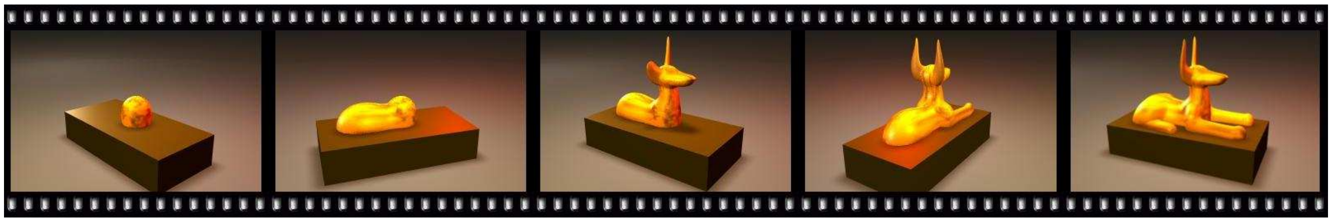
# Animated Sweepers: Keyframed Swept Deformations.

Alexis Angelidis
University of Otago
alexis@cs.otago.ac.nz

Geoff Wyvill
University of Otago
geoff@cs.otago.ac.nz


http://www.cs.otago.ac.nz/postgrads/alexis/anubis.mov

## Abstract

*This paper presents animated sweepers, a method for animating the space deformations called sweepers. This technique allows a user to animate the modeling of a shape, and to edit the animation until it satisfies the user's requirements. To place this method in the more general framework of animation with space deformation, we propose an overview of the possible relations between time and the deformation parameter. We formulate sweepers to take into account a varying time parameter, and present the data structure that allows the user to efficiently edit the modeling of a shape. This data structure has been implemented as a Maya plugin. Its implementation in the Maya API is described.*

***Keywords*—Free-form deformation, shape animation, non-linear transformations.**

## 1 Introduction

Metamorphosis, or animating an object undergoing high distortions, has applications ranging from special effects to medical imaging and scientific visualization. The most popular generic technique is mesh morphing [1], which transforms one mesh into another. Given two shapes, mesh morphing finds some path joining them. The quality of the result is influenced not only by the chosen path, but also by the shapes provided. We would like to give more control of the path to the user, which he will specify for modeling the desired shape. Space deformation provides a convenient formalism for defining the deformation of an object, although this technique has not yet been used for animating a shape undergoing great stretching and twisting.

Space deformation has been used in many ways as a framework for shape modeling. There are two possible approaches to the modeling process: the shape can either be put through a series of many simple deformations, as in [2, 3, 4], or a few but complex deformations, as in [5, 6]. In both cases, the shape is being morphed from a simple shape into the target object, and the only person who can visualize this complex morphing is the artist who creates it.

Space deformation has also been used for animating shapes, as for example in [7, 8, 9, 10]. Parameters of the deformation are controlled by time curves. These methods can achieve complex deformation but at a cost: either they are expensive to compute, or a lot of effort is required from the user to specify them.

In this paper, we are interested in animating the modeling of a shape, in order to present a spectator with the artist's view of the modeling process. This is valuable for teaching modeling skills and also as an art form in its own right. The key feature of our approach is the possibility of keyframing a deformation with large distortions, to edit the deformation, and finally to render it in high quality. In Section 2 we summarize the principle of the space deformations called sweepers. Then we study the possible relationships between the deformation parameter and the time parameter in Section 3. In Section 4, we present the 4D buffer structure that allows a user to edit the deformation of a shape efficiently. This structure has been implemented as a Maya plugin. We discuss our results in Section 5 with regard to both quality and interactivity.

## 2 Principle of Sweepers

In this section, we review the elements required for understanding the deformation, *sweepers*, introduced in [4], and we reformulate them in a suitable way for animation. Informally, a sweeper is a geometric tool together with a motion path. The basic idea is that the tool is placed somewhere in the region of a shape to be deformed and moved along the path. The motion drags a part of space with the tool subject to rules that prevent space foldover.

