new simplified force model used during deformation in *ArtNova*.

4.1 The Mesh Data Structure

We use a subdivision framework to represent our geometry. We store a coarse, triangular *base mesh* M_0 and several meshes at finer resolutions M_i ($i > 0$). By a single stage of Loop subdivision, each mesh M_i uniquely determines a finer mesh M_{i+1}^{sub}. M_{i+1}^{sub} is used as a reference mesh for the definition of M_{i+1}. Every vertex in the actual mesh M_{i+1} corresponds to a vertex of M_{i+1}^{sub}, but differs from it by a *displacement vector* stored with the vertex. In this way we can choose to edit at a specific resolution by moving vertices of a given mesh M_i. Vertices at finer levels retain their displacement vectors and are thus carried along by the motion of the subdivided surface.

In principle we could modify M_i without changing M_{i-1} at all, since the vertices of M_i are different from the vertices of M_i^{sub} (gotten by subdividing M_{i-1}). However, we also perform a smoothing step using a method given by Taubin [Tau95] to modify coarser levels. In this way, for instance, an accumulation of edits at a high resolution, all tending to raise up one side of the model, can result in a repositioning of the coarser level vertices to better reflect the new overall geometry of the model.

4.2 Deformation Algorithms

To edit the model, the user simply places the tool against the model, presses the PHANTOM button, and moves the tool. As the surface is edited, the user can feel a resisting force and see the surface deform. The edit resolution (the choice of mesh M_i to modify directly) is presented to the user as a "bump size."

4.2.1 Surface Modification

Surface deformation is performed by moving a single triangle of the edit mesh M_i. When the user begins a deformation, the point of contact with the surface determines a unique triangle at the edit resolution, and a unique reference point on that triangle. For each frame, the successive positions of the tool tip define a motion vector $\vec{m}$, which is used to move the three vertices of the selected triangle. Each vertex is moved in the direction of $\vec{m}$, and by a distance scaled so that vertices nearer the current point of the tool tip are moved farthest. More precisely, the distance d_i from the reference point to each vertex v_i is computed, and the movement vector $\vec{m}_i$ for each vertex is given by

$$\vec{m}_i = \left(1 - \frac{d_i}{d_0 + d_1 + d_2}\right)\vec{m}.$$

4.2.2 Force Model

When the user places the tool against the model, there is a restoring force generated by the haptic rendering library, based on collision information from H-Collide [GLGT00]. When the user begins deforming the surface, the restoring forces are turned off, and the initial 3-space location of the tool tip, p_0, is recorded. The user is then free to move the tool in any direction. To provide feedback, a Hooke's law spring force is established between the tool tip and p_0, given by

$$f = -k(p_{\text{tip}} - p_0),$$

where p_{tip} is the location of the tool tip and k is a small spring constant.

The spring constant is chosen so that the user can move the tip a sizable distance in screen space before the force becomes a substantial hindrance. When the user releases the button, the spring force is turned off and the usual restoring forces are turned on with the surface in its new position.

Because our force model is based on the initial position of the tool, the force computation is decoupled from the position of the surface. This provides a smoother feel to the user than computing the force from the instantaneous distance to the surface that is being moved, because computing the surface deformation is much slower than the 1 kHz haptic response rate.

5 Dynamic Viewing Techniques

As the user performs painting or modeling tasks over the entire model, the user will need to edit back-facing portions of the model from time to time. Typically the user repositions the model by performing a "grab and turn" operation using the application. We have developed several novel *user-centric* viewing techniques that make this task easier and more efficient. In addition to using the force feedback device for haptic manipulation, we also use it implicitly, and simultaneously, as a mechanism for viewpoint placement. These techniques allow users to express their intentions in a natural way, with a minimum of switching between editing and changing the view.

Our approach to the problem is to reposition the camera based on the configuration (i.e. position and orientation) of the haptic probe so that the region of interest on the model surface is placed at the center of the view. We call our techniques "user-centric" because the haptic tool indicates where the user wants to view the object from, rather than where the object should be.

