Warping as a Modelling Tool for CSG/Implicit Models

Brian Wyvill
University of Calgary
Department of Computer Science
2500, University Dr. NW Calgary, Canada
blob@cpsc.ucalgary.ca

Kees van Overveld
Department of Mathematics and Computer Science, Eindhoven University of Technology
PO Box 513; 5600 MB, Eindhoven, the Netherlands
wsinkvo@win.tue.nl

Abstract

Free form deformations are useful for describing a class of complex motions within an animation system. Such deformations have been described using a generalization of parametric surfaces and their application to modeling is well documented. In this paper we present a method that can be applied to implicit surfaces which are defined as an iso-surface around a set of skeletal elements. The resulting surface is approximated by a polygon mesh. Shape distortions, such as "squash and stretch" are applied automatically to models in motion by warping the space in which the models exist. A model will change its shape as the function defining the warped space can change over time or, the model will deform as it moves through the warped space. Our system also treats groups of skeletal implicit primitives as CSG primitives. Warping can also be applied to these primitives. Different warp functions, for example bend, taper *and* twist *can be applied locally or globally, the contribution from each primitive is calculated using a set of warp functions associated with that primitive. The interesting shapes, and possibly non-linear motion obtained from space warping, would be difficult to reproduce using other techniques.*

Key words: Computer Graphics, implicit surfaces animation, warping.

1. Introduction

For some time we have been investigating the use of implicit surfaces for computer animation. The objective of this work is to provide the animator with unified modeling and motion controls that emulate traditional character animation as well as provide interesting modelling tools to the designer. In the following we discuss the use of warping applied to skeletal implicit surface models [23] & [3] and introduce a number of warp functions that provide an animator with some useful techniques. One particular useful set of deformations are the *Barr Operators*, *twist*, *taper* and *bend* as described in [6]. The basic idea of our system is that an implicit surface model, represented by a skeleton, can be made to change its shape in a smooth manner by moving it through the warped space, or by applying a time dependent warp function to the space in which the model exists.

1.1 Overview of this paper

In section 2 we summarize previous work in this area. To make this paper self contained, section 3 deals with the concept of skeletal implicit surface modeling combined with CSG. In section 4 we present the basic notion of our warping technique and in section 5 we describe the use of the Barr operators. In section 6 we provide some implementation details. Illustrative examples are given throughout. Our conclusions are summarized in section 7.

1.2 Motivation

Implicit surface models are usually visualized as a polygon mesh, because modern graphics workstations have hardware for fast rendering of polygons. Free form deformations such as those described in [19] could be applied to this mesh (post-warping). One problem with such an approach is that the distribution of the polygon sizes is not predictable. Current polygonization methods e.g. [2] will optimize the unwarped mesh, so that flat areas are represented by few

large polygons and areas with high curvature will have a high polygon density. Large deformations can cause artifacts where areas of high curvature are represented by too few polygons. A possible remedy is to re-polygonize, but this is clearly a less efficient option than performing the warping prior to polygonization. We refer to these two approaches as pre and post-warping. It is often the case that polygonization is a method of prototyping before ray tracing. Ray tracing warped implicit/CSG primitives precludes the need to polygonize making post-warping an even less desirable option.

A second advantage of pre-warping is that the evaluation of the warp function can be part of the evaluation of the implicit function which gives rise to a more efficient algorithm.

As described in section 2, there are several ways in which pre-warping can be realized. In our implicit surface system a mathematical function is evaluated to find a value (the *field value*) at any point in space. Modifying the field value by a *warp function* is a concise and simple way to implement warping. For modelling, different warps can be associated with each primitive to give the concept of local warping as well as a global warp that would change the space in which all the objects exist. For character animation, a library of motion primitives (different warp functions controlled over time) can be used to achieve a variety of behaviors.

2. Previous Work

One of the most often used methods for free form deformations was introduced by Sederberg and Parry [19]. In their algorithm points in the original object are displaced by an amount that depends on the displacement of control points, initially defined in a rectangular 3 dimensional grid. The displacement vector applied to an arbitrary point in space is a tri-variate interpolation of the control points, using Bernstein polynomials. This assures a smooth deformation field, but requires relatively large amount of redundant data (the undisplaced control points) that have to be provided, especially in the case of a deformation with limited support (i.e. of finite spatial extent). Coquillart and Jancene [7] have applied deformations over time by applying key-frame interpolation to control points.