Space deformation provides a formalism to specify any modeling operation by successively deforming the space in which an initial shape, $S(k_0)$, is embedded. In this section, the reader can interpret $k$ as time. A deformed shape is given by the *modeling equation*[1]:

$$S(k_n) = \left\{ \mathop{\Omega}_{i=0}^{n-1} f_{k_i \mapsto k_{i+1}}(p) | p \in S(k_0) \right\} \tag{1}$$

where $f_{k_i \mapsto k_{i+1}} : \mathbb{R}^3 \to \mathbb{R}^3$ is the space deformation that deforms a point $p$ of the shape $S(k_i)$ into a point of the shape $S(k_{i+1})$. Since we want to animate the modeling process, we need the deformation to be continuous not only in space, but also in $k$, within the interval $[k_0, k_n]$. For this we modify Equation 1 into the *animation equation*:

$$S(k) \quad = \quad \left\{ \mathop{\Omega}_{i=0}^{n-1} f_{k_i \mapsto k_{i+1}}(p, k) | p \in S(k_0) \right\} \tag{2}$$

$$\text{where } f_{k_i \mapsto k_{i+1}}(p, k) \quad = \quad \begin{cases} p & \text{if } k \le k_i \\ f_{k_i \mapsto k_{i+1}}(p) & \text{if } k > k_{i+1} \\ f_{k_i \mapsto k}(p) & \text{otherwise} \end{cases}$$

where $f_{k_i \mapsto k} : \mathbb{R}^3 \to \mathbb{R}^3$ is the space deformation that deforms the shape $S(k_i)$ into the shape $S(k)$. Each deformation $f_{k_i \mapsto k_{i+1}}$ has to be continuous in $k \in [k_i, k_{i+1}]$. This can be achieved conveniently with the sweepers, shown below.

A *sweeping tool* is a field $\phi_k : \mathbb{R}^3 \mapsto [0, 1]$, animated in $k$. A simple way of specifying a tool is to use an animated rigid shape to which the distance $d_k$ can be computed:

$$\phi_k = \mu \circ d_k \tag{3}$$

where $\mu$ is a $C^2$ continuous piecewise polynomial, in which $\lambda$ is the thickness of a coating around the shape.

$$\mu(d_k) = \begin{cases} 0 & \text{if} \quad \lambda \le d_k \\ 1 + (\frac{d_k}{\lambda})^3 (\frac{d_k}{\lambda}(15 - 6\frac{d_k}{\lambda}) - 10) & \text{if} \quad \lambda > d_k \end{cases} \tag{4}$$

[1] $\mathop{\Omega}_{i=0}^{n-1} f_{k_i \mapsto k_{i+1}}(p)$ expresses the finite repeated composition of functions $f_{k_{n-1} \mapsto k_n} \circ \cdots \circ f_{k_0 \mapsto k_1}(p)$

Given a distance function $d$ to a static object in a canonical frame of reference, $d(p)$ can be conveniently animated with a matrix $M_k$:

$$d_k(p) = d((M_k)^{-1} p) \tag{5}$$

For the sake of efficiency, we have ignored the determinant previously used in [4] for rescaling the distance when the matrix $M$ scales the tool. Thus $d_k$ is not a Euclidean distance, but allows us to easily specify anisotropic scaling, useful for flattening the object locally. The deformation defined by a tool moving from $M_{k_i}$ to $M_{k_{i+1}}$ is swept using $s$ steps (see [4] for the formula giving $s$), subdividing the interval $(k_i, k_{i+1}]$ into "small enough" steps $\{\kappa_0, \dots \kappa_s\}$. Let us denote $M = M_{k_{i+1}} M_{k_i}^{-1}$ the transformation matrix. The sweeper modeling-deformation is thus the composition of $s$ functions:

$$f_{k_i \mapsto k_{i+1}} = \mathop{\Omega}_{j=0}^{s-1} f_{\kappa_j \mapsto \kappa_{j+1}}(p) \tag{6}$$

$$\text{where } f_{\kappa_j \mapsto \kappa_{j+1}}(p) = \frac{\phi_{\kappa_j}(p)}{s} \odot M$$

