Haptic Sculpting of Volumetric Implicit Functions

Jing Hua **Hong Qin**
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400, U.S.A.
{jinghua|qin}@cs.sunysb.edu

Abstract

Implicit functions characterized by the zero-set of polynomial-based algebraic equations and other commonly-used analytic equations are extremely powerful in graphics, geometric design, and visualization. But the potential of implicit functions is yet to be fully realized due to the lack of flexible and interactive design techniques. This paper presents a haptic sculpting system founded upon scalar trivariate B-spline functions. All the solids sculpted in our environment are semi-algebraic sets of volumetric implicit functions. We develop a large variety of sculpting toolkits equipped with an intuitive haptic interface to facilitate the direct manipulation of implicit functions in real-time. To facilitate multiresolution editing and different levels of details, we employ three techniques: hierarchical B-splines, CSG-based functional composition, and knot insertion. Our experiments demonstrate that our algorithms and haptics-based techniques can greatly overcome the modeling difficulties associated with implicit functions. The novel modeling techniques and their haptics-based design principle are extensible to the design of arbitrary implicit functions.

Keywords: *Geometric Modeling, B-splines, Implicit Function, Volume Sculpting, Marching Cubes Rendering, Haptic Interface.*

1. Introduction and Motivation

The efficiency and flexibility of shape modeling are vital to the success of graphics, geometric design, and virtual environments. Despite the prevalence of parametric forms in visual computing fields, the traditional representation of geometric entities such as commonly-used analytic shapes comes from implicit functions because of many of their attractive properties [1]. It can be shown that the set of implicit algebraic surfaces or solids is actually larger than that of rational parametric surfaces or solids. This set is also closed under certain geometric operations. Every rational parametric curve/surface/solid can be represented by an implicit algebraic equation, but not vice versa [13]. In contrast with parametric forms, implicit functions have a number of advantages such as point classification, intersection computation, unbounded geometry. Consider polynomial-based algebraic equations for example, the simplest form for implicit functions is the power basis expression of degree *n*

$$\sum_{i,j,k,i+j+k\le n} a_{ijk} x^i y^j z^k = 0 \tag{1}$$

Despite their representation potential, existing techniques associated with implicit functions have certain severe shortcomings. First, effectively digitizing and rendering an implicit function are oftentimes far from trivial. It is extremely difficult to control the shape of implicit solids while re-rendering the modified regions fast enough for their use within an interactive environment. Second, a designer often has no intuitive understanding of the effect of altering polynomial coefficients or adding/deleting components. Consider (1), the coefficients provide neither direct and natural geometric interpretation nor intuitive insight into the underlying shape. Third, there are no convenient tools for the intuitive shape control of this type of algebraic solids. Moreover, general implicit functions usually have a property of global control. In contrary, a large number of techniques and tools have been developed to afford global and local control of conventional parametric surfaces or solids. Yet, flexible and direct modeling techniques for implicit solids are under-explored.

We propose a novel modeling approach and a haptics-based sculpting principle that can integrate implicit functions with parametric representation such as piecewise scalar B-splines, which permit interactive and direct manipulation of implicit solids in real-time. Local control of the implicit solids can also be easily accomplished. This will enable designers to benefit from the low degree and computational efficiency of implicit functions. Ultimately, our endeavor should make it possible to achieve the full potential of implicit functions in commercial design systems.

Commonly used graphics systems often rely upon 2D mouse-based interfaces for 3D interaction. Direct operations on virtual objects with a 2D mouse are not as natural and intuitive as interaction via a 3D interface. To ameliorate this, we offer users a haptic interface for the intuitive and natural sculpting of volumetric implicit functions.

Haptics provides users a hand-based mechanism for intuitive, manual interactions with virtual environments towards realistic tactile exploration and manipulation. Haptics-based interaction has emerged as a critical metaphor in the fields of medicine, education, industry, entertainment, and computer arts. Our objective is to allow users to reach toward an object, feel the physical presence of its shape and manipulate it. With a standard haptic device, our approach permits users to interactively sculpt virtual materials having realistic properties and feel the physically realistic presence with force feedback throughout the design process. Using haptics in a virtual environment, designers are able to feel and sculpt real objects in a natural 3D setting, rather than being restricted to depend on 2D projections for input and output. Force feedback provides additional sensory cues to designers. This tactile exploration can afford designers to gain a richer understanding of the 3D nature. The use of haptics in a virtual design environment promises to increase the bandwidth of information between designers and the synthetic modeling world.

Prior research is primarily focused on haptic rendering (i.e. the feeling of rigid surfaces/solids). In contrast, our haptic sculpting system allows designers to interactively sculpt implicit solids in real-time. Our volumetric implicit functions are well suited for integration with a haptic approach because of the properties of implicit functions. In particular, the modeling of implicit functions simplifies the complicated computation of collision detection and depth penetration between implicit solids and points in 3D.

Throughout our system, the sculpted object is evaluated as a level set of a volumetric implicit function defined over a three-dimensional working space. Although we employ uniform or non-uniform B-splines as the underlying function, constituent functions may be of arbitrary type with or without the local control property. We further enhance scalar B-spline functions with additional features such as hierarchical decomposition and CSG-based operation. Through the knot insertion, our system takes advantage of both uniform and non-uniform B-spline functions so that the knot distribution will influence the local shape. Rather than indirectly modifying the coefficients associated with the volumetric implicit function as exhibited in prior work, our sculpting tools support the direct manipulation of implicit functions' scalar values. Our algorithms can automatically determine all of the unknown control coefficients and effectively reconstruct a new volumetric implicit function after the local/global modification. Our system offers a wide array of intuitive sculpting tools responsible for the effective construction of various complicated geometric shapes with diverse topologies. This allows designers to interactively and directly sculpt implicit solids with ease. The use of piecewise B-splines facilitates the rapid modification on arbitrary, localized regions. It may be noted that the integration of a haptic interface and volumetric implicit modeling should be of interest to much broader communities.

2. Background Review

2.1 Volume Sculpting

Galyean and Hughes [4] first introduced the concept of volume sculpting and developed a system with simple tools in 1991. Later, Wang and Kaufman [9] presented a similar sculpting system with sculpting tools of carving and sawing. In order to achieve real-time interaction, the system reduced the complex operations between the 3D tool volume and the 3D object to primitive voxel-by-voxel operations. Barentzen [11] proposed to use octree-based volume sculpting. The possibility to support multiresolution sculpting and its advantages were discussed at length. In a nutshell, the aforementioned sculpting systems were all dependent on the simple, voxel-based operation. The sculpted objects and the sculpting tools are represented using a discrete characteristic function. Unfortunately, only C^0 continuity could be achieved. In order to avoid the object spatial aliasing, the sculpted objects and sculpting tools need to undergo an appropriate filtering operation.