Jim Kleck, in his masters thesis, ([10]), mentions the possibility of space warping for skeletal implicit surfaces. Our own earlier work in this area (see [21]) which develops several examples of warping is summarised in section 4. Borrel and Bechmann ([4]) and later, Bechmann and Dubreuil have also worked on similar ideas in a somewhat more general context and give examples applied to parametric surfaces. They also present a general space deformation represented by a mapping from $\Re^n$ into $\Re^n$. Also similar is that the deformation is represented by a function that associates a point P by its displaced point P_w. The global nature of the deformation is localised by defining a bounding volume through which the space warp has effect. The volume is embedded in the global space, blending smoothly at the boundary. "The volume constraints is implemented using a pseudo-inverse technique to calculate the projection matrix to achieve the constraint." One way in which our work differs, is that locality is defined in a different manner obviating the need for the pseudo-inverse calculation.

Barr introduced the notion of global and local deformations using the operations of *twist*, *taper* and *bend* see [6]. These operations were applied to implicit surfaces by Pasko et al ([16]), who based the operations on functionally defined primitives. This functional approach, called the theory of *R-functions*, also encompasses an analytical description of blending and CSG operations. Despite these developments, for practical purposes, skeletal implicit surfaces offer an intuitive method for designing models which blend, ([3].

3. Skeletal Implicit and CSG Modelling

A skeleton is composed of a number of geometrical skeletal elements such as points and lines. A scalar field is manufactured by summing contributions from the fields defined around each skeletal element. Jim Blinn introduced the idea of modelling with iso-surfaces as a side effect of a visualization of electron density fields. [1]. Such models have various desirable properties including the ability to blend with their close neighbours. These models have been given a variety of names in particular: *Blobby Molecules* (Blinn), *Soft Objects* (Wyvill) [23] and *MetaBalls* (Nishimura) [15]. Jules Bloomenthal pointed out that these models could be grouped under the more general heading of *implicit surfaces*, defined as the point set: $F(P) = 0$ [2].

Implicit surface modelling techniques are now beginning to penetrate the animation industry. Several examples in commercial animation exist including at least one commercial system (the MetaEditor - Meta Corporation) an interactive editor which uses metaballs.

In the work described in this paper *Implicit Surfaces Models* are defined as groups of blended primitives based on skeletons. The groups are then combined using boolean expressions based on the techniques of Constructive Solid Geometry (CSG; [17], [13]). Such models have distinct advantages over blended implicit surface models or CSG models, i.e. CSG expressions where primitives may form blended groups instead of isolated skeletal elements. Automatic blending and CSG operations form a powerful combination for the rapid design of complicated models. To distinguish the combined models we refer to them as Boolean Compound Soft Objects, (**BCSO**). Some CSG systems use algebraic functions to define blends between primitives (E.g. SvLis [5]) which also provides a polygoniser for visualiz-

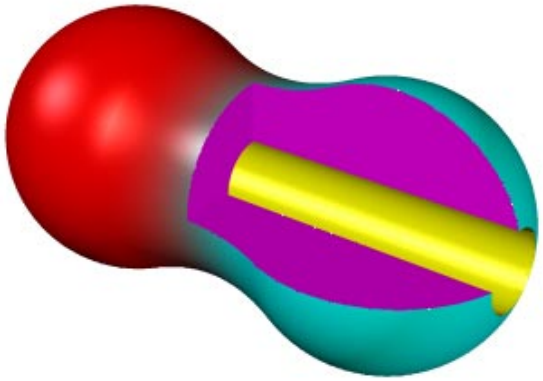


Figure 1. Boolean Compound Soft Object

ing of the models (see [20]). However our system is rather different in that blending is done between implicit surface primitives.

3.1. BCSO Models

Boolean Compound Soft Objects are defined in terms of groups of blended skeletal elements. The relationships between the groups are defined by means of a boolean expression. The primitives available in our system along with the defining geometry in parenthesis are listed below:

- Sphere (point)
- Cylinder with hemispherical ends (line segment and radius)
- Cone with hemispherical ends (line segment and two radii)
- Ellipsoid (3 orthogonal vectors)
- Torus (point, two radii and normal vector)
- Polygon (point list and normal vector)
- Plane (point and normal vector)

These primitives are assembled firstly into blended groups. Each group is a collection of skeleton elements whose contibution to the field function are summed, hence the surface of these elements are blended together. These groups may play the role of primitive terms in a CSG expression, hence the combination of both soft blends and hard joints.