5.1 Grabbing Operation

In the typical grabbing operation [RH92], the object is moved preserving the transformation between the virtual probe and the grabbing point. At the instant of grabbing, two points are picked: A, a point in the object, and B, the current position of the virtual probe. The transformation T_{BA} from the probe to the object is set constant as the virtual probe moves. Therefore, a displacement of the probe, ΔT_B, produces a displacement of the object, ΔT_g.

$$T_{BA} = T_A \, {T_B}^{-1},$$

$$\Delta T_g = T_{BA} \, \Delta T_B \, {T_{BA}}^{-1}.$$

where T_A and T_B are the transformations that represent the location and orientation of the object and the virtual probe.

5.2 Viewpoint Navigation

We introduce a new concept called "viewpoint navigation". This functionality is based on moving the viewpoint around the object to be modeled or painted, based on the gestures of the user. The position and orientation of the haptic device at each time are used as "commands", which set a transformation, T_n, on the position of the virtual probe and the camera, as shown in Fig. 6.

In our system we apply the inverse transformation to the object that is being manipulated, producing the same visual effect. In addition, we apply this transformation incrementally, whenever the viewpoint navigation is enabled.

$$\Delta T_n = K \, \Delta T_H.$$

where ΔT_H represents the variation of the position and orientation of the haptic device.

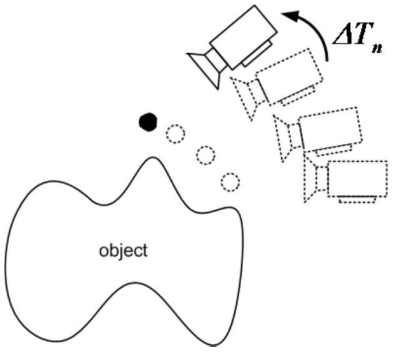

Figure 6. Viewpoint navigation based on user's intentions.

5.3 Automatic Repositioning

Another novel functionality called *automatic repositioning* allows the user to reorient the model quickly without interrupting the work flow to change tools. When the automatic repositioning key is pressed, the last point touched on the model is taken to define the region of interest. A transformation T_r is computed that moves the camera to a location along the normal direction of the object at the point of interest (as shown in Fig. 7). Then, we apply the inverse transformation to the object incrementally, so that the object appears to rotate to the new position.

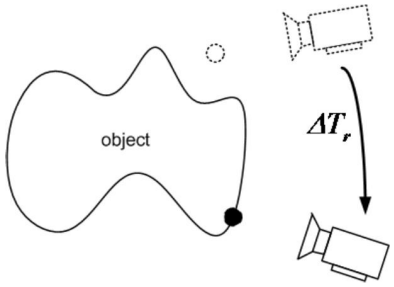

Figure 7. Automatic repositioning of viewpoint.

5.4 Combining all the Functionalities

The transformation that sets the location and orientation of the object is updated every time step for better viewing to aid the model manipulation and editing tasks, depending on which functionality is enabled. We have:
$T_{A,k+1} = \Delta T_g \, T_{A,k}$, if grabbing is enabled;
$T_{A,k+1} = {\Delta T_n}^{-1} \, T_{A,k}$, if viewpoint navigation is enabled;
$T_{A,k+1} = {\Delta T_r}^{-1} \, T_{A,k}$, if automatic repositioning is enabled.

6 Results

We have designed and implemented *ArtNova* as a proof-of-concept prototype system. In this section, we demonstrate the system capability and discuss issues related to the user interface design.

6.1 Prototype Demonstration

We use a dual-processor Pentium III PC as a haptic server, a SensAble Technologies' PHANTOM as a haptic device, a Silicon Graphics Inc. R12000 Infinite Reality for rendering, a large screen with a back projection system for graphical display, and UNC's VRPN library [VRPN] for a network-transparent interface between application programs and our haptic system. The system is written in C++ using the OpenGL and GLUT libraries. However, the design framework of our system is applicable to all types of haptic devices and libraries, as well as graphical display and computing platforms.

Several users with little or no experience in using modeling or painting systems were able to create interesting models using *ArtNova* with little training. Each user was given

Figure 8. A tree.

a selection of simple base meshes of various shapes (e.g. spherical, toroidal, etc.) and light gray in color. A few examples of their art work are given in Figures 1, 3, 8, and 9.

The accompanying videotape illustrates a few examples of the modeling and painting operations performed using *ArtNova*. Along with some more images of the models created by our users, the video clips are also available at:

http://www.cs.unc.edu/∼geom/ArtNova.

