Department of Computer Science,
University of Otago

Technical Report OUCS-2003-03

Modeling by Deformation Using Swept User-Defined Tools

Authors:

Alexis Angelidis, Geoff Wyvill
Department of Computer Science
Marie -Paule Cani
GRAVIR-IMAG

Status: Rejected by SIGGRAPH'03

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/trseries/

Modeling by Deformation Using Swept User-Defined Tools.

Alexis Angelidis * University of Otago
Geoff Wyvill† University of Otago
Marie-Paule Cani‡ GRAVIR-IMAG§

Abstract

We present a new class of space deformations suitable for interactive virtual sculpture. The artist describes a basic deformation as a path through which a tool is moved. Our tools are simply shapes, subsets of 3D space. So we can use shapes already created as customized tools to make more complex shapes or to simplify the modeling process.

When a tool is moved it causes a deformation of the working shape along the path of the tool. This is in accordance with a clay modeling metaphor and easy to understand and predict. More complicated deformations are achieved by using several tools simultaneously in the same region.

It is desirable that deformations for modeling are 'foldover-free', that is parts of deformed space cannot overlap so that the deformations are reversible. There are good intuitive reasons to believe that our deformations are foldover-free but we have not yet completed a proof.

We have an efficient formulation for a single tool following a simple path (translation, scaling or rotation) and we can demonstrate the effects of multiple tools used simultaneously. The prototype implementation described has been used to create a variety of models quickly and conveniently.

CR Categories: I.3.5 [Computer Graphics]: Object Modeling—Deformations, Geometric Modeling, Curves and surfaces

Keywords: free-form deformation, shape modeling, non-linear transformations

1 Introduction

The process a sculptor uses to create a shape can be regarded as a definition of the shape. From this point of view, a representation such as a NURB or implicit surface is merely an intermediate device between the acts of modeling and rendering. Foley and Van Dam remark, "The user interfaces of successful systems are largely independent of the internal representation chosen" [Foley et al. 1994]. This, surely, is evidence that the representations are inherently unsuitable.

Our thesis is that the primary representation of a model must allow straightforward and intuitive editing by an artist. By intuitive, we mean that the editing operations must work in accordance with a consistent metaphor that is clear to the artist.

Existing mathematical representations are not directly suitable for editing operations, while most existing editing operations are not intuitive according to a suitable metaphor. For most virtual modeling tools, this observation results from the fact that the mathematical representation is strongly linked to the editing operations; for example editing the control points of a NURB patch manually. Space

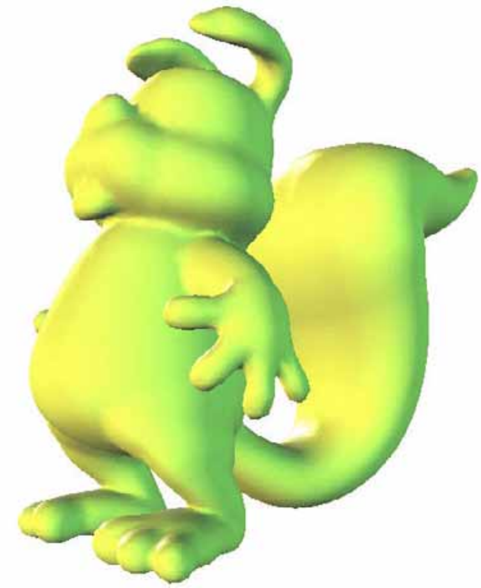


Figure 1: Squirrel character modeled out of an initial ball. The artist modeled only one side, while the other is automatically made at the same time thanks to the simultaneous tool. There are no discontinuities caused by the symmetry.

deformations stand apart from this, and can be used with any mathematical model, including implicit surfaces when the deformation is reversible. However, space deformation has had more success adjusting existing models than with creating entirely new ones, mainly because the deformation operations have not been developed to create a rich set of features. With the exception of [Mason and Wyvill 2001], deformation operations do not prevent surfaces from self-intersecting. This is crucial, since space deformation cannot un-intersect a self-intersecting surface.

We see all these things as obstacles to the creativity of artists. This paper proposes a class of *operations for sculpture* independent of the shape's underlying mathematical model. It can be applied in principle to any standard model. All the examples in this paper, however, are deformations of a single sphere. These deformation operations are specified intuitively as transformations of tools where a tool is any shape. They are continuous (at least C^0 and in most cases C^2). They are local in operation, within some user-defined distance of the tools and most importantly they are foldover-free, preserving the shape's coherency. The remainder of this paper is organized as follows. In Section 2, we discuss the limits of existing techniques. In Section 3 we introduce our new deformations as a class of operations applicable to space in any number of dimensions. In section 4 we develop closed forms for the efficient application of a single tool in a 3-dimensional scene. In Section 5 we present the details required to implement the technique in an interactive modeler. We show our results in Section 6.

*e-mail: alexis@cs.otago.ac.nz
†e-mail: geoff@cs.otago.ac.nz
‡e-mail: Marie-Paule.Cani@imag.fr
§GRAVIR is a joint lab of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

2 Related work

Space deformation provides a formalism to specify any editing operation, by successively deforming the space in which an initial shape S^{t_0} is embedded:

$$S_n = \left\{ \overset{n-1}{\underset{i=0}{\Omega}} f^{t_i \mapsto t_{i+1}}(p) | p \in S^{t_0} \right\}$$

where $f^{t_i \mapsto t_{i+1}} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a deformation of space[1]. The reason why space deformations are independent of the mathematical model of a surface is that they apply to the space in which the model is embedded and can deform regions of space where there is no surface, if required. Note that, as for non-virtual sculpture, the operations do not commute under function composition, $\circ$.