The skeletal elements within one implicit surface model (ISM) group becomes a CSG primitive. There are no non-G^1 junctions within those elements. On the other hand, two ISM primitives are combined in the CSG-sense, and hence a visible junction arises there. So both types of junctions are supported within one surface representation scheme. Figure 1 shows such a BSCO model.

3.2. Rendering

When ray tracing is employed for BCSO's, CSG-type operations can be performed on-the-fly where the operands are individual primitives [24], but ray tracing is computationally expensive and not suitable for interactive display. On the other hand, when the model is polygonized first, then CSG-type operations can be performed afterwards on the resulting polygon meshes ([14]), since they are closed manifolds, but this has a high complexity in terms of the number of triangles in the meshes involved: the fully triangulated meshes of all input ISM's have to be available, even if a given ISM only contributes a small fraction of its surface. Also, this strategy cannot be used if one the participating CSG-primitives (ISM's) is unbounded, as for instance when intersecting with a planar half space in order to 'cut an object in half'. In our proposal of BCSO's we perform the CSG-operations on-the-fly while polygonizing the *resulting* surface. This means that the complexity is linear in the number of triangles of the resulting surface only, even if some of the contributing ISM's would have given rise to much larger triangular meshes. For details of our triangulation algorithm please see [22].

If the resulting BCSO is used for further manipulations that require a parameterized representation, the number of triangles can be reduced using resampling ([18]), and next use e.g. *Loop patches* ([12]), so if need be, ISM's can serve in a CAGD-context. BCSO's serve as an extension of the variety of shapes that may be modeled with plain ISM's, so they may be applied in the same areas.

3.3. The Implicit Function

The skeleton is surrounded by a scalar field $f_{total}(\mathbf{p})$ (equation 1). The value of the field being the highest on the skeleton, and decreasing with distance from the skeleton. The function $f_{total}(\mathbf{p})$, relates the field value (intensity) to distance from the skeleton, it has an impact on the shape of the surface, and determines how separate surfaces blend together (see [9]). The surface is defined by the set of points in space for which the intensity of the field has some chosen constant value (or iso-value thus the name *iso-surface*). Fields from the individual elements of the skeleton (belonging to the same smoothly blended group) are added to find the field value at some chosen point. (Values can be negative or positive). The value at some point in space is calculated as follows:

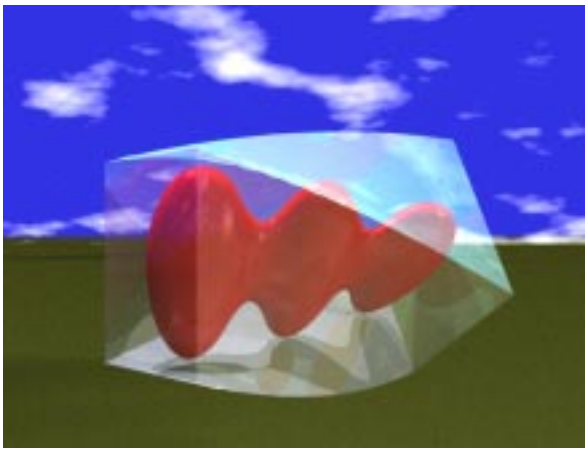


Figure 2. Warp Space

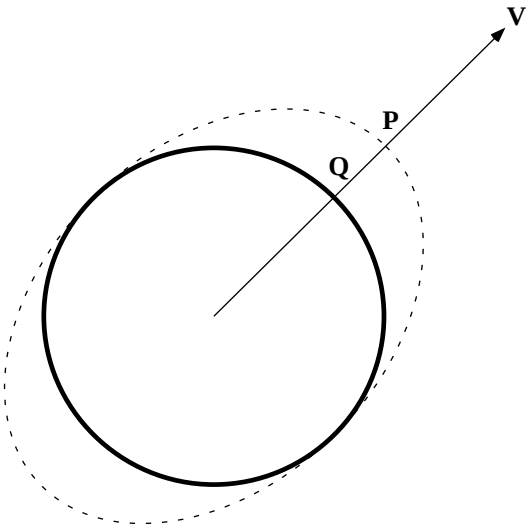


Figure 3. p is warped to q. The original sphere is warped to an ellipsoid.

$$f_{total}(\mathbf{p}) = \sum_{i=1}^{i=n} c_i f_i(r_i) \quad (1)$$

where $\mathbf{p}$ is a point in space

$f_{total}(\mathbf{p})$ is the value of the field at $\mathbf{p}$

n is the number of skeletal elements

c_i is a scalar value (c_i can be either positive and negative)

f_i is the field function of the i_{th} element

r_i is the distance from $\mathbf{p}$ to the nearest point $\mathbf{q}_i$ on the i_{th} element.