Recently, Raviv and Elber [5] presented a 3D interactive sculpting paradigm that employed a set of scalar uniform trivariate B-spline functions as underlying representation. The sculpted object was represented as the zero set of the trivariate functions. Users can indirectly sculpt objects to a desirable shape by directly modifying relevant scalar control coefficients of the underlying functions with tools. This work pioneers the use of a continuous characteristic function in 3D sculpting.

2.2 Implicit Functions

Blinn [3] demonstrated that implicit functions are well suited for both scientific visualization and the modeling tasks in computer graphics. Typical techniques include forcing an algebraic surface to interpolate a set of (regular or scattered) points or a network of spatial curves, and using piecewise algebraic patches to form a complex shape satisfying certain continuity requirements across patch boundaries. Sederberg [14][15] discussed the modeling techniques for cubic algebraic surfaces.

Hoffmann [16] systematically reviewed the implicit function techniques including the implicitization, parameterization, and the parametric/implicit conversion in CAGD. Bajaj and Ihm [17] presented an efficient algorithm to implement Hermite interpolation of low-degree algebraic surfaces with C^1 or G^1 continuity. Note that, neither point nor curve interpolation is an attractive mechanism for defining an implicit surface because it is difficult for designers to predict the surface behavior beyond interpolating curves and points.

Implicit functions can also be used to represent a solid. Commonly-used, yet simple solids such as spheres, cubes, cylinders, and tori are oftentimes used as primitives. To create more interesting shapes, primitive solids can be collected into a hierarchical organization with the help of Boolean operations. More complicated operations through the use of functional composition are also possible to generate more interesting shapes. The common feature essential to all implicit solid modeling methods [18][19] is the creation of an oriented three-dimensional boundary surface which partitions the entire 3-space into two distinct regions, namely the one occupied by the solid interior and the one outside of the defined solid.

2.3 Haptic Rendering

Haptic rendering is the process of applying forces through the use of force-feedback devices and augmenting a virtual environment with a haptic interaction. Haptic rendering requires: (1) sensing the position of the user's finger; (2) locating the contact point; and (3) appropriately generating a force to be applied to the finger. Thompson et al. [6] derived efficient intersection techniques that can be applied to nearly any type of haptic interface. Dachille et al. [20] developed a haptic interface to permit the direct manipulation of dynamic surfaces. McDonnell et al. [7] employed haptic toolkits to explore the dynamic subdivision solids. Avila et al. [8] presented a haptic interaction that is suitable for both volume visualization and modeling application. Despite the widespread application of haptics in visual computing areas, haptics-based interaction was mainly applied to parametric representations for shape sculpting. We integrate the principle of haptic modeling with the direct manipulation of implicit solids.

3. Volumetric Implicit Functions

3.1 Tensor-Product Scalar B-splines

Throughout this paper, we utilize scalar trivariate B-spline functions as the underlying shape primitives for object representation. The use of implicit B-spline functions for solid modeling is strongly inspired by their attractive properties including simplicity, generality, local control, etc. The generic B-spline functions are of the following form:

$$s(u,v,w)=\sum_{i=0}^{l-1}\sum_{j=0}^{m-1}\sum_{k=0}^{n-1} p_{ijk}B_{i,r}(u)C_{j,s}(v)D_{k,t}(w) \quad (2)$$

where $s(u,v,w)$ represents the scalar value at position (*u, v, w*) in parametric domain. *u, v, w* change from 0 to *U*, *V*, *W*, which represent the size of sampling points along three dimensions of parametric domain. p_{ijk} are the scalar control coefficients with the domain of *I, J, K* that are [*0, l-1*], [*0, m-1*], [*0, n-1*], respectively. In addition, $B_{i,r}(u)$, $C_{j,s}(v)$ and $D_{k,t}(w)$ are the basis functions corresponding to p_{ijk}, evaluated at (*u, v, w*). The degrees of the three basis functions are *r-1*, *s-1*, and *l-1*, respectively. To simplify the mathematical notation, (2) can also be expressed as the following matrix form:

$$\mathbf{s}=(\mathbf{B}\otimes\mathbf{C}\otimes\mathbf{D})\mathbf{p} \quad (3)$$

where $\otimes$ denotes Kronecker Product, and

$$\mathbf{s}=[\cdots,s_{ijk},\cdots]^T(i\in[0,U],j\in[0,V],k\in[0,W])$$

$$\mathbf{p}=[\cdots,p_{ijk},\cdots]^T(i\in[0,l-1],j\in[0,m-1],k\in[0,n-1])$$

B, **C**, and **D** are matrices composed of the sampling of basis functions. They are of the following forms:

$$\mathbf{B}=\begin{bmatrix} B_{0,r}(u_0) & B_{1,r}(u_0) & \cdots & B_{I,r}(u_0) \\ B_{0,r}(u_1) & B_{1,r}(u_1) & \cdots & B_{I,r}(u_1) \\ \vdots & \vdots & \cdots & \vdots \\ B_{0,r}(u_U) & B_{1,r}(u_U) & \cdots & B_{I,r}(u_U) \end{bmatrix} \quad \mathbf{C}=\begin{bmatrix} C_{0,s}(v_0) & C_{1,s}(v_0) & \cdots & C_{J,s}(v_0) \\ C_{0,s}(v_1) & C_{1,s}(v_1) & \cdots & C_{J,s}(v_1) \\ \vdots & \vdots & \cdots & \vdots \\ C_{0,s}(v_V) & C_{1,s}(v_V) & \cdots & C_{J,s}(v_V) \end{bmatrix}$$

$$\mathbf{D}=\begin{bmatrix} D_{0,t}(w_0) & D_{1,t}(w_0) & \cdots & D_{K,t}(w_0) \\ D_{0,t}(w_1) & D_{1,t}(w_1) & \cdots & D_{K,t}(w_1) \\ \vdots & \vdots & \cdots & \vdots \\ D_{0,t}(w_W) & D_{1,t}(w_W) & \cdots & D_{K,t}(w_W) \end{bmatrix}$$

$(\mathbf{B}\otimes\mathbf{C}\otimes\mathbf{D})$ could be precomputed in order to save runtime computation and improve real-time performance.

3.2 Implicit Solids

The implicit function can be generally characterized as:

$$\{(x,y,z)\mid F(x,y,z)=0\} \quad (4)$$

The bounding surface defined by an implicit function is a level-set

$$\begin{cases} w=F(x,y,z) \\ w=w_0 \end{cases} \quad (5)$$

By collecting all the level sets whose return values are greater (or smaller) than a given threshold, we could define a implicit solid.

$$\begin{cases} w=F(x,y,z) \\ w>w_0 \end{cases} \quad (6)$$

The advantages of implicit forms have been briefly documented in Section 1. In principle, the modeling schemes founded upon implicit forms are much more powerful than that of parametric-driven geometric modeling. However, modeling techniques based on implicit functions are not yet widespreadly explored due to the lack of direct manipulation mechanism. Our modeling system can bridge the large gap towards the full realization of all the modeling power of implicit functions.