This section reviews existing classes of deformations, organized in three groups according to what we call their *genericity*: the deformations that are not suitable for sculpture and can produce a limited set of shapes, the deformations that can produce a large set of shapes given enough parameters for a few functions $f^{t_i \mapsto t_{i+1}}$, and the deformations that can produce a large set of shapes given enough simple functions $f^{t_i \mapsto t_{i+1}}$.

2.1 Global deformations

[Barr 1984] defines space tapering, twisting and bending transformations via a matrix that is a function of a space coordinate. An interesting result is the proof that the surface normal vector transformation is given by the transformation's Jacobian co-matrix[2]. [Blanc 1994a] generalizes this work to deformations that are functions of more than one space coordinate. [Chang and Rockwood 1994] propose a polynomial deformation that efficiently achieves "Barr"-like deformations and more, using a Bézier curve with coordinate sets defined at control points. [Mikita 1996] extends this method to triangular Bézier surfaces. A restriction of these methods is the initial rectilinear axis or planar surface. [Crespin 1999] proposes to use a technique based on recursive subdivision in order to use an initially deformed tool. His deformations do not prevent the shape from self-intersecting.

All these deformations are global, and can be handled easily by the user because they have few control parameters. However because of their non-locality, they are not suitable for surface sculpture.

2.2 Many parameters, few functions

[Sederberg and Parry 1986] introduced Free-Form Deformations (FFDs) which allow continuous space deformations with multiple points transformed. The method involves defining the lattice of a Bézier volume, and then moving its control points. The embedded space is then smoothly deformed by interpolating the control point coordinates. A major restriction of FFD is the regularity of the grid. [Coquillart 1990] and [Blanc 1994b] extend this work for a non-regular lattice. Still, a problem is that a correspondence between the edited shape and the lattice has to be done manually. [Hsu et al. 1992] propose a way of doing direct manipulation of a single point or multiple points in space with FFD. Still, the regularity and fixed size of the grid along with computing costs restrict its utility.

[MacCracken and Joy 1996] use subdivision volumes, allowing arbitrary lattices. Customizing the lattice onto the shape is however tiresome.

[Borrel and Bechmann 1991] generalize this to arbitrarily positioned control points, where no lattice is needed: the shape is non-linearly projected into a space of higher dimension; the deformation is a linear projection back onto $\mathbb{R}^3$ (or $\mathbb{R}^4$ for controlling animation). In *Scodef* (Simple Constrained Deformation) instead of just control points, [Borrel and Rappoport 1994] use also control areas, and the control features can be assigned orientations to perform twists. These methods define the deformation as a projection of a built space of higher dimension. Issues arise for controlling the deformation, because the pseudo-inverse computation involved does not always behave intuitively.

[Moccozet and Magnenat-Thalmann 1997] propose another approach to get rid of lattice regularity. They use a method developed by [Farin 1990] to define a continuous parametrization over the Sibson coordinates. Still, control points have to be placed manually, and computing the Sibson coordinates is expensive and difficult.

These methods can achieve very complex deformations but at a cost: either they are computationally intensive, or the effort required from the user is high.

2.3 Many functions, few parameters

Another approach to space deformation is the definition of simple deforming tools. In this framework a shape is modeled by combining many simple deformations.

The first introduced surface editing tool that looks like space deformation is *warping*, by [Parent 1977]. Vertices within a distance (discrete number of edges) to a selected vertex are *warped*, that is, a weighted transformation of the selected vertex is applied to them.

[Decaudin 1996] proposes a tool that allows modeling a shape by iteratively adding or removing the volume of simple $3D$ shapes (eg. sphere, ellipsoid). These deformations do not allow bending or twisting shapes.

[Wyvill et al. 1996] introduce *feature modeling*, local space deformations applied to a parametric surface. A translation, twist or bend is applied around a point within a limiting ellipsoid. The deformation has a second-order continuity. The interesting point is that intuitive editing is performed within the scene's space, as opposed to the surface's parametric space. Also, it shows that a space deformation tool can easily be turned into a surface editing tool.

[Kurzion and Yagel 1997] present deformations they call *ray deflectors*. An inverse deformation can be computed, thus allowing to deform the rendering instead of the shape. Their tool can translate, rotate and scale space, contained in a sphere, locally and smoothly: the deformation is however interpolated only by the center point of the tool. Moreover, they define a discontinuous deformation that allows one to *cut* space.

[Singh and Fiume 1998] introduce *wires*, a geometric deformation technique which can easily achieve a very rich set of deformations with curves as control features; however the deformation does not prevent the object from self-intersecting, and the only features that can remain undistorted are curves.

[Crespin 1999] introduces the IFFD (Implicit Free Form Deformation). Note that though it is called implicit, the deformation applied to an embedded shape is explicit: the field generated by a skeleton modulates affine transformations. He also proposes two ways to combine many transformations simultaneously.

[Mason and Wyvill 2001] introduce *blendeforming*, using reversible (foldover-free) local deformations that can specify the deformation by controlling the position of a point or the

[1] We denote $\overset{n-1}{\underset{i=0}{\Omega}} f^{t_i \mapsto t_{i+1}}(p) = f^{t_{n-1} \mapsto t_n} \circ \cdots \circ f^{t_0 \mapsto t_1}(p)$

[2] Matrix of the cofactors

control points of a curve.

The modeling philosophy of all these methods is to apply simple deformations one after the other as a sculptor would do. In the zone deformed by the tool, the portion of the shape that is undistorted is not or can hardly be controlled.

A drawback of all the methods above resides in the relation between the deformation and the clay: either it is manually defined by the user, or making the correspondence is the bottleneck of the algorithm. As a result it is difficult to push or pull a particular part of the surface predictably.

3 Definitions and algorithm

Before describing how we perform the general deformations, we define the subsets and the matrix notation we use. Then, we introduce how we handle foldover-free deformation with a single tool. We conclude this section with the complete deformation expressions.