4. Warping

A useful tool in our system is the ability to distort the shape of a surface, by warping the space in its neighbourhood . A warp is a continuous function, w , from $\Re 3$ into $\Re 3$. In the following section we suggest some specific warp functions that are useful for producing some unusual models and animations. Sederberg provides a good analogy for warping when describing free form deformations, [19]. He suggests that the warped space can be likened to a clear, flexible plastic parallelpiped in which the objects to be warped are embedded. Although we have not as yet implemented FFD in our implicit surface system, figure 2 shows such a parallelpiped containing three blended ellipsoids. The parallelpiped is in fact made from six intersecting half space planes and the whole thing twisted.

The warped surface is defined from equation (1) above:

$$f_{total}(\mathbf{p}) = \sum_{i=1}^{i=n} c_i f_i(r_i) \quad (2)$$

where $r_i = dist(w_i(\mathbf{p}), \mathbf{q_i})$

where $w_i(\mathbf{p})$ is the position of the point $\mathbf{p}$ in warped space. In fact each skeletal element may reside in a different warped space. So when evaluating the contribution from the i^{th} skeletal element, $\mathbf{p}$ is first warped to the appropriate position before evaluating the distance function.

As a first example, we study a warp function $w_i(\mathbf{p})$, which warps a point $\mathbf{p}$ to a point $\mathbf{q}$ where the warp is paramterised by means of a vector $\mathbf{v}$; it may be given by the vector equation:

$$w_i(\mathbf{p}) = \mathbf{p} - (\mathbf{v}.\mathbf{p}')\hat{\mathbf{v}}$$

where $\mathbf{p}'$ is $\mathbf{p} - S_{0,i}$.

$S_{0,i}$ is the origin of the i_{th} skeleton

$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$

To understand how this affects the iso-surface consider $\mathbf{p}$ to be a point some distance from the surface of a sphere, such that the point is warped in the direction of the center of the sphere, to a position $\mathbf{q}$ which is on the iso-surface. The value returned for $\mathbf{p}$ by the implicit function is the value that would have been returned for $\mathbf{q}$ if warping were not in effect. In this case that value is the iso-value. Thus $\mathbf{p}$ becomes a point on the surface and the sphere is warped to an ellipsoid. (See figure 3). One application is in animation, John Lasseter notes [11] the importance of squash and stretch in traditional animation. The example he gives is of a ball traveling along a parabolic path and bouncing. The shape of the ball distorts into an ellipsoid to give the feeling of speed, when it bounces the distortion changes so that the long axis of the ellipse is parallel to the ground, in other words a squash effect.

To simulate the distortion of the ball to the ellipsoid, the usual computer graphics approach would be to use a scale

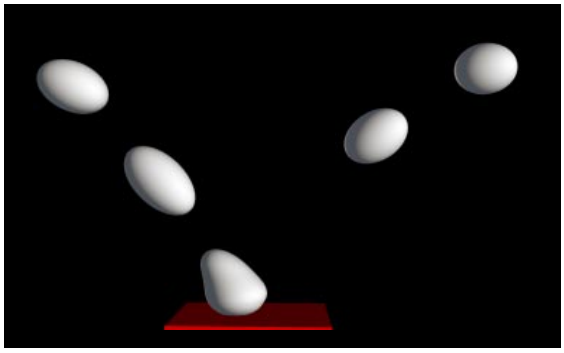


Figure 4. Ball Animation

operation over time. Since the scale is in the direction of the velocity vector v, two rotations are necessary, to first align the object with one of the major axes, then to rotate back again. With a complex 3D object consisting of many skeletal elements, shape distortion using a scaling operation may not be exactly what the animator requires. For instance on the impact plane, the ball should flatten out and the distortion is different from the deformation when the ball is further away, an effect which cannot be achieved with linear transformations. By exaggerating the non-linearity the ball could appear to be made of putty. Such a non-linear operation can easily be achieved by a warp operation, as can be seen in figure 4.