3.3 Volumetric Implicit Functions

We shall collect different B-spline patches defined over the 3D working space to form a volumetric implicit function that can be collectively used to represent objects of complicated geometry and arbitrary topology. Note that, significantly different from commonly-used parametric B-splines, implicit B-spline functions formulate the scalar value distribution in 3D where implicit solids are uniquely defined as semi-algebraic point sets. Raviv and Elber [5] used a similar representation to implement a freeform sculpting system. In our system, we further enhance the B-spline representation power by incorporating the modeling advantages from hierarchical splines, generalized CSG-based Boolean operations, and non-uniform knot insertion.

3.3.1 Hierarchical Organization

Let us assume users have defined *N* B-spline patches over the sculpting working space, which are located at any location and with any orientation. In general, these patches may be formulated by different number of control coefficients in order to achieve the goal of multiresolution analysis and level-of-details control. Then the scalar value at the location (*x, y, z*) can be computed as

$$F(x,y,z)=\sum_{i=1}^{N} s_i(\mathbf{T}_i(x,y,z)) \tag{7}$$

where $\mathbf{T}_i$ is an affine transformation from the Euclidian space to the parametric domain of patch s_i. Since the trivariate B-spline has the affine invariance property, this transformation can be easily implemented. For each different patch s_i, there is a corresponding transformation $\mathbf{T}_i$. Now $F(x,y,z)$ becomes a new volumetric implicit function defined over the 3D working space. Without loss of generality, we make use of cubic B-splines with nonperiodic knot vectors. In order to make the boundaries of different trivariate patches achieve C^1 continuity, the first and last 4 layer control coefficients along three principal directions of the parametric domain should be set to zero.

3.3.2 CSG-based Operations

Users may intend to sculpt implicit solids to form sharp features over their boundaries or change the continuity requirements across their smooth boundaries. Feature-based sculpting tools can significantly improve the system performance. In light of this demand from users, our system provides CSG-based operations on any user-defined trivariate patche in order to facilitate the rapid construction of complicated models satisfying many feature-oriented requirements. Therefore, complicated geometry is readily available in our system through the use of

$$F(x,y,z)=\Omega_{i=1}^{N} s_i(\mathbf{T}_i(x,y,z)) \tag{8}$$

where Ω is a Boolean operation such as Union, Intersection, or Difference. In addition, $\mathbf{T}_i$ and s_i have the same geometric meaning as those appeared in (7). In our system, the Boolean operation information will be stored in a tree structure in order to speedup the data query.

3.3.3 Non-uniform Knot Distribution

The use of uniform knots to model sophisticated objects may result in a extremely large number of knots and control points. This will lead to information redundancy and deformation difficulty as many degrees of freedom must be employed in any localized small region. However, the use of non-uniform knot sequences affords additional shape control and the modeling of a much larger class of shapes than what the uniform knot vectors can offer. Furthermore, the use of non-uniform B-splines can overcome certain modeling difficulties associated with uniform B-splines. For example, it is almost impossible for B-splines with uniform knot vectors to interpolate highly unevenly spaced data points without the unwanted scenario of oscillations or loops [2].

Our system allows users to specify a non-uniform vector during the initialization phase of the object design session. In addition, users could insert more knots anywhere into current knot vector at any time after the sculpting manipulation is underway. When new knots are inserted, the system will generate corresponding control coefficients and the sculpted object will be reevaluated upon the refined knot vector. In our system, the knot spacing is proportional to the distances of the data points:

$$\frac{\Delta_i}{\Delta_{i+1}}=\frac{\|\Delta x_i\|}{\|\Delta x_{i+1}\|}$$

where Δ_i represents the spatial difference between the (*i+1*)th knot and *i*th knot, Δx_i represents the spatial difference between the (*i+1*)th data point and *i*th data point. Thus, the underlying model represented by volumetric implicit functions is essentially a non-uniform B-spline.

Through the different combination of these three techniques, our system could offer users a large array of modeling operations and enhance the already-powerful shape variation of implicit B-splines with the additional flexibility in a hierarchical fashion.

4. System Description

The sculpted object of an implicit B-spline function is discretized into a voxel raster in our system for rendering purpose. Every voxel contains a scalar value, called density value, sampled at a grid point. The volumetric implicit function described in Section 3 is employed to assign the density value to the sample points to indicate if the location has material. The function will be used to formulate the density distribution over the 3D working space and represent the sculpted object by a given level set. Fig.1 (see the color page) shows voxel maps in 2D space and 3D space, respectively.

This voxelmap defines a function, where the solid particles (colored in red) denote locations in which material exists and the empty particles (colored in gray) denote locations in which there is no material. Although we use binary material distribution in Fig.1 to illustrate the concept, however, in our system the characteristic function is **not** a binary function, rather it is a continuous function.

When a sculpting tool is used to sculpt the object, the density values of the working space inside the tool volume will be modified correspondingly. Then the system will reconstruct the volumetric implicit function to represent the new, modified object undergoing deformation. By using local Marching Cubes technique [12][21], the isosurface of the object could be displayed interactively.

The haptic interface of our system allows users to reach toward an object, feel the physical presence of its shape, and sculpt it with force feedback. Through the use of many haptic tools available in our system, users can obtain both intuitive feeling and better understanding of the virtual sculpting. The feedback forces are computed directly based on the object representation.

4.1 Octree-based Data Structure

Since the sculpted object is discretized in a voxel raster, usually there are many homogeneously empty regions outside the object of interest. If those regions could be quickly separated from the sculpting region, it will significantly reduce the memory consumption and speed up the volume rendering and modeling tasks. Therefore, an octree-based data structure is employed in our system, similar to the scheme used in [11].

The working space is recursively subdivided until either the subdivided volume is empty, or the subdivision has reached a pre-defined maximal subdivision depth. In the first case, the subdivided volume is an empty leaf node, while the second situation means that the current location is not empty and the material property at that location should be recorded. Every time the sculpted object is modified by a sculpting tool, the octree data structure could locate where the modification is performed and only needs to locally update the volumetric implicit function for efficiency purpose. Our system uses Marching Cubes technique to render the isosurface of the sculpted object. This local update property can speed up the Marching Cubes rendering by only conducting the reevaluation task of the modified parts.

4.2 Volume Sculpting

4.2.1 Tool Modeling

Tools are represented by any 3D implicit function $w_0 = G(x, y, z)$. It is easy to determine whether a location is inside the tool volume by simply evaluating the function. In order to prevent object spatial aliasing, a filtering operation must be used inside the tool volume. The filtering algorithm used in our system is similar to the one in [4][9]. Given a location (*x, y, z*), the shortest distance from (*x, y, z*) to the boundary of the tools is computed using the evaluation function. Then this shortest distance is used to filter the density values at the location (*x, y, z*). Here we use a linear filter. The minimal density value is assigned to the boundary and the maximal one is assigned to the center of the tool. The density values at the intermediate locations are linearly interpolated. So the density value at (*x, y, z*) is proportional to the shortest distance to the boundary. Later we will explain how to further generalize this concept in haptic interface to obtain realistic force feedback.