3.1 Terminology and notations

We call *tool* j a scalar field $\phi_j^t : p \in \mathbb{R}^n \mapsto [0,1]$ (the superscript t denotes time). To specify tools easily, we use the following C^2 piecewise polynomial function $\mu_j : \mathbb{R} \mapsto [0,1]$ of a distance field $d_j^t : \mathbb{R}^n \mapsto \mathbb{R}$:

$$\mu_j(d) = \begin{cases} 0 & \text{if} \quad \lambda_j \leq d \\ 1 + (\frac{d}{\lambda_j})^3(\frac{d}{\lambda_j}(15 - 6\frac{d}{\lambda_j}) - 10) & \text{if} \quad d < \lambda_j \end{cases}$$

We define $\phi_j^t = \mu_j \circ d_j^t$, as shown in Figure 2. Note that each tool has a different coating thickness λ_j. For the following, the minimum of its derivative will be needed:

$$\min(\frac{\delta \mu_j}{\delta d}) = \frac{-1.875}{\lambda_j}$$

where this field is local, and C^2 where the distance is smooth within a λ_j-neighborhood of the tool. We distinguish three zones:

- the *inside* T_j^t, where $\phi_j^t(p) = 1$.
- the *coating* K_j^t, where $\phi_j^t(p) \in (0,1)$.
- the *outside* O_j^t, where $\phi_j^t(p) = 0$.

We represent a tool's transformations by keyframes $(t_0, \ldots t_n)$, with the corresponding matrices (the transformations we consider are 4×4 matrix products of translations, uniform scaling and rotations):

- absolute transformations $M_j^{t_i}$, used to compute the distance to the tool.
- relative transformations $M_j^{t_i \mapsto t_{i+1}} = M_j^{t_{i+1}}(M_j^{t_i})^{-1}$.

In order to compute the transformed scalar field $\phi_j^{t_i}$ at t_i, we need the transformed distance:

$$d_j^{t_i}(p) = det(M^{t_i})^{\frac{1}{3}} d_j^{t_0}((M^{t_i})^{-1}p)$$

Loosely speaking, the scalar $\phi_j^{t_i}(p)$ is the *amount of deformation* of tool j at time t_i at p. To blend or to compute fractions of deformations, we use the operator $\odot$ and $\oplus$ defined by M. Alexa [Alexa 2002][3]. The naïve deformation of a point with a single tool would be:

$$f^{t_i \mapsto t_{i+1}}(p) = \phi_j^{t_i}(p) \odot M_j^{t_i \mapsto t_{i+1}} p$$

The latter does not prevent however the space from folding onto itself.

[3] $\odot$ and $\oplus$ for matrices behave like $\cdot$ and $+$ for scalars.

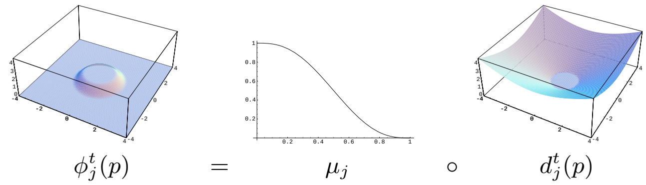


Figure 2: $2D$ scalar field for a disk of radius 1, with $\lambda_j = 1$.

3.2 Single tool and foldover issue

We introduce our deformations with a single tool j to underline how we solve the foldover issue. Suppose for instance that $M_j^{t_i \mapsto t_{i+1}}$ is a translation of length larger than the coating thickness λ_j; it would map points from $T_j^{t_i}$ onto points of $O_j^{t_{i+1}}$, folding space onto itself, as shown in the left of Figure 3. However, if we decompose the transformation into a series of s small enough transformations, foldovers can be avoided, as shown in the right of the figure. If there was a closed form expression for the deformation when $s \to +\infty$, we would not need to bother with stating a foldover-free condition. In practice, computing this closed form is impossible, and taking the smallest number for which the deformation is foldover-free is enough. We will therefore define a lower bound to s, and create equally spaced sub-keyframes $\{\tau_0, \ldots \tau_s\}$, such that $\tau_0 = t_i$ and $\tau_s = t_{i+1}$.

Let us simply note the relative transformation $M_j = M_j^{t_i \mapsto t_{i+1}}$, as for the rest of the paper we will focus on a single interval $[t_i, t_{i+1}]$. The in-between absolute transformations are:

$$\left(\frac{k}{s} \odot M_j\right) * M_j^{t_i}, \; k \in [0, s-1]$$

and the in-between relative transformations are all the same:

$$\frac{1}{s} \odot M_j$$

We have shown in appendix A that the following is a lower bound to the required number of steps:

$$-\min(\frac{\delta \mu_j}{\delta d}) \max_{l \in [1,8]} ||\log(M_j)p_l|| < s \qquad (1)$$

where $p_{l \in [1,8]}$ are the corners of a bounding box around $K_j^{t_i}$.

3.3 Deforming with many tools

Applying more than one tool at the same time at the same place has applications such as shown in Figure 1, where we modeled a symmetric object by applying the same tool symmetrically with respect to a plane. It is also used when defining a deformable tool made of several rigid parts such as a hand, and it allows the surface to be pinched. This could be useful later when we extend our method to incorporate topology changes.

Let us define n tools sharing the same keyframes t_i, with each tool associated with a scalar field $\phi_j^{t_i}$. Each tool is also associated with a relative transformation $M_j^{t_i \mapsto t_{i+1}}$ between keyframes t_i and t_{i+1}. The following expression provides a piecewise smooth[4] combination of all the transformations at any point p in space (we note $\phi_j = \phi_j^{t_i}$ and $M_j = M_j^{t_i \mapsto t_{i+1}}$ to simplify the expression):

$$\begin{cases} I & \text{if } \sum_{k=1}^n \phi_k(p) = 0 \\ \bigoplus_{j=1}^n \left(\left(\frac{(1 - \prod_{i=1}^n (1 - \phi_i(p)))}{\sum_{k=1}^n \phi_k(p)} \phi_j(p) \right) \odot M_j \right) & \text{if } \sum_{k=1}^n \phi_k(p) \neq 0 \end{cases}$$

[4] as smooth as the ϕ_i.