In our approach, this merely requires defining an appropriate warp vector field **v**, where, in order to achieve the squeeze and stretch effects, this field should depend on the actual velocity of the object to be deformed as in the case of the putty ball example. Many kinds of warp are possible, by simply writing a function that transforms a point from ordinary space into warped space. Each skeletal element contributes in a local way to the warped space, since each has its own local warp function associated with it.

5. Using Barr's deformations

Barr [6] combines the deformations in a hierarchical modelling structure, creating complex objects from simpler ones. In our system we also allow the warp operations to operate locally or globally and warps can be similarly combined.

Figure 5 shows examples of bend, taper and twist (see section 6). The object that has been warped is the same as that shown in figure 1.

5.1. Local Warping

Warping may be bounded by applying a warp to a group of primitives within one model. The contribution of each primitive is found by evaluating the implicit equation 2 above.

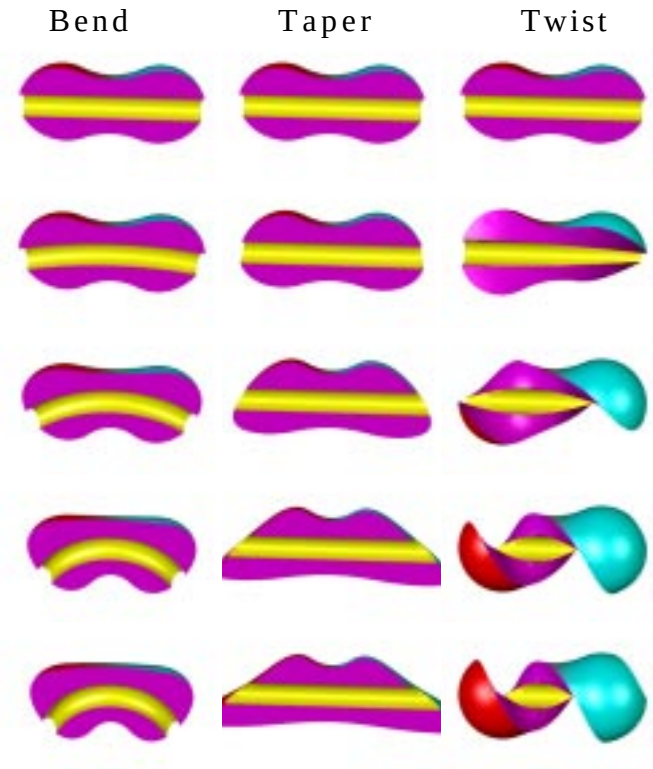


Figure 5. Bend, Taper and Twist applied to BCSO object

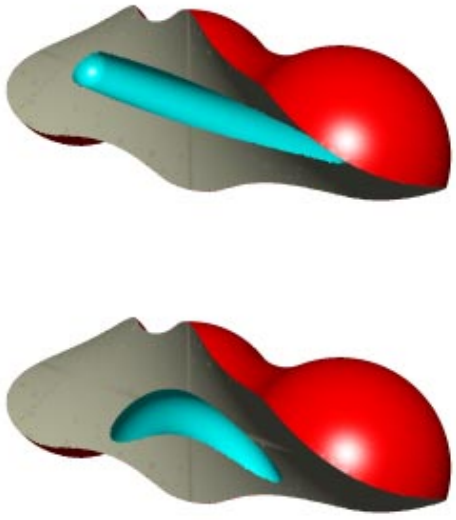


Figure 6. Local Warp Operations

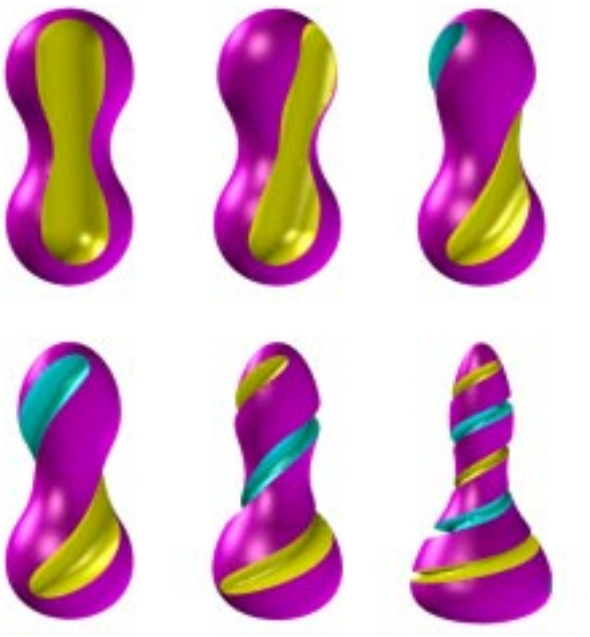


Figure 7. Twist and Taper

Each primitive has a list of warp functions associated with it. An example of different warps applied to different skeletal elements can be seen in figure 6. The figure shows three blended spheres and a cylinder subtracted from the model. The intersection with a plane reveals the inside of the object. In the upper image all the primitives have been warped using the Barr twist function, the lower image shows the same model only this time the cylinder has had the Barr bend operator instead of twist applied to it before it is CSG subtracted from the blended spheres.

5.2. Multiple Warps

A point can be warped several times to achieve nested warps. For example figure 7 shows some frames from an animation sequence. The initial model consists of two vertically aligned cylinders which have been subtracted from a pair of blended sphere primitives. Twist and taper are applied over time. For every point evaluated, the warped point $\mathbf{p}'$ is given by $twist(taper(\mathbf{p}))$. To view the animations and colour versions of the figures in this paper please see http://www.cpsc.ucalgary.ca/ blob/home.html.

5.3. Affine Transformations

The affine transformations can also be applied as warp functions. Although the effect of this is no different from applying the same transformations to the skeletons in normal space, the advantage is that a skeletal element can be defined