4.2.2 Tool-Object Interaction

When users assign a sculpting tool to a new location, the tool is mapped to the coordinate system, which contains the sculpted object. The boundary box of the tools is then computed. And the density values at the locations inside the tool volume are modified as described in Section 4.2.1. If the tool is to add material, those density values should be greater than the object iso-value. If the tool is to remove material, those density values should be less than the iso-value. After this initial modification on material distribution, we have to reconstruct the volumetric implicit function of B-splines according to the new density distribution. Currently, our system only allows single operation to modify exactly one patch at any time during the session of volume sculpting. So only control coefficients that belong to one B-spline patch need to be modified at every time of sculpting. Real-time performance with realistic haptic feedback can be easily achieved. The mathematics of B-spline manipulation is formulated as follows:

$$(\mathbf{B} \otimes \mathbf{C} \otimes \mathbf{D})\mathbf{p}_{new} = \mathbf{s}_{new} \qquad (9)$$

where $\mathbf{s}_{new}$ represents the new density distribution over the sculpted patch region and $\mathbf{p}_{new}$ are new control coefficients, **B**, **C**, **D** are sampling matrices of basis functions as shown in Section 3.1. Because of the local support property of B-splines, only a very small subset of the control coefficients needs to be modified. Hence, we only need to solve this system of linear equations within the tool sculpting region. Therefore, (9) can be further simplified into:

$$(\mathbf{B}'\otimes\mathbf{C}'\otimes\mathbf{D}')\mathbf{p}_{\text{mod}} = \mathbf{s}_{\text{mod}} \tag{10}$$

where $\mathbf{s}_{\text{mod}}$ and $\mathbf{p}_{\text{mod}}$ only come from the modified region. **B**', **C**' and **D**' are small sets of the original basis matrices, which are corresponding to the local modified region.

Now, in essence the problem of volumetric sculpting is equivalent to a typical data fitting application: Given a set of points (x_i, y_j, z_k) in the parametric domain, and the density value d_{ijk} at every point, find the best possible solution that fits the data set either through interpolation (when one unique solution exists) or approximation (when the system becomes over-constrained). Usually the number of control coefficients is less than the hardware-permitted resolution of 3D working space. So we employ the Mean Square Error for data approximation:

$$MSE = \frac{1}{LMN}\sum_{i=1}^{L}\sum_{j=1}^{M}\sum_{k=1}^{N}(d_{ijk} - s(x_i, y_j, z_k))^2 \tag{11}$$

where *LMN* represents the total number of data points that have been modified using certain sculpting tools. Therefore, it is necessary to seek a function *s*(*x* ,*y* ,*z*) that minimizes the mean square error. It is convenient to use matrix algebra to symbolically formulate the solution to the preceding problem. Using the matrix forms, the Mean Square Error can be written as:

$$MSE = \frac{1}{LMN}\left[\mathbf{d} - (\mathbf{B}'\otimes\mathbf{C}'\otimes\mathbf{D}')\mathbf{p}_{\text{mod}}\right]^2 \tag{12}$$

where $\mathbf{d}$ is a vector of density values whose elements are d_{ijk} . Differentiating with respect to the elements of $\mathbf{p}_{\text{mod}}$, and setting the derivative to zero leads to the solution:

$$\mathbf{p}_{\text{mod}} = \left[(\mathbf{B}'\otimes\mathbf{C}'\otimes\mathbf{D}')^T(\mathbf{B}'\otimes\mathbf{C}'\otimes\mathbf{D}')\right]^{-1}(\mathbf{B}'\otimes\mathbf{C}'\otimes\mathbf{D}')^T\mathbf{d} \tag{13}$$

which is equivalent to the least-square fitting.

After the new control coefficients are generated, the system uses the Local Marching Cubes algorithm to render the modified part to generate the new isosurface of the deformed object.

4.3 Haptic Feedback

In order to enhance the realism of the virtual sculpting, our system offers haptic interactions, which can give users a realistic feel of the virtual objects. Thus, users can gain a richer understanding of their sculpted model. Our work significantly extends the notion of simply touching compliant objects (i.e., haptic rendering) to interactively and directly sculpting of virtual solids (i.e., haptic modeling).

From the standpoint of volume sculpting, the following problems must be addressed in order to provide meaningful force feedback for haptic interaction:

- Force computational rate: the computational rate must be high and latency must be low. Inappropriate values can cause an improper feel of the virtual environment.
- Generation of contacting forces: this creates the "feel" of the object. Contacting forces can represent the stiffness of the object, damping, friction, surface texture, etc.
- Fast data modification and rendering: this could make the sculpting operation consistent with the haptic force feedback.

Since the sculpting could only be performed within a small region at every time step, it is natural to only allow the computation of haptic interactions to occur within a localized region, in order to meet the high frequency requirement set by the haptic device and make the sculpting consistent with the force feedback. In addition, oftentimes users' meaningful sculpting operations would not exceed the region limitation within a very tiny time step along time axis (e.g., only 1~2 ms in our system). So this assumption is reasonable and does not introduce any limitations in our volume sculpting system. In our system, we use a point contact force model [8]:

$$\vec{F} = R(\vec{V}) + S(\vec{N}) \tag{14}$$

where $\vec{V}$ is moving speed of a contacting point, $R(\vec{V})$ is a damping force that tends to resist motion along the opposite direction of the contacting point's movement, $S(\vec{N})$ is a stiffness force along the normal of the contacting point ($\vec{N}$). $\vec{F}$ is the feedback force to users, which is equal to the sum of the motion damping force and stiffness force. The force calculation should be very fast to meet the PHANToM update rate (i.e., greater than 1kHz). Otherwise, users would have uncomfortable feelings such as buzzing during the haptic interaction, hence, destroying the purpose of using haptics to augment realism. As we described in Section 4.2.1, the density distribution in our model is proportional to the distance map. So it is natural for us to use the density field instead of the distance field to calculate the force. The motion damping force and stiffness force are calculated, respectively, as follows:

$$R(\vec{V}) = -\vec{V}f_r(d) \tag{15}$$

$$S(\vec{N}) = \frac{\vec{N}}{\left|\vec{N}\right|}f_s(d) \tag{16}$$

where d is a density value, f_r and f_s are transfer functions, which map density values to force magnitudes. The transfer functions f_r and f_s that we are currently using in our environment are as follows:

$$f_r(d) = a(d_{new} - d) + b \tag{17}$$

$$f_s(d) = k(d_{new} - d) \quad (18)$$

where d is current density value at (x, y, z), d_{new} represents the new density value at $(x+\Delta x, y+\Delta y, z+\Delta z)$, which is the very next time step. a, b and k are control variables which can be interactively set up by users. Using different transfer functions, we could let users feel different force effects, increasing the flexibility of our haptic interaction.