We recall that the operator $\odot$, is defined in [11] as follows: $\alpha \odot M = \exp(\alpha \log M)$. Note that it is tempting to replace the nesting operator, $\Omega$, with the matrix products, $\prod$, in Equation 1, but this would not be correct since $\phi_{\kappa_j}$ requires the transformed point in order to be evaluated. For computing the distance in Equation 5, we propose using $M_{\kappa_j}$:

$$M_{\kappa_j} = (\frac{j}{s} \odot M) M_{t_i}$$

In order to use sweepers for animation, we have to define $f_{k_i \mapsto k}$ for $k \le k_{i+1}$. Since $M_k$ can be assumed to be defined for all $k$, adapting sweepers to animation is as simple as substituting $M_{k_{i+1}}$ for $M_k$ in Equation 6. Figure 1 illustrates the substitution of $M_{k_1}$ for $M_k$.

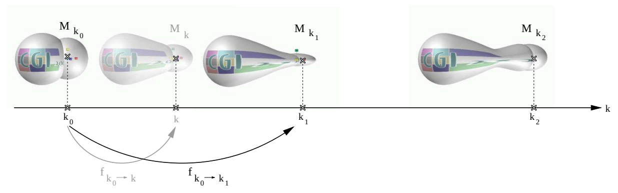


**Figure 1. Since the movement of the tool is continuous in $k$, so is the deformation. In this example, the tool's translation and scaling are animated.**

Thus, the sweeping of a tool defines a deformation that is smooth in both space and time; smooth meaning the field $\phi_k$ is $C^2$ in space, and the matrix $M_k$ is $C^1$ in $k$.

Note that the property of sweepers to rigidly transform the space inside the tool, allows the user to grab a portion of space with a movement of the tool. This movement has a parameter of its own, which is assimilated by the time parameter in the case of modeling. However, it can be used independently for animation, providing several ways of animating a shape with a sweeping deformation, as we show in the next section.

## 3 Animating a space deformation

The movement of a sweeping tool defines a deformation parameterized in $k$, while there is also an independent time parameter $t$. Figure 2 shows a surface defined as the Cartesian product of the two sets. The curve $k = 0$ is in the modeling space, where the unaltered object lies (white edge). At the other extremity of the surface is the animated shape (dark edge).

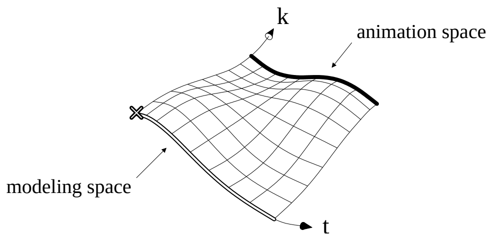


**Figure 2. Surface of the Cartesian product between time and the deformation parameter.**

Providing the user with a unified tool that would allow him to sample the entire surface proves to be both hard to implement and difficult for the user to visualize as an efficient animating tool, for in addition to the surface, the user has to provide other deformation parameters (scaling or rotating the tool). However, some special cases can be implemented easily and specified efficiently by the user. These involve the control of a single curve which will generate a surface. We describe three of these special cases below.

### 3.1 Straight deformation

By leaving the tool at a fixed position in the modeling space and animating only its destination through time, the animator only has to specify a curve (see Figure 3), the effect being to grab a portion of an object and to displace it straight from its original position to an animated target position. In Figure 4, a tool grabs a head smoothly, while the neck stays in place.

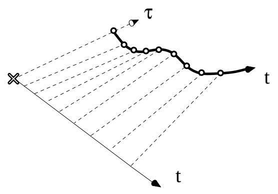


**Figure 3. The dark curve represents the animated portion of the modeling space, controlled by the user. The straight lines aligned with $k$ bring the shape straight from modeling space to animated space.**