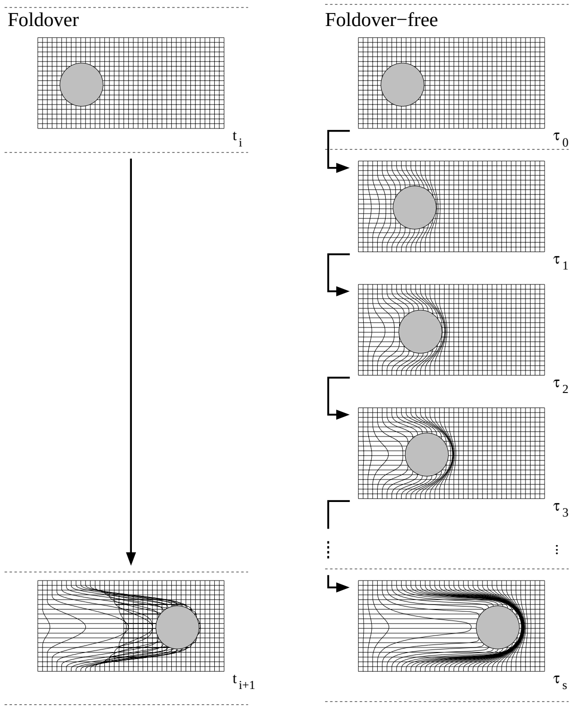


Figure 3: $2D$ illustration of our solution to foldovers. Left: the deformation maps space onto itself. Right: the deformation is decomposed into small foldover-free steps.

This expression can be computed more efficiently:

$$\boxed{\begin{cases} I & \text{if } \sum_{k=1}^n \phi_k(p)=0 \\ \exp \frac{1-\prod_{i=1}^n(1-\phi_i(p))}{\sum_{k=1}^n\phi_k(p)} \sum_{j=1}^n\big(\phi_j(p)\log(M_j)\big) & \text{if } \sum_{k=1}^n\phi_k(p)\neq 0\end{cases}} \quad (2)$$

where:

- $\frac{1}{\sum_{k=1}^n \phi_k(p)}$ is required to produced a **normalized** combination of the transformations. This prevents for instance two translations of vector $\vec{d}$ producing a translation of vector $2\vec{d}$, which would send a point far away from the tools (the problem is also discussed in [Singh and Fiume 1998]).
- $1-\prod_{i=1}^n(1-\phi_i(p))$ **smooths** the deformation in the entire space, required in the boundary between $K_j^{t_i}$ and $O_j^{t_i}$. Indeed, smoothness would be lost if we only used the above normalization.

An interesting point about this expression is that when compared to the solution proposed by [Crespin 1999], there is no extra scalar field required (only ϕ_j) to ensure continuity in $\mathbb{R}^n$. The following expression is a lower bound to the required number of steps, generalizing the single tool condition (see appendix A):

$$\boxed{-\sum_j \min\left(\frac{\delta\mu_j}{\delta d}\right)\max_{l\in[1,8]}\left|\left|\log M_j p_{l_j}\right|\right| < s} \qquad (3)$$

where $p_{l_j\in[1,8]}$ are the corners of the bounding box around $K_j^{t_i}$. To apply the deformation, the steps are the following:

1. Compute the number of steps, s, using expression (3).
2. Deform the vertices s times using expression (2), replacing M_j with $\frac{1}{s}\odot M_j$, and updating the absolute transformation with the latter matrix.

Normal deformation: In order to deform the normals, we need to compute the co-matrix of the Jacobian [Barr 1984]. Even though a closed form can be derived from the above transformation, its length makes it difficult to code and time consuming. In practice, computing the Jacobian with finite differences works well enough[5].

4 Fast expressions for interactive sculpture

When using multiple tools, the time of the scene must be frozen in order to input each tool one at a time. However this is not the case for editing with a single tool. In this scenario, the transformations may just be pure translations, uniform scaling and rotations. The transformations of a point and its normal are much simpler to compute, as there is a closed form to the logarithm of such simple transformations. In this section, in addition to efficient expressions for computing the number of required steps, we provide fast deformation functions for a vertex and its normal. For the normal, computing the Jacobian's co-matrix is not always required: $(\mathrm{com}J^t)\vec{n}$ leads to much simpler expressions for translations and uniform scaling. Note that the normal's deformations do not preserve the normal's length. It is therefore necessary to divide the normal by its magnitude. We note $\vec{\gamma}^t = (\gamma_x^t, \gamma_y^t, \gamma_z^t)^\top$ the gradient of ϕ^t at p.

4.1 If M is a translation:

The use of $\odot$ can be simplified with translation vector $\vec{d}$. The minimum number of steps is:

$$-\min\left(\frac{\delta\mu^{t_i}}{\delta d}\right)||\vec{d}|| < s$$

The s vertex deformations are:

$$f^{\tau_k\mapsto\tau_{k+1}}(p) = p + \frac{\phi^{\tau_k}(p)}{s}\vec{d}$$

The s normal deformations are:

$$g^{\tau_k\mapsto\tau_{k+1}}(\vec{n}) = (1+\frac{1}{s}\vec{\gamma^{\tau_k}}^\top\vec{d})\vec{n} - \frac{1}{s}(\vec{d}^\top\vec{n})\vec{\gamma^{\tau_k}}$$

4.2 If M is a uniform scaling operation:

Let us define the center of the scale c, and the scaling factor σ. The minimum number of steps is:

$$-\min\left(\frac{\delta\mu^{t_i}}{\delta d}\right)\sigma\log(\sigma)S_{\max} < s$$

where $S_{\max}$ is the largest distance between a point in the deformed area and the center c, approximated using a bounding box. The s vertex deformations are:

$$f^{\tau_k\mapsto\tau_{k+1}}(p) = \sigma^{\frac{\phi^{\tau_k}(p)}{s}}(p-c)+c$$

Let $\vec{\chi} = \frac{1}{s}\log(\sigma)(p-c)$. The s normal deformations are:

$$g^{\tau_k\mapsto\tau_{k+1}}(\vec{n}) = (1+\vec{\gamma^{\tau_k}}^\top\vec{\chi})\vec{n} - (\vec{\chi}^\top\vec{n})\vec{\gamma^{\tau_k}}$$

[5]We used C++ double precision float numbers with $\epsilon = 1e{-}12$, with coating values λ_j between 0.2 and 10.