In our system, the density distribution over the 3D working space is represented as a continuous volumetric implicit function of B-splines. This property can help to avoid the discontinuous force feedback, which leads to unrealistic feelings such as buzzing. Another advantage for integrating haptic interaction with implicit functions is that it is much easier to compute the contacting point and determine if the contacting point is inside the object to be sculpted. The density value at any location could be given by simply evaluating the volumetric implicit function. Therefore, the damping force and stiffness force could be computed efficiently to satisfy the high update rate of haptic interaction,

$$R(\vec{V}) = -\vec{V} \bullet (a(F(x+\Delta x, y+\Delta y, z+\Delta z) - F(x,y,z)) + b) \quad (19)$$

$$S(\vec{N}) = \frac{\vec{N}}{|\vec{N}|} \bullet k \bullet (F(x+\Delta x, y+\Delta y, z+\Delta z) - F(x,y,z)) \quad (20)$$

where F is obtained from (7) or (8). The normal $\vec{N}$ at (x, y, z) can be computed analytically as $(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z})$.

5. Implementation

Our system is implemented on a Microsoft Windows NT PC with a 550MHz CPU and 512 MB RAM. A PHANToM 1.0 3D Haptic input/output device from Sensable Technologies is employed to provide natural and realistic force feedback. The entire system is written in Microsoft Visual C++ and the graphics rendering component is built upon OpenGL. Fig. 2 (see the color page) shows the system interface.

When using haptic tools, to reduce the latency and maximize the throughput, we resort to a parallel technique that can multithread the haptics, graphics, and sculpting processes with weak synchronization. This technique leads to the possible performance improvement and ultimately, the parallel processing of haptic sculpting given the high-end multi-processor environment. Therefore, our system is readily available in many different configurations. Fig. 3 shows the structure of the multithreads, where thick arrows represent data flow and thin arrows represent control flow.

The haptic loop is implemented in a single thread. It maintains the haptic refresh rate which is no less than 1KHz. This requirement is critical to the realistic feedback of haptic interaction. If the update rate is below the threshold of 1KHz, users would feel uncomfortably. In our system, the haptic thread has the highest priority.

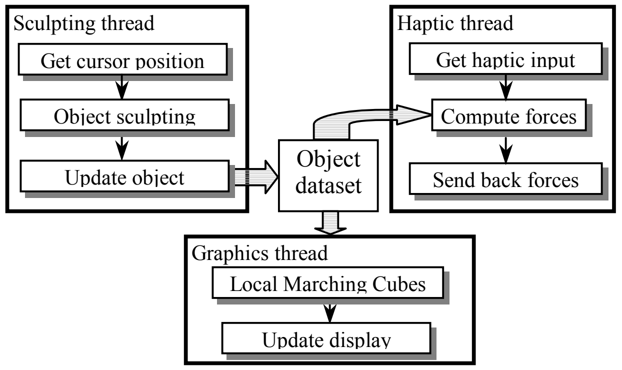


Fig. 3 The structure of multithreads

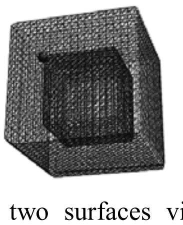
The object sculpting loop is implemented in another thread. It controls the object sculpting. In order to keep up with haptic update rate, any sculpting operation within one time step is limited to a small region. As we specified in Section 4.3, usually users' sculpting operations would not exceed this limited region within a tiny time step.

The graphics loop is developed to handle the rendering of volumetric objects. The rendering task makes use of local Marching Cubes algorithm and only update the very small region in order to achieve interactive speed and make graphics display consistent with sculpting operation and force feedback.

6. Interactive Sculpting Toolkits

Our system offers several haptic tools such as *haptic iso-surface feeling, haptics-based probing,* and *haptics-based driller*.

Besides feeling the boundary surface of a volumetric object, users can choose any iso-value from the allowable range of the volumetric implicit function and the system generates corresponding iso-surfaces quickly. The figure on the left shows two different iso-surfaces with the wireframe display mode in order to make two surfaces visible at the same time. Users can feel different iso-surfaces of the sculpted object by moving the cursor and navigating over those iso-surfaces. This tool allows users to examine the smoothness of objects' surface and the interior structure tactilely.



Through the use of the contacting force model described in Section 4.3, our system can afford users to feel the tiny difference of an object's stiffness (or density) while users move the cursor inside the object. When active, the probing tool exerts a force on the user's finger proportional to the local density values within a given radius of the tool.

Using the haptics-based driller, user can make a drill to the object along any direction. In this process, users could feel the realistic force coming from the object's stiffness and the motion resistance. The driller could be any kind of shape such as spheres, cubes or stars.

For tool operations, Fig. 4 (see the color page) shows a number of toolkits including *sphere-based carving and addition tools, cylinder-based carving and addition tools, rectangle-based carving and addition tools, torus-based carving and addition tools, chisel tools, copy and composing tools, squirt tools, inflation and deflation tools, moving and bending tools*, etc. Next we explain some tool configurations and their operations.

When using copy and composing tools to build up a complicated scene with the same object, users can define a number of trivariate patches at any locations and along any directions. The collection of those patches is based on the union operation for CSG models. Through the use of the simple copy operation, the patch coefficients can be duplicated from one region to another region of interest, a set of similar objects of the same geometry could be easily created. In Fig. 4 where *copy and composing operations* were undertaken, 3×3 patches were created parallel to each other in the working space. When the chair sculpting was completed inside one patch, the coefficients of that patch were copied and loaded in other 8 patches to create the scene.

The inflation and deflation tools cause a localized region to grow or shrink. Users can interactively define a region. When using a deflation tool, the coefficients of the scalar trivariate implicit function inside the region decrease in order to shrink the specified part of the object. For the inflation tool, those coefficients should be increased instead. In Fig. 4 where *inflation and deflation operations* were undertaken, the soccer player's head was inflated and one of his arms was deflated.

The moving tool moves a selected region of the coefficients to other locations. The bending tool deforms a selected region of coefficients to new positions. In Fig. 4 where *moving and bending operations* were undertaken, we first moved both arms up through the use of the moving tool, and then, bended the player to get the second gesture.

7. Experimental Results

We have developed a novel modeling system for haptic sculpting of the volumetric implicit function based on non-uniform B-splines. The implicit solid can be generated with a variable number of control coefficients and with variable sampling rates. We have conducted a large number of experiments and recorded the running time for the sculpting of volumetric implicit functions. The experiments are based on a working space sampled at $128 \times 128 \times 128$. The tool size is given as the number of data points that the tool affects. The results are detailed in Table 1.