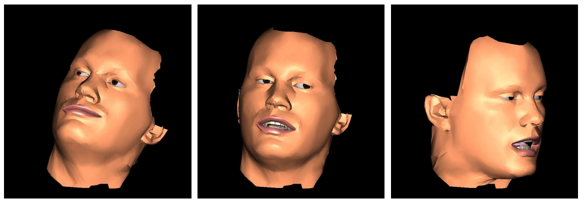


**Figure 4. Deformation of a neck using a straight deformation. The facial expressions though are procedural [12].**

### 3.2 Sliding straight deformation

By specifying a deformation in a local coordinate set, and animating the coordinate set, the animator can slide a deformation through the scene, as shown in Figure 5. For instance, if the deformation is a scale, it can be used for animating a bulge moving inside an object. This approach has been used for instance by [8], where instead of moving the deformation relative to the object, it is the object that is moved relative to the deformation. Also, we have implemented it with sweepers as a Maya plugin, which was used in a short animation as shown in Figure 6.

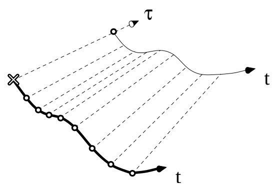


**Figure 5. Sliding a deformation through the scene. The curve controlled by the user is symbolized in thick black.**

### 3.3 Modeling deformation

By identifying the deformation parameter $k$ to the time parameter $t$, a user can easily specify and control an

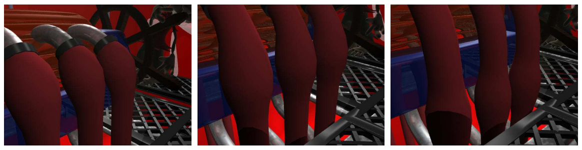


**Figure 6. Screenshots from a short animation using a simple deformation plugin under Maya. The bulge is animated by moving its locator inside the pipe.**

*animated sweeper*. Every new deformation is composed with the previous one, thus building up an increasingly complex function. As shown in Figure 7, each function is not animated and is constant in $t$. The control-points on the line $k = t$ in this figure correspond to key positions $M_{t_i}$.

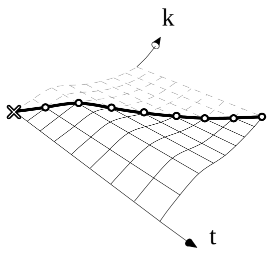


**Figure 7. Animating the modeling deformation. To each control-point corresponds a constant function, parallel with axis $t$. The only curve controlled by the user is symbolized in thick black.**

As time increases, the complexity of the deformation increases too. To provide quick feedback to the user, we propose in the next sections a data flow structure illustrated by an implementation.

## 4 Structure for an animated sweeper

Let us place ourselves in the context of a user working on an animation. The sweeping tool is controlled through eleven curves:

- Translation X, Y, Z (float)
- Rotation X, Y, Z (float)
- Scale X, Y, Z (float)
- Thickness (float): the constant $\lambda$ in Equation 4
- Active (boolean): if false, $f_{t_i \mapsto t} = \text{id}$ regardless of the ten other curves, otherwise the tool behaves normally. This is useful for moving the tool around the scene without interfering with the shape.

By modifying the control points of a curve driving the sweeping tool, the user is transparently modifying the animation equation (2). An example is shown in Figure 8.

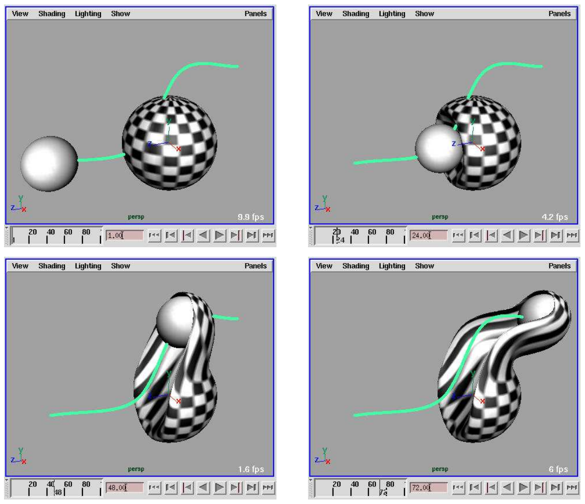


**Figure 8. Control of a deformation by keyframing the position of a tool. The green curve is the tool's motion trail.**

It would be rather inefficient if the shape $S(t)$ was to be recomputed from the inital shape $S(t_0)$ every time a key between $t_0$ and $t$ has been modified, or every time the user selects a different time in the time slider. To prevent this, we propose an efficient architecture that caches intermediate shapes so that the user can work in a reasonable amount of time, real-time in most cases.