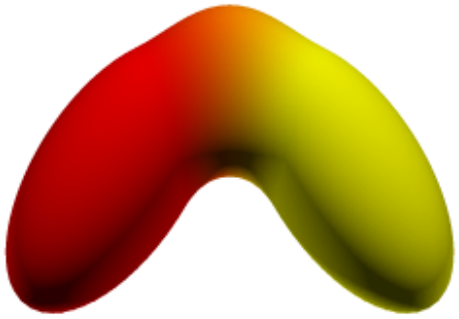


Figure 8. Using warping to perform affine transformations

in its canonical position and orientation and a warp used to transform it into the world space, as is common practice in many ray tracers. The advantage in our system is that warping and transformations may be treated in a consistent fashion.

The warp transforms each point by the appropriate affine transformation, and the field evaluated at the warped point. For example, figure refellipsoids shows two blended ellipsoids, each has been transformed by a scale, rotation and translation into the warped space. The only primitive used is a prototype sphere, which is more computationally efficient and saves a great deal of programming effort. Figure 9 was constructed from a single prototype cylinder. Various warps are used to position and twist the cylinders. The left-hand version is a union of the three cylinders, whilst the right-hand version treats the cylinders in a single blending group. Figure 11 shows some letters made up from twisted cylinders. Some of the letters are blended, others use the union operation.

6. Implementation

One of the features of using warping is that it is comparatively easy to implement by making simple modifications to an existing implicit surface polygoniser. A useful tutorial on the implementation of the Barr deformations is given in [8].

The data structure which is used to store the primitives is a linked list. The head of the list contains information as to the type of primitive, various attributes such as colour and the blending group to which the primitive belongs. Each subsequent node represents a warp, and contains information pertinent to that warp (see below).

To calculate the contribution of a particular primitive to a particular point; the point is first transformed successively

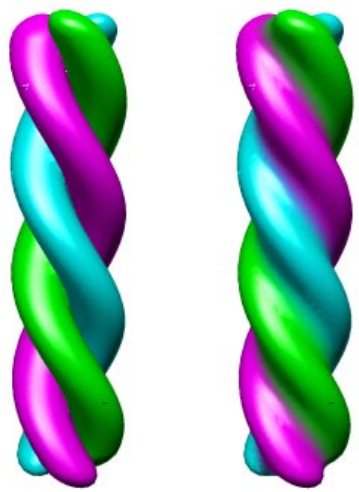


Figure 9. Twisted Cylinders, union (left) and blended (right)

by each of the local warp functions stored in the linked list. $warptype$ identifies the appropriate warp function. The value 0 is set aside for the null warp, i.e. $w(\mathbf{p}) = \mathbf{p}$,

Barr ([6]) gives details of how to find normal vectors to the warped primitive. Since the polygonizer is for prototyping only and warping may involve other warps than those defined by Barr, the Jacobian method has not been implemented. Instead, a numerical technique is used, the x component of the gradient is found from $\frac{f(x+\delta,y,z)+f(x-\delta,y,z)}{2\delta x}$ where δx is a small vector in the x direction and similarly for the y and z components.