4.3 If M is a rotation:

Let us define a quaternion $q(\theta)$ of rotation angle θ, center of rotation r and vector of rotation $\vec{v} = (v_x, v_y, v_z)^\top$. The minimum number of steps is:

$$- \min(\frac{\delta \mu^{t_i}}{\delta d})\theta R_{\max} < s$$

where $R_{\max}$ is the distance between the axis of rotation and the farthest point from it, approximated using a bounding box. The s vertex deformations are:

$$f^{\tau_k \mapsto \tau_{k+1}}(p) = q(\theta \frac{\mu^{\tau_k}(p)}{s})(p - r)\overline{q}(\theta \frac{\mu^{\tau_k}(p)}{s}) + r$$

As the expression we obtained for $(\mathrm{com} J^{\tau_k})\vec{n}$ was not as simple as in previous cases, the s normal deformations are simply given as:

$$g^{\tau_k \mapsto \tau_{k+1}}(\vec{n}) = (\mathrm{com} J^{\tau_k})\vec{n}$$

where:

$$J^{\tau_k} = \begin{pmatrix} v_x A + \gamma_x^{\tau_k} B + \vec{n}_x & v_y A + \gamma_y^{\tau_k} B + \vec{n}_y & v_z A + \gamma_z^{\tau_k} B + \vec{n}_z \end{pmatrix}$$

$$\begin{array}{rclrcl} \vec{a} &=& p - r & \vec{n}_x &=& (C, Sv_z, -Sv_y)^\top \\ \vec{\xi} &=& \vec{a} - (\vec{a}\vec{v})\vec{v} & \vec{n}_y &=& (-Sv_z, C, Sv_x)^\top \\ C &=& \cos\left(\frac{\theta \mu^{\tau_k}(p)}{s}\right) & \vec{n}_z &=& (Sv_y, -Sv_x, C)^\top \\ S &=& \sin\left(\frac{\theta \mu^{\tau_k}(p)}{s}\right) \\ A &=& (1 - C)\vec{v} \\ B &=& \frac{\theta}{s}(C\vec{v} \wedge \vec{a} - S\vec{\xi}) \end{array}$$

5 Outline for an interactive modeler

Though modeling could be performed by a script, it is much more convenient to provide the designer with immediate visual feedback of the current state of the shape. We provide in this section complementary information for a practical implementation. The limitations imposed by speed requirements are discussed.

5.1 Shape model

Because modern graphics hardware accelerates the rendering of polygons, we chose to handle a polygonization of the surface for interactive display. Although our deformations could be applied to the control points of any parametric surface, we chose to represent the modeled shape with a mesh, refined or simplified in order to keep a homogeneous sampling. Thus, the scene is initialized with a polygonal model of a sphere with sampling properties on the size of the edges and the normal variation[6]. To refine and simplify the mesh, a simple edge split/collapse algorithm is applied at runtime, between each sub-keyframes. In order to quickly fetch the vertices that are deformed and the edges that require splitting or collapsing, these are inserted into octrees. Note that this spatial limitation is not restrictive for the artist, as he can scale and translate the entire model with our deformations. To fetch the part of the scene requiring update, a query is done with the tool's bounding box. Note that this bounding box is the one used in expression (3).

[6] A simple way to obtain an homogeneous sphere polygonization consists of starting with an octahedron, putting all its edges longer than k in a queue, splitting them and putting the pieces longer than k back in the queue. Each time a split is performed, the new edges are flipped to maximize the smallest angle.

Limitation: Suppose the scene is at time t_k, so that the shape S^{t_k} is shown to the user, and that he performs a deforming operation $f^{t_k \mapsto t_{k+1}}$ with the mouse. All the mesh refinements and simplifications are performed in S^{t_k}. This is however an approximation, as ideally the operations should be performed in the initial shape S^{t_0}, and $\underset{i=0}{\overset{k}{\Omega}} f^{t_i \mapsto t_{i+1}}$ should be applied to the new vertices. This would however become more and more time consuming as k gets longer. The approximation works well enough in practice.

5.2 Tool model

We propose to control the position, size and orientation of the tools by clicking on a *controller* with the mouse that allows to perform translations, uniform scaling and rotations along three axes or in the viewing plane. The tools can have three modes: if the user performs a right click on a tool, it is in *positioning* mode, and can be translated, scaled or rotated without deforming the space. If the user performs a left click, the tool is in *deforming* mode, and any transformation will deform space and the shape embedded in it in real time. If the user performs a middle click, the clock of the scene is frozen, the tools are in *multiple deforming* mode. This allows the user to position as many tools as required between t_i and t_{i+1}, which will deform space in parallel when the user presses an acknowledge key.