6.2 User Experiences

We have asked several users to evaluate our system by creating and texture-painting a few models. We briefly summarize some of their comments here:

- The spring-based force model feels natural. Comparing the experiences of performing model deformation with and without the physically-based force model, most of the users found the newly added force model to be an improvement.
- The users commented that the force feedback was useful in detecting and maintaining contact with the model surfaces, when performing highly detailed painting.
- Users also felt that texture painting was easy to use, and had little trouble getting visually pleasing results.
- Users found automatic viewpoint adjustment and repositioning to be intuitive and natural.
- The overall graphical user interface with our new 3D tool metaphor was found to be intuitive and easy to understand.

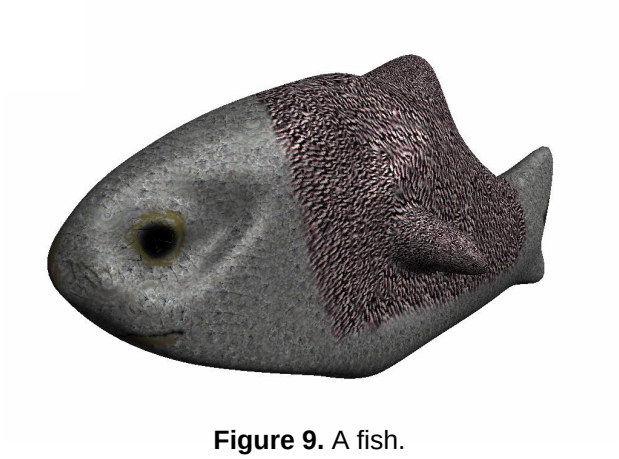

Figure 9. A fish.

7 Summary

We have presented an integrated system for 3D texture painting and multiresolution modeling with a haptic interface and user-centric viewing. An artist or a designer can use this system to create and refine a three-dimensional multiresolution polygonal mesh, and further enhance its appearance by directly painting textures onto its surface. The system allows users to naturally create complex forms and patterns aided not only by visual feedback but also by their sense of touch. Based on preliminary user feedback, we believe these features considerably improve the ease and expressiveness of 3D modeling and texture painting. We are currently planning an extensive, formal user study to carefully evaluate and assess the contribution of various elements of this system on enhancing users' experiences in digital model design.

8 Acknowledgments

This research is supported in part by a fellowship of the Government of the Basque Country, NSF DMI-9900157, NSF IIS-9821067, ONR N00014-01-1-0067 and Intel. We would like to thank the following individuals for their suggestion and assistance: Arthur Gregory, Stephen Ehmann, Michael Rosenthal, Adrian Ilie, Sarah Hoff, Bryan Crumpler, Derek Hartman, Scott Cooper, and Joohi Lee.

References

[ABL95] Maneesh Agrawala, Andrew C. Beers, and Marc Levoy. 3D painting on scanned surfaces. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 145–150. ACM SIGGRAPH, April 1995.

[AS96] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. *Proceedings of Visualization'96*, pages 197–204, 1996.

[BVIG91] Chakib Bennis, Jean-Marc Vézien, Gérard Iglésias, and André Gagalowicz. Piecewise surface flattening for non-distorted texture mapping. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 237–246, July 1991.

[CMS88] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. In John Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 121–129, August 1988.

[COR00] COREL. Painter. *http://newgraphics.corel.com/products/painter6.html*, 2000.

[DKT98] T. DeRose, M. Kass, and T. Troung. Subdivision surfaces in character animation. *Proc. of ACM SIGGRAPH*, 1998.

[FB88] D. Forsey and R.H. Bartels. Heirarchical B-spline refinement. In *Proc. of ACM Siggraph*, pages 205–212, 1988.

[GEL00] A. Gregory, S. Ehmann, and M. C. Lin. *inTouch*: Interactive multiresolution modeling and 3d painting with a haptic interface. *Proc. of IEEE VR Conference*, 2000.

[GH91] Tinsley A. Galyean and John F. Hughes. Sculpting: An interactive volumetric modeling technique. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 267–274, July 1991.

[Gib95] S. Gibson. Beyond volume rendering: Visualization, haptic exploration, and physical modeling of element-based objects. In *Proc. Eurographics workshop on Visualization in Scientific Computing*, pages 10–24, 1995.

[GLGT00] A. Gregory, M. Lin, S. Gottschalk, and R. Taylor. Real-time collision detection for haptic interaction using a 3-dof force feedback device. *Computational Geometry: Theory and Applications, Special Issue on Virtual Environments*, 15(1-3):pp. 69–89, February 2000.