Table 1: Run time of Tool-Object interaction

Control coefficient resolution	Tool size	Update time (ms)
$32 \times 32 \times 32$	$10 \times 10 \times 10$	1
	$20 \times 20 \times 20$	9
	$40 \times 40 \times 40$	69
$64 \times 64 \times 64$	$10 \times 10 \times 10$	1.5
	$20 \times 20 \times 20$	11
	$40 \times 40 \times 40$	92
$128 \times 128 \times 128$	$10 \times 10 \times 10$	2
	$20 \times 20 \times 20$	20
	$40 \times 40 \times 40$	151

Within our implicit function modeling framework and without using any other external resource, we have created several interesting objects and scenes from scratch. Fig. 5 (see the color page) shows a series of actions of a soccer player. They were sculpted with $64 \times 64 \times 32$ uniform knots and control coefficients. Fig. 5(a) shows a standing one. We shall use this example to explain how to sculpt an object using our system. We began with a cubic block. By carving and haptically drilling the cubic block several times, a rectangular body was created. The neck was sculpted using cylinder-based addition. The head was placed on top of the neck using a sphere-based additive tool. By way of a rectangular tool, the shoulder part was sculpted. The two arms were created using cylinder-based addition. The two legs were obtained in a similar fashion. The feet were sculpted by adding rectangular materials and sculpting with sphere-based tools and the haptics-based driller. Other motions of the soccer player are all based on the initialized model. Through the use of moving and bending operations, the animated sequences of models were subsequently created. Fig. 6 (see the color page) shows several characters mounted on a rectangular stone. The working space contains a single patch with $64 \times 64 \times 32$ non-uniform knots and $64 \times 64 \times 32$ control coefficients. The region that contains all characters has much more knots and control coefficients than the flat region where no deformation is undertaken.

Fig. 7, Fig. 8 and Fig. 9 (see the color page) show three scenes, which were sculpted entirely using our system (without resorting to any other external resource) and rendered using the commercial software of POV-Ray. For example, Fig. 7 shows a cartoon train running in a desert environment. The train was sculpted with nine $64 \times 64 \times 64$ patches. One patch was for sculpting the body of the train. And other eight patches were used for sculpting the eight wheels. The railroad was sculpted in a $64 \times 64 \times 64$ patch. Every cactus was sculpted in a $32 \times 32 \times 32$ patch. The collection of the patches in the working space was based on hierarchical organization and union operation.

8. Conclusion

We have presented a novel haptics-based volumetric sculpting environment that employs trivariate scalar non-uniform B-splines as underlying representation. All the volumetric objects sculpted in our modeling system are characterized by piece-wise implicit functions. We have proposed a new approach that unifies implicit functions and parametric representations within a single haptics-based sculpting system. We have developed a large variety of algorithms and toolkits that afford designers the mechanism of interactive and direct manipulation of implicit solids in real-time, augmented by a realistic and intuitive haptic interface. To facilitate multiresolution editing and direct control on different levels of details, we have also incorporated three popular modeling techniques: hierarchical B-splines, CSG-based functional composition, and knot insertion into our environment, making our novel implicit modeling techniques even more powerful and flexible to handle both complicated geometry and arbitrary topologies.

Our experiments have demonstrated that our algorithms and direct editing techniques based on nonuniform B-spline implicit functions can not only overcome the existing disadvantages associated with conventional modeling of implicit functions, but realize all the potentials exhibited in implicit functions in visual computing fields as well. More importantly, the powerful 3D haptics-based interface of our system is more intuitive and natural than conventional 2D mouse-based interfaces, making it possible for our implicit function modeling system to appeal to a spectrum of users ranging from highly trained engineering designers, computer professionals, artists, to even computer illiterates. Our sculpting system permits designers to create real-world, complicated models in real-time. Finally, the novel modeling techniques and their haptics-based design principle are extensible to the design of arbitrary implicit functions.

Acknowledgements

The authors wish to thank Kevin T. McDonnell and Wei Hong for providing some of the code used in implementing this system. This research was supported in part by the NSF CAREER award CCR-9896123, the NSF grant DMI-9896170, the NSF ITR grant IIS-0082035, and research grants from Ford Motor Company and Honda America Inc.

References

[1] J. Bloomenthal, B. Wyvill. Interactive techniques for implicit modeling. Computer Graphics, Vol. 24, No. 2, pp 109-116, March 1990.

[2] L. Piegl and W. Tiller. Curve and surface constructions using rational B-splines. Computer-Aided Design, Vol. 19, No. 9, pp 485-498, November 1987.

[3] J. F. Blinn. Generalization of algebraic surface drawing. ACM Trans. On Graphics, Vol. 1, No. 3, pp 235-256, July 1982.

[4] T. A. Galyean and J. F. Hughes. Sculpting: An interactive volumetric modeling technique. Computer Graphics, Vol. 25, No. 4, pp 267-274, July 1991.

[5] A. Raviv and G. Elber. Three dimensional freeform sculpting via zero sets of scalar trivariate functions. In Proc. of 5th ACM Symposium on Solid Modeling and Applications, pp. 246-257, 1999.

[6] T. V. Thompson, D. E. Johnson and E. Cohen. Direct haptic rendering of sculptured models. In Proc. of the 1997 Symposium on Interactive 3D Graphics, pp 167-176, 1997.

[7] K. T. McDonnell, H. Qin and R. A. Wlodarczyk. Virtual Clay: A real-time sculpting system with Haptic Toolkits. In Proc. of the 2001 Symposium on Interactive 3D Graphics, pp.179-190, 2001.

[8] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. In Proc. of the 7th IEEE Visualization '96, pp 197-204, 1996.

[9] S. W. Wang and A. E. Kaufman. Volume sculpting. In Proc. of the 1995 Symposium on Interactive 3D Graphics, pp 151-156, 1995.

[10] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. Computer Graphics, Vol. 22, No. 4, pp 205-211, August 1988.

[11] J. Andreas Barentzen. Octree-based volume sculpting. IEEE Visualization '98, Late Breaking Hot Topics Proceedings, pp. 9-12, 1998.

[12] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. Computer Graphics, Vol. 21, No. 4, pp 163-169, July 1987.

[13] H. Qin. Physics based geometric design. International J. of Shape Modeling, Vol. 2, No. 2&3, pp 139-188, 1996.

[14] T. Sederberg. Techniques for cubic algebraic surfaces. IEEE Computer Graphics and Application, Vol. 10, No. 4, pp 14-25, 1990.

[15] T. Sederberg. Techniques for cubic algebraic surfaces. IEEE Computer Graphics and Application, Vol. 10, No. 5, pp 12-21, 1990.