Computing the distance to a tool is required to compute the scalar field μ_j. The easiest tools that can be implemented are simple objects (sphere, cube) which have closed form expression for their distance to a point. It is however convenient for an artist to choose or to manufacture his own tools, as every artist has his own way of sculpting. For this purpose, we propose the possibility to *bake* the pieces of clay in order to use them as a tools (see Figure 4). By baking, we mean pre-computing a data structure such that the distance field can be efficiently computed. Various algorithms exist, and information can found in [Guéziec 2001]. Presenting them is however beyond the scope of this paper. In our implementation, we have used a BSP of the Voronoï diagram of the vertices, and computed the distance using the surrounding triangles.

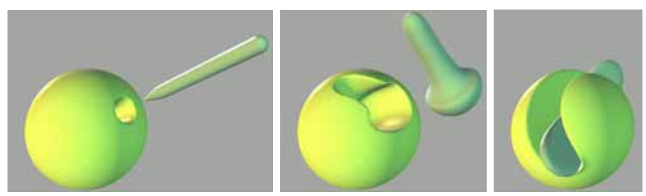


Figure 4: Example of customized tools deforming a sphere.

6 Results

Even though we limited ourselves to a few transformations (translation, uniform scale and rotation), the set of possible deformations is already very high because of the arbitrary shape and coating of the tools, and also because many tools' deformations can be blended. The shapes shown were modeled in an hour at most, and were all made starting with a sphere.

Figures 1, 5(a) and 5(b) show the use of the multi-tool to achieve smooth and symmetric objects. Figure 5(d) shows that sharp features can be easily modeled. Figure 5(d) and 5(h) show the advantage of foldover-free deformations, as the artist did not have to concentrate on avoiding selfintersections: our deformations do not change the topology of space and thus preserve the topology of the initial object.

7 Conclusion and future work

We have presented a new class of smooth and normalized space deformations that are intuitive corresponding to a clay modeling metaphor and preserve the shape's coherency. In order to do this, we combine transformations non-linearly in matrix logarithmic space, allowing us to parametrize and decompose the deformations using a foldover-free conjecture that still has to be proved. In the case of simple transformations for single tools, we provide fast expressions used for real time modeling. Future work consists of specifying more useful scalar fields, possibly using convolution surfaces. Also, for deforming implicit surfaces a fast way of inverting the function is required. This is theoretically feasible since our deformations are diffeomorphisms of space. We are also investigating ways to incorporate changes in topology.

A Foldover-free conjecture

To simplify notation, let us note:

$$\beta_j(p) = \frac{1 - \prod_{i=1}^{n}(1-\phi_i(p))}{\sum_{i=1}^{n}\phi_i(p)}\phi_j(p)$$

Let us define two points in space $p, q \in \mathbb{R}^d$. To find a condition on the deformation being foldover-free, we prove the following: if $q \neq p$, then their image should be different:

$$\bigoplus_{j=1}^{n}(\beta_j(q) \odot M_j)\, q \neq \bigoplus_{j=1}^{n}(\beta_j(p) \odot M_j)\, p$$

We consider q being in the neighborhood of p, i.e. the reachable space along the n paths of the deformation, with $h_j \rightarrow 0^+$:

$$q = \bigoplus_{j=1}^{n}(h_j \odot M_j)\, p$$

We substitute q and rearrange the equation:

$$\bigoplus_{j=1}^{n}(-\beta_j(p) \odot M_j)\bigoplus_{j=1}^{n}(\beta_j(q) \odot M_j)\bigoplus_{j=1}^{n}(h_j \odot M_j)\, p \neq p$$

Because $h_j \rightarrow 0^+$, the two leftmost matrices commute, and their product commutes with the rightmost matrix. We can therefore write the condition:

$$\bigoplus_{j=1}^{n}((\beta_j(q) - \beta_j(p) + h_j) \odot M_j)\, p \neq p$$

We suppose p is not an eigenvector associated with eigenvalue 1 of the above matrix, so we can generalize this vertex inequality to a matrix inequality:

$$\bigoplus_{j=1}^{n}((\beta_j(q) - \beta_j(p) + h_j) \odot M_j) \neq I$$

Applying the determinant and rearranging the expression:

$$\prod_{j=1}^{n}\det(M_j)^{-\frac{\beta_j(q)-\beta_j(p)}{h_j}} \neq \prod_{j=1}^{n}\det(M_j)$$

Since $h_j \rightarrow 0^+$:

$$\prod_{j=1}^{n}\det(M_j)^{-\frac{\delta\beta_j(q)}{\delta h_j}} \neq \prod_{j=1}^{n}\det(M_j)$$

Because $\forall \alpha, x \in \mathbb{R}$ the function $x \mapsto \alpha^x$ is increasing with respect to x, the deformation is foldover-free if $\forall j$:

$$-\frac{\delta\beta_j(q)}{\delta h_j} \neq 1$$

By substituting for β_j:

$$-\frac{\delta}{\delta h_j}\left(\frac{(1-\prod_i(1-\mu_i(d_i(q))))}{\sum_i \mu_i(d_i(q))}\mu_j(d_j(q))\right) \neq 1$$

Applying the chain rule:

$$\sum_k -\frac{\delta d_k(q)}{\delta h_j}\frac{\delta}{\delta d_k}\left(\frac{(1-\prod_i(1-\mu_i(d_i)))}{\sum_i \mu_i(d_i)}\mu_j(d_j)\right) \neq 1$$

By developing the derivative:

$$\begin{aligned}&-\tfrac{\delta d_j(q)}{\delta h_j}\tfrac{\delta \mu_j(d_j)}{\delta d_j}\tfrac{\prod_i(1-\mu_i(d_i))}{1-\mu_k(d_k)}\\ &-\textstyle\sum_{k\neq j}\tfrac{\delta d_k(q)}{\delta h_j}\tfrac{\delta \mu_k(d_k)}{\delta d_k}\tfrac{\mu_j(d_j)}{\sum_i \mu_i(d_i)}\left(\tfrac{\prod_i(1-\mu_i(d_i))}{1-\mu_k(d_k)} - \tfrac{1-\prod_i(1-\mu_i(d_i))}{\sum_i \mu_i(d_i)}\right) \neq 1\end{aligned}$$