[GW92] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 331–340, July 1992.

[HCT+97] J. Hollerbach, E. Cohen, W. Thompson, R. Freier, D. Johnson, A. Nahvi, D. Nelson, and T. Thompson II. Haptic interfacing for virtual prototyping of mechanical CAD designs. *CDROM Proc. of ASME Design for Manufacturing Symposium*, 1997.

[HDD+94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of ACM SIGGRAPH*, pages 295–302, 1994.

[Hem00] Right Hemisphere. Deep paint. *http://www.us.deeppaint.com/*, 2000.

[HH90] Pat Hanrahan and Paul E. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 215–223, August 1990.

[IC01] T. Igarashi and D. Cosgrove. Adaptive unwrapping for interactive texture painting. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 209–216, 2001.

[JTK+99] D. Johnson, T. V. Thompson II, M. Kaplan, D. Nelson, and E. Cohen. Painting textures with a haptic interface. *Proceedings of IEEE Virtual Reality Conference*, 1999.

[KS99] A. Khodakovsky and P. Schröder. Fine level feature editing for subdivision surfaces. *Proceedings of ACM Symposium on Solid Modeling and Applications*, 1999.

[Mas98] Thomas Massie. A tangible goal for 3D modeling. *IEEE Computer Graphics and Applications*, May/June, 1998.

[MQW01] K. McDonnell, H. Qin, and R. Wlodarczyk. Virtual clay: A real-time sculpting system with haptic interface. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 179–190, 2001.

[MRF+96] William Mark, Scott Randolph, Mark Finch, James Van Verth, and Russell M. Taylor II. Adding force feedback to graphics systems: Issues and solutions. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 447–452, 1996.

[MS94] T. M. Massie and J. K. Salisbury. The phantom haptic interface: A device for probing virtual objects. *Proc. of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 1:295–301, 1994.

[MYV93] Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 27–34, August 1993.

[OL01] M. Otaduy and M. Lin. User-centric viewpoint computation for haptic exploration and manipulation. *Proc. of IEEE Visualization*, 2001. To appear.

[PBG92] C. Phillips, N. Badler, and J. Granieri. Automatic viewing control for 3D direct manipulation. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 71–74, 1992.

[PF01] R. Perry and S. Friskin. Kizamu: A system for sculpting digital characters. *Computer Graphics (ACM SIGGRAPH'01)*, 2001.

[PFC+97] J. Pierce, A. Forsberg, M. Conway, S. Hong, R. Zeleznik, and M. Mine. Image plane interaction techniques in 3D immersive environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 39–44, 1997.

[PFH00] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. *Proc. of ACM SIGGRAPH*, pages 465–470, 2000.

[PT97] L. A. Piegl and W. Tiller. *The NURBS Book*. Springer Verlag, 1997. 2nd Edition.

[QT96] Hong Qin and Demetri Terzopoulos. D-NURBS: A physics-Based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96, March 1996. ISSN 1077-2626.

[RE99] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. *ACM Symposium on Solid Modeling and Applications*, 1999.

[RH92] Warren Robinett and Richard Holloway. Implementation of flying, scaling, and grabbing in virtual worlds. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 189–192, March 1992.

[RKK97] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, August 1986.

[ST99] SensAble Technologies Inc. $freeform^{TM}$ modeling system. *http://www.sensable.com/freeform*, 1999.

[SZ98] P. Schröder and D. Zorin. Subdivision for modeling and animation. *ACM SIGGRAPH Course Notes*, 1998.

[SZMS98] T. Sederberg, J. Zheng, M.Sabin, and D. Sewell. Non-uniform recursive subdivision surfaces. *Computer Graphics (ACM SIGGRAPH'98)*, 1998.

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 351–358. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[VRPN] Virtual Reality Peripheral Network. http://www.cs.unc.edu/research/nano/manual/vrpn.

[Wer94] Josie Wernecke. *The Inventor Mentor*. Addison-Wesley, 1994.

[WO90] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 175–183, 1990.

[ZF99] Robert Zeleznik and A. Forsberg. Unicam 2D gestural camera controls for 3D environments. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 169–173, 1999.

[ZSS97] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. *Computer Graphics (ACM SIGGRAPH'97)*, 1997.