We suggest placing a cached shape at every time where at least one key has been set. As shown in Figure 9, the list of cached shapes does not necessarily correspond to the control points of a curve.

### 4.1 Graph nodes

The cached shapes will be handled by nodes in a scene graph, and will be updated only when necessary through the connections between the nodes. In fact, since our system has been implemented in Maya 5.0, the nodes we describe in the following sections are customized Dependency Graph (DG) nodes. The reader is referred to [13] for a description on how data is pulled through the connections only when required.

*Shape node:* The shape node is responsible for displaying the current shape $S(t)$. It possesses an array of connec-

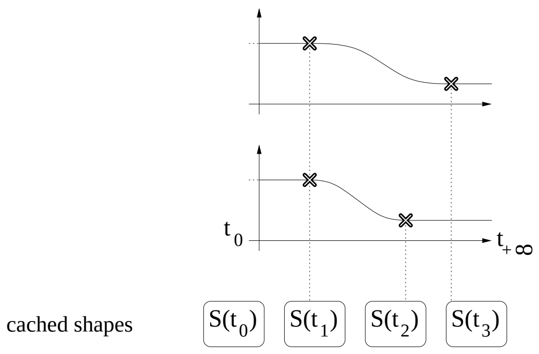


**Figure 9. Example of a configuration of cached shapes relative to the curves.**

tions to the cached nodes' output shapes, sorted by increasing time, and a connection to the current time. When the current time changes, the shape node determines which interval $(t_{i-1}, t_i]$ the value $t$ belongs to, and asks cache node $i$ its output shape.

*Cache nodes:* A cache node has as input the cached shape of the previous cache node. If there is no previous cache node, it generates shape $S(t_0)$. A cache node is responsible for computing and storing the cached shape $S(t_i)$ when requested by the next cache node, and is also responsible for computing output shape $S(t)$ when requested by the shape node.

*Tool node:* Draws the tool in the real time modeling view, and will not be rendered by default.

*Tool transform node:* The tool transform node connects all the animation curves, and provides the matrix positions $M_t$ to the cache nodes, so that they can deform the shape. The tool transform node also provides this matrix to the tool node so that it can be drawn at the right position.

The list of connected cache nodes represents a $4D$ buffer, and holds deformed shapes along time. This buffer is used to compute a single shape: the one displayed in the interface, at the current time.

### 4.2 Initialization

At initialization, there are two cache nodes at time $t_0$ and $t_{+\infty}$. The cache node at time $t_0$ contains the initial shape, before deformation. The cache node at time $t_{+\infty}$ contains the current final shape. The curves are clear of any key. By themselves, the two extremity caches define an inbetween function $f_{t_0 \mapsto t_{+\infty}}$ which is the identity: $S(t) = S(t_0)$. Keys can then be inserted on the curves while the buffer structure updates.

### 4.3 Operations

The operations are performed on the tool as if it were a classic animated object, through the eleven curves. These operations are performed on the scene graph via curve editing callbacks.

*Key insertion:* When a control point is inserted on a curve at time $t_i$, we first check whether a cache node already exists. If it does, we mark the node dirty. Otherwise, we insert a node in the node list.

*Key deletion:* When a control point is deleted on a curve at time $t_i$, we first check whether another curve has a control point at that time. If it does, we mark the node dirty. If not, we remove the node from the list.

*Key modification:* This is done by deletion followed by insertion.