It can be easily shown that $\frac{\mu_j(d_j)}{\sum_i \mu_i(d_i)} \in [0,1]$, $\frac{\prod_i(1-\mu_i(d_i))}{1-\mu_k(d_k)} \in [0,1]$ and $\frac{1-\prod_i(1-\mu_i(d_i))}{\sum_i \mu_i(d_i)} \in [0,1]$. Also, we have shown in appendix B that $\forall h \in \mathbb{R}, \frac{\delta d(h\odot Mp, T^{t_i})}{\delta h} \leq ||\frac{\delta h \odot Mp}{\delta h}||$, and we know that $\forall d \in [0,1]$, $-\frac{\delta\mu(d)}{\delta d} \leq -\min(\frac{\delta\mu}{\delta d})$. Thus, the deformation is foldover-free if $\forall j$:

$$\begin{aligned}&-\min(\tfrac{\delta\mu_j}{\delta d})\left|\left|\tfrac{\delta h\odot M_j p}{\delta h}\right|\right|_{h=0}\\ &-\textstyle\sum_{k\neq j}\min(\tfrac{\delta\mu_k}{\delta d})\left|\left|\tfrac{\delta h\odot M_k p}{\delta h}\right|\right|_{h=0} < 1\end{aligned}$$

We can rewrite these n conditions in a single one:

$$-\sum_i \min(\frac{\delta\mu_i}{\delta d})\left|\left|\frac{\delta h \odot M_i p}{\delta h}\right|\right|_{h=0} < 1$$

Note that $\left|\left|\frac{\delta h\odot M_j p}{\delta h}\right|\right|_{h=0} = ||\log M_j p||$. Because a matrix is a diffeomorphism, we can define n bounding boxes $p_{k_j \in [1,8]}$ around $K_j^{t_i}$ to approximate $||\log M_j p||$. Also, since taking fractions of the transformations prevents the space to fold on itself, we can introduce the number of steps we look for:

$$-\sum_j \min(\frac{\delta\mu_j}{\delta d})\max_{k_j\in[1,8]}\left|\left|\log(\frac{1}{s}\odot M_j)p_{k_j}\right|\right| < 1$$

Since $1 < s$:

$$-\sum_j \min(\frac{\delta\mu_j}{\delta d})\max_{k_j\in[1,8]}\left|\left|\log M_j p_{k_j}\right|\right| < s$$

This does not constitute a proof since we haven't shown that p is not an eigenvector associated with eigenvalue 1 of the concerned matrix.

B Proof $\forall h \in I\!R, \frac{\delta d(h\odot Mp, T^{t_i})}{\delta h} \leq ||\frac{\delta h\odot Mp}{\delta h}||$

Let $q \in T^{t_i}$ be the point of the tool that is closest to p: $d(p, T^{t_i}) = d(p, q)$. Once p has moved, q may not be the closest point anymore, so $\forall h \in \mathbb{R}, d(h \odot Mp, T^{t_i}) \leq d(h \odot Mp, q)$. Therefore we can introduce this inequality:

$$\begin{aligned}\tfrac{\delta d(h\odot Mp, T^{t_i})}{\delta h} &\leq \lim_{\epsilon\rightarrow 0}\tfrac{d((h+\epsilon)\odot Mp, q) - d(h\odot Mp, q)}{\epsilon}\\ &\leq \tfrac{\delta d(h\odot Mp, q)}{\delta h}\end{aligned}$$

To compute the derivative of the distance to a point, we use the following formula, obtained by deriving $\sqrt{(h \odot Mp - q)^2}$:

$$\frac{\delta d(h \odot Mp, q)}{\delta h} = \frac{(h \odot Mp - q) * \frac{\delta h\odot Mp}{\delta h}}{\sqrt{(h \odot Mp - q)^2}}$$

And finally, because the length of a vector is shorter when multiplied by a normal vector:

$$\left|(h \odot Mp - q)\frac{\frac{\delta}{\delta h}h \odot Mp}{||\frac{\delta}{\delta h}h \odot Mp||}\right| \leq \sqrt{(h \odot Mp - q)^2}$$

So we can substitute the latter:

$$\frac{\delta d(h \odot Mp, q)}{\delta h} \leq ||\frac{\delta h \odot Mp}{\delta h}||$$

References

ALEXA, M. 2002. Linear combination of transformations. In *Proceedings of SIGGRAPH'02*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 380–387.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Proceedings of SIGGRAPH'84*, ACM Press / ACM SIGGRAPH, vol. 18(3) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 21–30.

Blanc, C. 1994. A generic implementation of axial procedural deformation techniques. In *Graphics Gems*, vol. 5, 249–256. Academic Press.

Blanc, C. 1994. *Techniques de modélisation et de déformation de surfaces pour la synthèse d'images.* PhD thesis, Université Bordeaux I.

Borrel, P., and Bechmann, D. 1991. Deformation of n-dimensional objects. In *Proceedings of the first symposium on Solid modeling foundations and CAD/CAM applications*, 351–369.

Borrel, P., and Rappoport, A. 1994. Simple constrained deformations for geometric modeling and interactive design. In *ACM Transactions on Graphics*, vol. 13(2), 137–155.

Chang, Y.-K., and Rockwood, A. 1994. A generalized de Casteljau approach to 3d free-form deformation. In *Proceedings of SIGGRAPH'94*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 257–260.

Coquillart, S. 1990. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *Proceedings of SIGGRAPH'90*, ACM Press / ACM SIGGRAPH, vol. 24(4) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 187–195.

Crespin, B. 1999. Implicit free-form deformations. In *Proceedings of the Fourth International Workshop on Implicit Surfaces*, 17–24.

Decaudin, P. 1996. Geometric deformation by merging a 3d object with a simple shape. In *Graphics Interface*, 55–60.