[16] C. Hoffmann. Implicit curves and surfaces in CAGD. IEEE Computer Graphics and Applications, Vol. 13, No. 1, pp 79-88, 1993.

[17] C. Bajaj and I. Ihm. Algebraic surface design with Hermite interpolation. ACM Transactions on Graphics, Vol. 11, No. 1, pp 61-91, 1992.

[18] J. L. Blechschmidt and D. Nagasuru. The use of algebraic functions as a solid modeling alternative. Advances in Design Automation, B. Ravani Ed., ASME Design Conference, Chicago, IL, pp. 33-41, Sept. 1990.

[19] V. Shapiro. Real functions for representation of rigid solids. Computer Science Tech. Report TR91-1245, Cornell Univ., Ithaca, NY, 1991.

[20] F. Dachille IX, H. Qin and A. E. Kaufman. A novel haptics-based interface and sculpting system for physics-based geometric design. Computer-Aided Design, Vol. 33, No. 5, pp 403-420, 2001.

[21] G. Wyvill, C. McPheeters and B. Wyvill. Data structure for soft objects. The Visual Computer, Vol. 2, No. 4, pp 227-234, 1986.

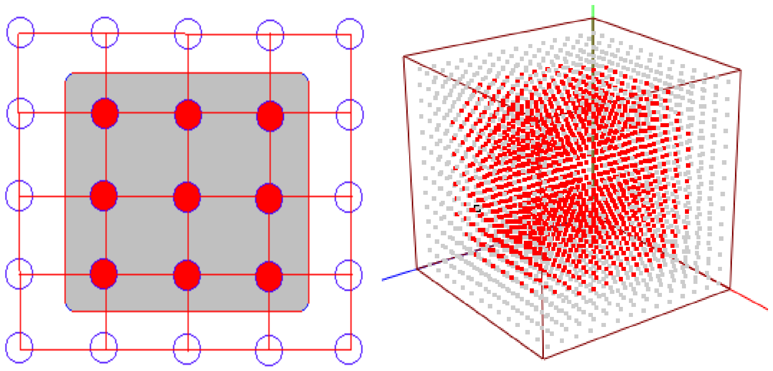


Fig. 1 Voxelmap in 2D and 3D

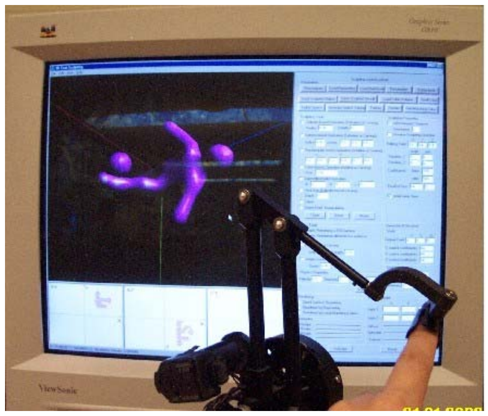
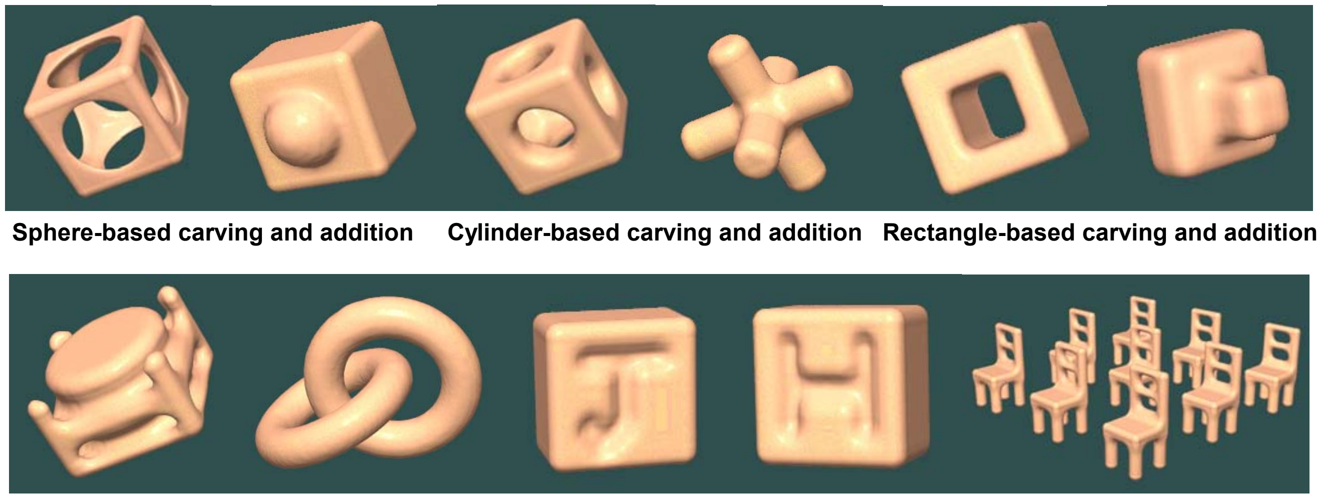
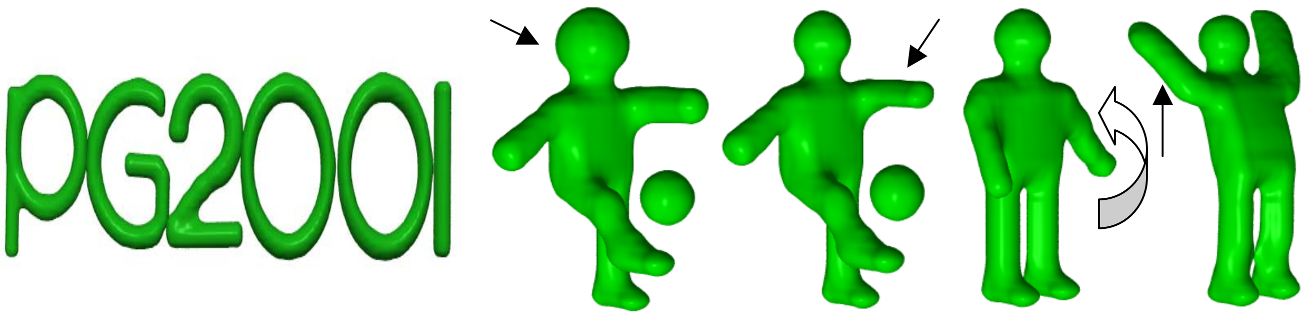