Some examples of the Barr warps used in the figures in this paper are given below. The barr warps *taper, twist and bend* and the affine transformations *translate, rotate, scale and shear* have been implemented as a call with the following parameters:

```
VECTOR warp(p, w);
```

The point to be warped is **p** and `w` is a class data structure which contains the type of the warp, (bend, taper, etc.) the axis along which the warp is to take effect, and a vector of three values: `warpFactor` which are interpreted by the individual warp functions (methods).

Taper: A linear taper applied along one axis has proved to be the most useful, although quadratic and cubic tapers are also implemented. In citeBarr:84 an example is given of tapering along the z-axis, in which case both x and y are changed. In our system taper only affects one of the other coordinates and if both are required to change then two different calls to taper may be nested.

E.g. Linear Taper x along z:

$$w(x,y,z) = \left\{ \begin{array}{c} x*(1+z*warpFactor1) \\ y \\ z \end{array} \right\}$$

Bend: This operation is also applied along one major axis. For the bend example below the bending rate is k measured in radians per unit length, the point $xo, 1/k$ is the axis of the bend and the angle θ is defined as $(x - x0) * k$.

E.g. Bend x along z:

$$w(x,y,z) = \left\{ \begin{array}{c} -sin(\theta)*(y-1/k)+x0 \\ cos(\theta)*(y-1/k)+1/k \\ z \end{array} \right\}$$

Twist: In this example a twist has been applied around the $zaxis$ by $\theta = z * warpFactor1$.

E.g. Twist around z:

$$w(x,y,z) = \left\{ \begin{array}{c} x*cos(\theta)-y*sin(\theta) \\ x*sin(\theta)+y*cos(\theta) \\ z \end{array} \right\}$$

Shear: A linear shear has been implemented along with the other affine transformations, however Hamman in [8] extends the Barr deformations to a non-linear shear in the spirit of the other functions. This has been implemented as follows:

E.g.: shear along x dependent on z:

$$w(x,y,z) = \left\{ \begin{array}{c} x+f(z) \\ y \\ z \end{array} \right\}$$

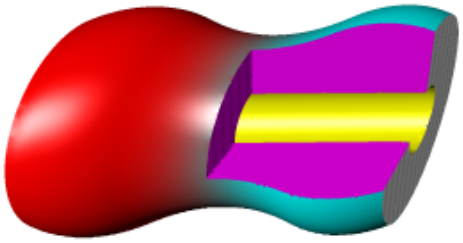
An example of this operation is given in figure 10, the two images top to bottom, show linear ($f(z) = k * z$) and quadratic (parabolic) ($f(z) = k * z^2$) shear.

7. Conclusions and Future Work

We have shown the usefulness of warping for both special modelling effects, such as the Barr operators and also as a simple way to implement geometric transformations of primitives. Warps may be nested and apply locally to particular primitives. Several examples have been given demonstrating the usefulness of warping for both modelling and animation.

The system of global and local warping is not very satisfactory as each instance of a primitive in a group has to be changed if a new warp is required to operate on that group. Clearly the linked list is an inadequate structure. A better idea would be to make warping a type of node in the CSG tree. Currently the CSG primitives in our system are defined as blended groups of implicit primitives. This

Linear Shear



Parabolic Shear

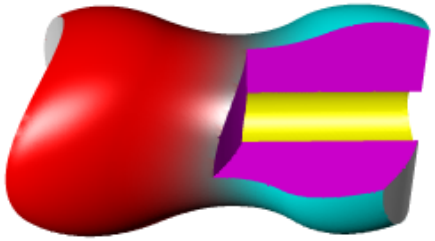


Figure 10. Shear Operations

Figure 11. Letters made from twisted cylinders, blended and unblended

is also not very satisfactory as blending groups can't contain primitives related by CSG operations. Adding a node which blends it's two children would somewhat ameliorate this problem. The new CSG tree will contain the following nodes: $\cap(union)$, $\cup(intersection)$, $-(difference)$, $+(blend)$ and $\omega(warp)$. This allows the designer to blend arbitrary sub-trees of primitives which may have some complex CSG and blending relationships.