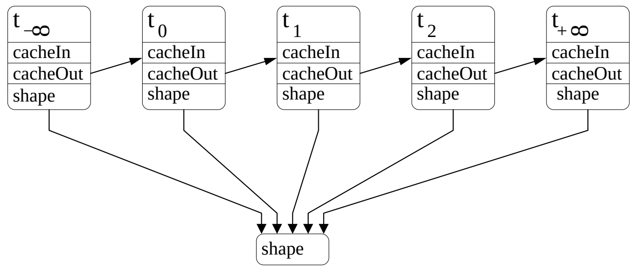


**Figure 10. Simplified data graph.**

### 4.4 Sampling the matrix curves

Maya provides a mechanism for $C^1$ interpolation of matrices, different from the one in [11], which was used for sweeping transformations [4]. Thus, using the foldover-free condition with Maya's interpolation is incorrect. To solve this, we sample the matrix at every frame using Maya's interpolation scheme, and we sweep the transformation within that interval using the foldover-free condition. We have observed that, most of the time, the number of substeps is equal to one.

## 5 Results

We have implemented the structure in C++ using the Maya 5.0 API [13]. The only difficulty of the implementation in Maya is the absence of a callback for detecting which control points have been modified. However, since it is possible to set a callback to know which curve has been modified, it is possible to circumvent this issue by duplicating the curve nodes (eleven, for each parameter). By counting the number of points, it is possible to know if a key has been modified, inserted or deleted, and by comparing the control points one by one, the modified control points of the curve can be found. Also, for rendering a custom shape, it is necessary to pass the custom mesh to a built-in Maya node.

Regarding interactivity, the system is fast and allows the user to specify and modify an animated sweeper with ease. Once all the cache nodes have computed their cached shape, playing the animation in real time is just a matter of deforming the shape within small time intervals $(t_i, t_{i+1}]$. Also, every cache is computed or updated only when required, that is when the current time in the time slider is greater than $t_{i+1}$. In order to precompute all the cached shapes, or update all the cached shapes when the first key has been modified, the user can select the last time $t_n$ on the time slider. This operation can be slow if there are many deformations. However most of the time, the modeling is done locally both in space and time, allowing the computation to be done in real time.

The title page animation of a sphere morphing into an Anubis statue has been done conveniently with the technique described, and then rendered in high quality. The first step flattens the bottom of the sphere using a vertical scaling. The tool is then disabled and placed on another part to pull out the body of the statue. The rest of the morphing is also done by pulling, scaling, twisting, disabling and enabling the tool.

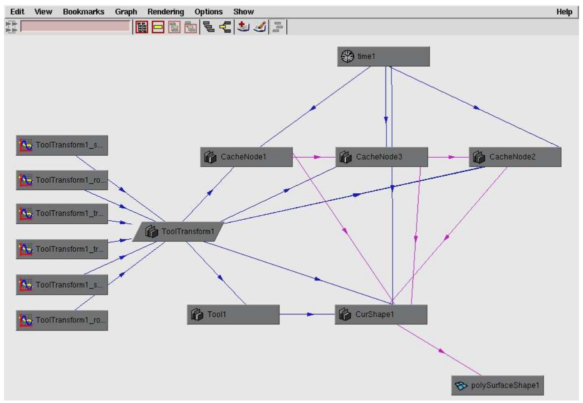


**Figure 11. Snapshot of the Maya dependency graph. Three keys have been set on translation X, rotation X and scale X, at the same time parameter. To this key corresponds a single cache node. Each curve has a duplicate used to modify the graph when a curve is edited.**

## 6 Conclusions

The method of deformation, sweepers, has been extended to show animations of the modeling process. Because of the inherent motion parameter of sweepers, this extension is straightforward.