Farin, G. 1990. Surfaces over Dirichlet tessellations. *Computer Aided Geometric Design 7(1-4)* (June), 281–292.

Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J., and Phillips, R. 1994. *Introduction to Computer Graphics.* Addison-Wesley. p.392.

Guéziec, A. P. 2001. "Meshsweeper": Dynamic point-to-polygonal-mesh distance and applications. *IEEE Transactions on Visualization and Computer Graphics 7*, 1 (January/March), 47–61.

Hsu, W. M., Hughes, J. F., and Kaufman, H. 1992. Direct manipulation of free-form deformations. In *Proceedings of SIGGRAPH'92*, ACM Press / ACM SIGGRAPH, vol. 26(2) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 177–184.

Kurzion, Y., and Yagel, R. 1997. Interactive space deformation with hardware assisted rendering. *IEEE Computer Graphics and Applications 17(5)* (September/October), 66–77.

MacCracken, R., and Joy, K. 1996. Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH'96*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 181–188.

Mason, D., and Wyvill, G. 2001. Blendeforming: Ray traceable localized foldover-free space deformation. In *Proceedings of Computer Graphics International (CGI)*, 183–190.

Mikita, M. 1996. 3d free-form deformation: basic and extended algorithms. In *Proceedings of the 12th Spring Conference on Computer Graphics*, 183–191.

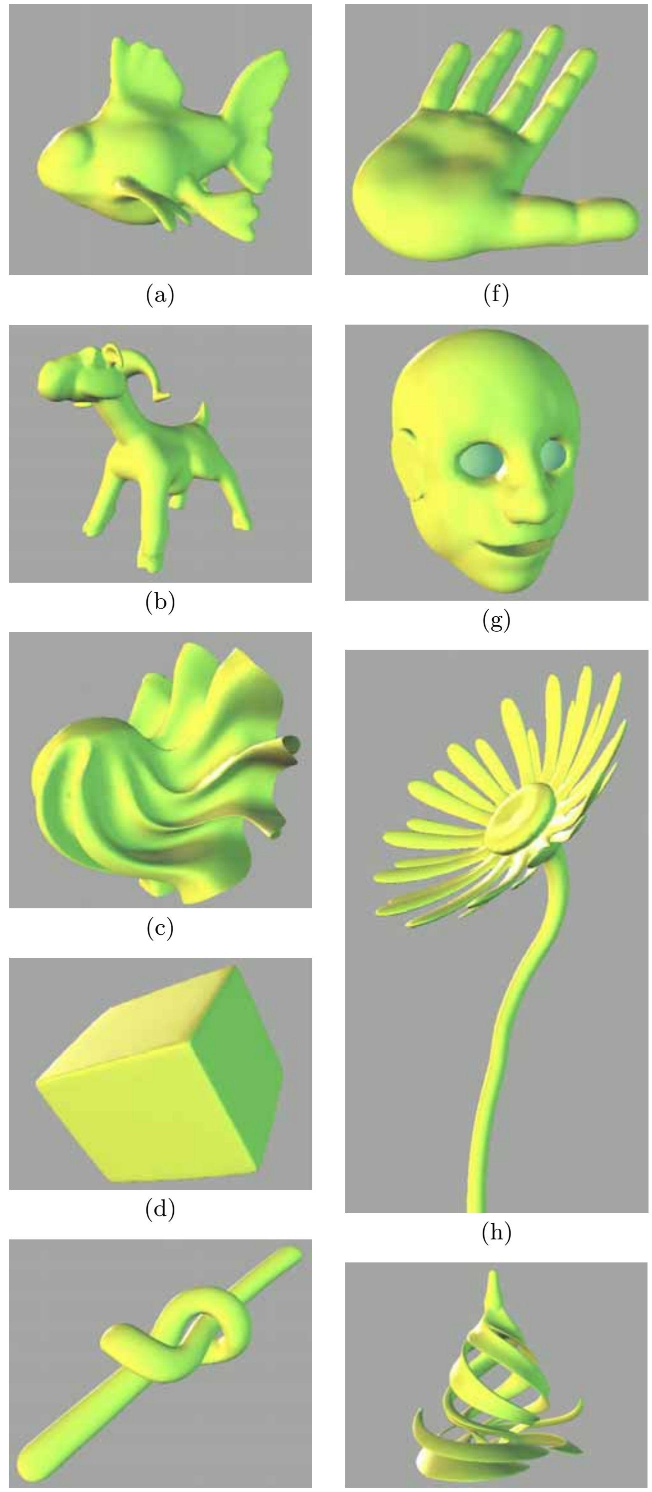
Moccozet, L., and Magnenat-Thalmann, N. 1997. Dirichlet free-form deformation and their application to hand simulation. In *Computer Animation'97*, 93–102.

Parent, R. 1977. A system for sculpting 3d data. In *Proceedings of SIGGRAPH'77*, ACM Press / ACM SIGGRAPH, vol. 11(2) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 138–147.

Sederberg, T., and Parry, S. 1986. Free-form deformation of solid geometric models. In *Proceedings of SIGGRAPH'86*, ACM Press / ACM SIGGRAPH, vol. 20(4) of *Computer Graphics Proceedings, Annual Conference Series*, ACM, 151–160.

Singh, K., and Fiume, E. 1998. Wires: a geometric deformation technique. In *Computer graphics, Proceedings of SIGGRAPH'98*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 405–414.

Wyvill, G., McRobie, D., Haig, C., and McNaughton, C. 1996. Free form modeling with history. *International Journal of Shape Modeling 2(4)* (December), 275–282.



(a) (f) (b) (g) (c) (h) (d) (e) (i)

Figure 5: All these shapes were obtained starting with a sphere, in at most one hour. In (c), the first modeling step was to squash the sphere into a very thin disk. In (g), eyeballs were added.