8. Acknowledgements

The authors would like to thank the following students, Eric Galin (Ecole Centrale de Lyon) and Andrew Guy (University of Calgary) for their input into the design of our implicit surface system and also fellow researchers, Geoff Wyvill (University of Otago) and Jules Bloomenthal (Microsoft) for their very considerable contributions to this work over the years.

This work is partially supported by grants from the Natural Sciences and Engineering Research Council of Canada.

References

[1] J. Blinn. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1:235, 1982.

[2] J. Bloomenthal. Polygonisation of Implicit Surfaces. *Computer Aided Geometric Design*, 4(5):341–355, 1988.

[3] J. Bloomenthal and B. Wyvill. Interactive Techniques for Implicit Modeling. *Computer Graphics*, 24(2):109–116, 1990.

[4] P. Borrel and D. Bechmann. Deformation of n-dimensional objects. In *Symposium on Solid Modelling Foundations and CAD/CAM Applications (Austin, Texas, June 1991)*. ACM Press, 1991.

[5] A. Bowyer. *Svlis Introduction and User Manual 2nd edition*. Information Geometers, 47 Stockers Ave., Winchester, UK, 1995.

[6] R. Cook. GLobal and Local Deformations of Solid Primitives . *Computer Graphics (Proc. SIGGRAPH 84)*, pages 21–30, July 1984.

[7] S. Coquillart and P. Jancene. Animated Free Form Deformation. *Computer Graphics (Proc. SIGGRAPH 91)*, 24(4):23–26, July 1991.

[8] A. Hamman. Deformation Based Modelling. Master's thesis, University of Calgary, October 1991.

[9] Z. Kacic-Alesic and B. Wyvill. Controlled Blending of Procedural Implicit Surfaces. Technical Report 90/415/39, University of Calgary, Dept. of Computer Science, 1990.

[10] J. Kleck. Modeling Using Implicit Surfaces. Master's thesis, University of California, Santa Cruz, June 1989.

[11] J. Lasseter. Principles of Traditional Animation Applied to 3D Computer Animation. *Computer Graphics (Proc. SIGGRAPH 87)*, 21(4):35–44, July 1987.

[12] C. Loop. Smooth Spline SUrfaces over Irregular Meshes. *Computer Graphics (Proc. SIGGRAPH 93)*, pages 303–310, August 1993.

[13] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland 20850, 1988.

[14] B. Naylor, J. Amantides, and J. Thibault. Merging BSP trees yields polyhedral set operations. *Computer Graphics (Proc. SIGGRAPH 90)*, 224(4):115–124, August 1990.

[15] H. Nishimura, A. Hirai, T. Kawai, T. Kawata, I. .Shirakawa, and K. Omura. Object Modelling by Distribution Function and a Method of Image Generation. *Journal of papers given at the Electronics Communication Conference '85*, J68-D(4), 1985. In Japanese.

[16] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 2(8):429–446, 1995.

[17] A. Requicha. Representations for Rigid Solids: Theory, Methods, and Systems. *ACM computing surveys*, 12(4):437–464, December 1980.

[18] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of Triangle Meshes. *Computer Graphics (Proc. SIGGRAPH 92)*, 26(2):65–70, July 1992.

[19] T. Sederberg and S. Parry. Free Form Deformation of Solid Geometric Models. *Computer Graphics (Proc. SIGGRAPH 86)*, 23(3):151–160, August 1986.

[20] J. Woodwark and K. Quinlan. The derivation of graphics from volume models by recursive division of the object space. *Proc. Computer Graphics 80 Conference, Brighton, UK, Online*, pages 335–343, 1980.

[21] B. Wyvill, J. Bloomenthal, G. Wyvill, J. Blinn, A. Rockwood, T. Bier, and J. Cleck. Course Notes. *SIGGRAPH '90, Course #23, Modeling and Animating with Implicit Surfaces*, 1990.

[22] B. Wyvill and K. van Overveld. Constructive *Soft* Geometry: The unification of CSG and Implicit Surfaces. Technical report, University of Calgary, Dept. of Computer Science, 1995.

[23] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2(4):227–234, February 1986.

[24] G. Wyvill and A. Trotman. Ray tracing soft objects. *Proc. CG International 90*, 1990.