Fig. 2 System Interface comprised of on-screenGUI, a PHANToM device



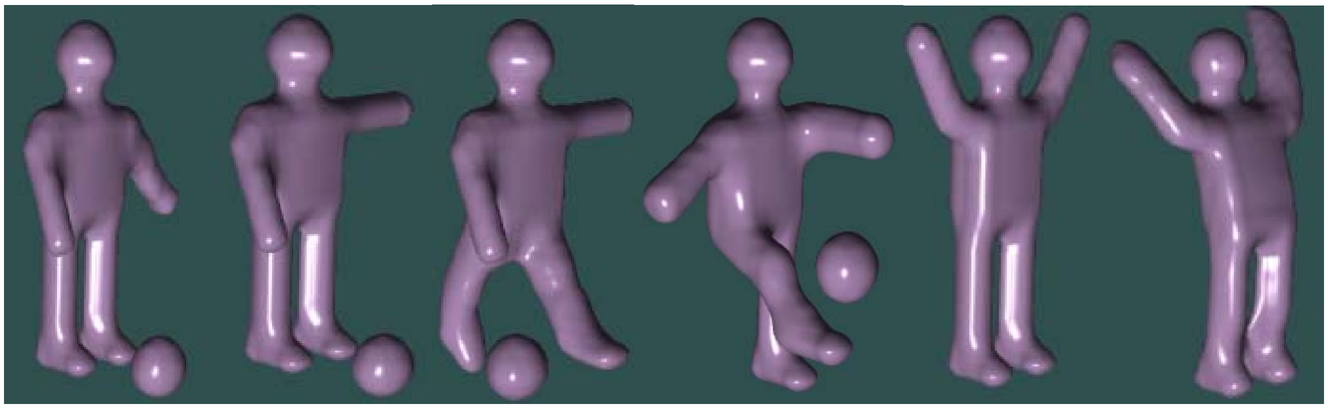
Sphere-based carving and addition Cylinder-based carving and addition Rectangle-based carving and addition

Torus based carving and addition Chisel operation Copy and composing operation



Squirt operations Inflation and deflation operations Moving and bending operations

Fig. 4 A set of typical toolkits and sculpted examples


(a) (b) (c) (d) (e) (f)

(a) Model initialization, (b) Starting position, (c) Moving, (d) Kicking, (e-f) Celebrations

Fig. 5 A motion series of a soccer player and his entire kicking actions (a, b, c, d, e, f)

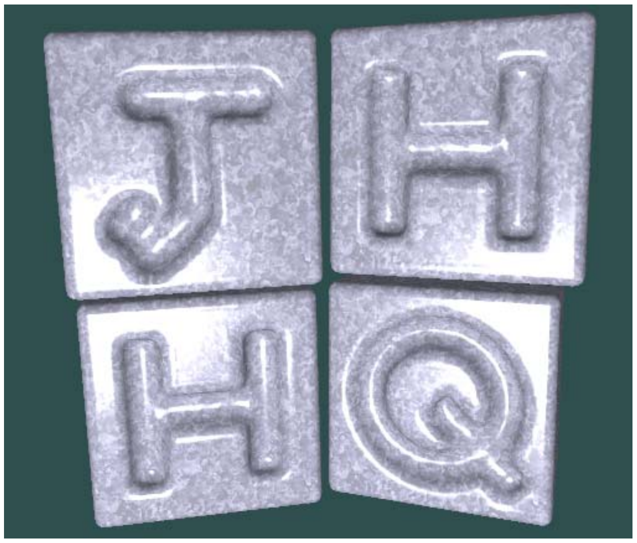


Fig. 6 English letters mounted on stone plates

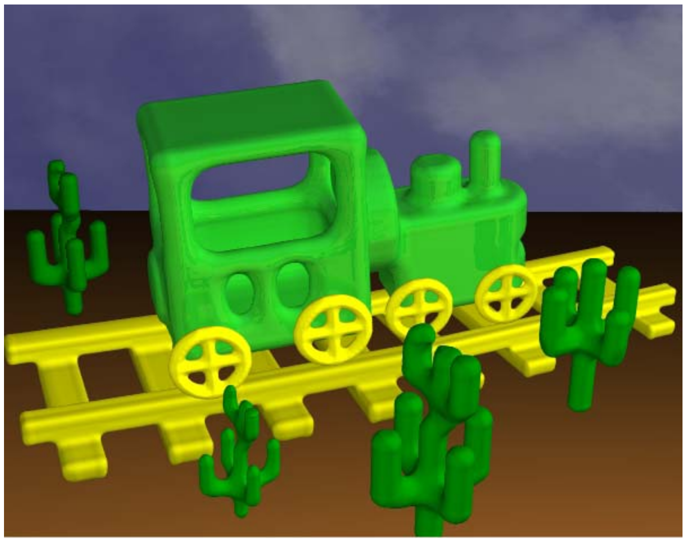


Fig. 7 A Running Cartoon Train

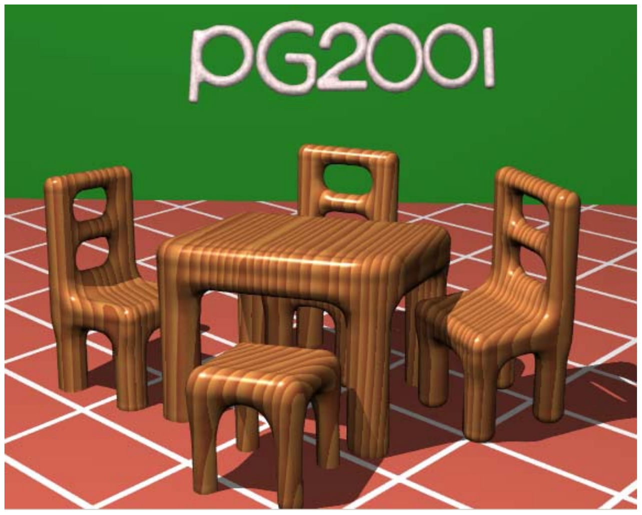


Fig. 8 A corner of PG discussion room

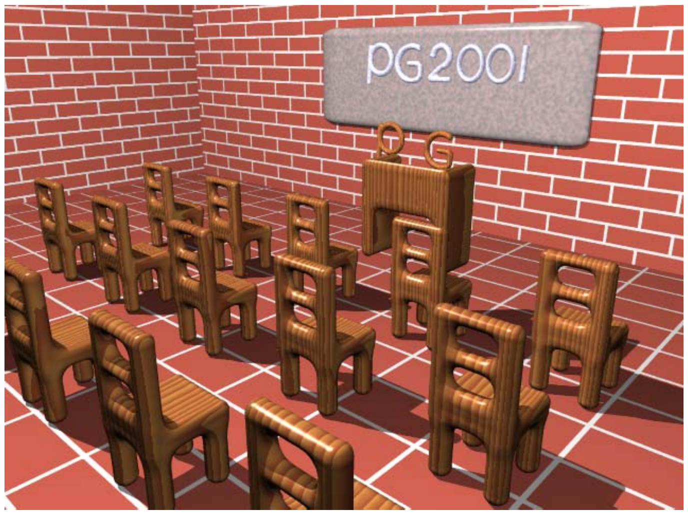


Fig. 9 PG conference room