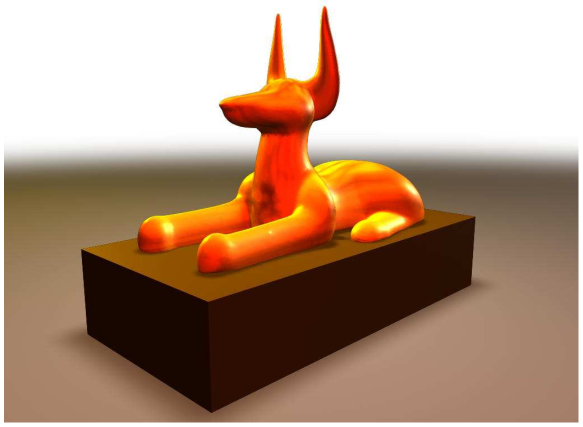


**Figure 12. Snapshot of the Anubis animation. The 2D texture (a noise) has been stretched with the deformation.**

We have explained the useful relations between time and the deformation parameter. Also, we have reformulated the modeling equation into an animation equation and presented a structure for animating the modeling of a shape in Maya. The system proves to be useful as a tool for animating a deformation, which would have been otherwise difficult to do with existing methods. The spectator can visualize the modeling of a shape in high quality rendering, without requiring the presence of the artist.

Future work includes specifying more than one tool at a time to animate the object, as has already been done in a modeling context [4].

Further, with space deformation, the texture undergoes intense stretching when provided with the initial model. This may or may not be wanted. A self-adjusting mapping of texture coordinates is being developed to control this problem.

## 7 Acknowledgments

Many thanks to Sui-Ling Ming-Wong for carefully proof-reading this paper, Scott King for deforming heads and Matthew Billowes for bulging pipes. Thanks also to Alias Technical Support for being quick and helpful, and to Alias for providing the licences. This work was supported by the Marsden Fund.

## References

[1] M. Alexa, “Recent advances in mesh morphing,” in *Computer Graphics Forum*, vol. 21, no. 2. ACM, June 2002, pp. 173–198.

[2] P. Decaudin, “Geometric deformation by merging a 3d object with a simple shape,” in *Graphics Interface*, May 1996, pp. 55–60.

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

[4] A. Angelidis, G. Wyvill, and M.-P. Cani, “Sweepers: Swept user-defined tools for modeling by deformation.” in *Shape Modeling International*. IEEE Computer Society Press, 2004, http://www.cs.otago.ac.nz/postgrads/alexis/smi04.pdf.

[5] P. Borrel and D. Bechmann, “Deformation of n-dimensional objects,” in *Proceedings of the first symposium on Solid modeling foundations and CAD/CAM applications*, 1991, pp. 351–369.

[6] W. M. Hsu, J. F. Hughes, and H. Kaufman, “Direct manipulation of free-form deformations,” in *Proceedings of SIGGRAPH’92*, ser. Computer Graphics Proceedings, Annual Conference Series, vol. 26(2), ACM. ACM Press / ACM SIGGRAPH, July 1992, pp. 177–184.

[7] A. H. Barr, “Global and local deformations of solid primitives,” in *Proceedings of SIGGRAPH’84*, ser. Computer Graphics Proceedings, Annual Conference Series, vol. 18(3), ACM. ACM Press / ACM SIGGRAPH, July 1984, pp. 21–30.

[8] S. Coquillart and P. Jancène, “Animated free-form deformation: An interactive animation technique,” in *Proceedings of SIGGRAPH’91*, ser. Computer Graphics Proceedings, Annual Conference Series, vol. 25(4), ACM. ACM Press / ACM SIGGRAPH, July 1991, pp. 23–26.

[9] L. Moccozet and N. Magnenat-Thalmann, “Dirichlet free-form deformation and their application to hand simulation,” in *Computer Animation’97*, June 1997, pp. 93–102.

[10] K. Singh and E. Fiume, “Wires: a geometric deformation technique,” in *Computer graphics, Proceedings of SIGGRAPH’98*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, July 1998, pp. 405–414.

[11] M. Alexa, “Linear combination of transformations,” in *Proceedings of SIGGRAPH’02*, ser. Computer Graphics Proceedings, Annual Conference Series, ACM. ACM Press / ACM SIGGRAPH, July 2002, pp. 380–387.

[12] S. A. King, A. Knott, and B. McCane, “Language-driven nonverbal communication in a bilingual conversational agent,” in *16th International Conference on Computer Animation and Social Agents (CASA 2003)*, may 2003, pp. 17–22.

[13] D. A. Gould, *Complete Maya Programming: An Extensive Guide to MEL and the C++ API*, E. Science, Ed. Morgan Kaufman, 2003.