Bender: Deforming 3D Shapes by Bending and Twisting a Virtual Ribbon with both Hands

Ignacio Llamas, Alexander Powell, Jarek Rossignac and Chris Shaw

College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Bender is an interactive tool for bending and warping triangulated surfaces. The designer uses a virtual ribbon to grab a portion of the shape and to deform it through direct manipulation. The ribbon is defined by its centerline—a wire made of two smoothly joined circular arcs—and by its twist—the continuous field of normal directions along the wire. The wire and the twist are controlled by a Polhemus tracker in each hand. The deformation model is based on a new formulation of a 3D space warp that uses screw-motions to map coordinate systems aligned with the initial ribbon to corresponding coordinate systems aligned with the final ribbon. Circular biarcs are easy to control and permit the correct handling of situations where a vertex is influenced by different sections of the wire. Screw-motions define smoother and more intuitive warps than other formulations. The combination significantly extends the editing capabilities of previously proposed shape deformation tools and produces smooth and predictable results for configurations where the radius of the tubular region of influence around the ribbon does not exceed the radii of the arcs.


1. Introduction

The use of two-handed 3D interaction for sketching and editing 3D shapes has the potential to enhance productivity and artistic freedom for designers. We present an interactive surface deformation tool, Bender, which is not meant to replace existing 3D sculpting tools but complements them by providing unprecedented ease for bending and twisting 3D shapes through direct manipulation.

Like several previously proposed approaches, Bender lets the user control a local space warp which is applied to the vertices of a triangle mesh representation of the surface being edited (Figure 1). In contrast to shape deformations based on space warps that satisfy point displacement constraints [Borrel and Rappoport 1994] and even position and orientation constraints [Gain 2000], [Llamas et al. 2003], Bender makes it easy to bend and twist long protrusions which dominate most animal, organic, and manufactured shapes.

A warp is specified by grabbing a subset of space around a user-controlled virtual ribbon and then changing the shape of the ribbon interactively using a Polhemus [Polhemus 2002] tracker in each hand (see Figure 2). The ribbon is constructed around a central wire. The six degrees of
freedom of each tracker control the position of one end-
point of the wire, the direction of the tangent to the wire at
this end-point, and the twist of a ribbon around this tangent
direction. The concept of the ribbon is used here to capture
and communicate to the user how this twist is distributed
along the wire (Figure 2). A ribbon that interpolates these
two sets of end-conditions is constructed and its image
updated as the user keeps moving the trackers. When the
user presses a button attached to one of the trackers, the
current shape of the ribbon is saved as the initial ribbon.
The surface in the vicinity of that initial ribbon will be
affected by subsequent motions of the hands. The effect of
the deformation of a point will lessen following a user-
controlled function of the distance between that point and
the initial ribbon. The extent of the Region of Influence
(RoI) and the decay function may be quickly adjusted by
the user to support large global deformations or the creation
of small details. A decay function with a plateau may be
used to ensure the preservation of fine details. As the user
continues to move the two trackers, the ribbon is moved,
rotated, bent and twisted. A space warp that interpolates
position, shape, and twist of the initial and the current
ribbon is computed and applied in real-time to the surface
being edited. This graphic feedback supports the direct
manipulation of 3D shapes. When the desired shape is
obtained, the user releases the button, hence freezing the
warp and saving the new shape for further deformations, if
desired.

Each space warp is entirely defined by two pairs of
coordinate systems. The initial pair defines an initial ribbon
used to grab a portion of the shape when the user presses a
button. The final pair is captured when the user releases the
button and defines the final ribbon. As in previously
proposed approaches [Lazarus et al. 1994], [Singh and
Fiume 1998] the vertices of the mesh that lie sufficiently
close to the initial ribbon are affected
by the warp.

In addition to providing an effective direct manipulation
paradigm, we propose a new representation of the ribbon
and a new mathematical model of the warp which offers
specific advantages over previous approaches. In particular,
the use of a circular biarc for the central wire of the ribbon
leads to an intuitive direct manipulation and a very fast
computation of all the wire points where the distance to a
particular vertex goes through a local minimum. In fact, we
prove that only two local minima exist. The use of a warp
formulation based on a screw-motion leads to natural shape
warps and permits graceful blending of the influences that
two distinct regions of the wire may have on a vertex,
therefore eliminating the tearing problem which occurs when
the two vertices of an edge are pulled in different directions
by two distinct portions of the wire.

We demonstrate the ease-of-use and power of this
formulation in an interactive system called Bender.
Although we have not used any spatial indexing to optimize
performance, Bender provides 3D graphics feedback at
more than 10 frames a second when manipulating surfaces
with about 70K triangles. We use adaptive mesh
subdivision to refine the surface in areas where the initial
tessellation may become visible.

The rest of the paper is organized as follows. In Section
2, we review relevant prior art. In Section 3, we present
implementation details and design choices. Finally, we show
results and conclude.

2. Related work

A variety of approaches have been followed for creating
and changing the shape of a surface more than one vertex at
a time. The challenge is to find a pleasing, predictable and
controllable method that can be computed in real-time.
Some approaches construct surfaces that interpolate 2D
profiles [Igarashi et al. 1999] or 3D curves [Wesche and
Seidel 2001], [Grossman et al. 2002]. Others provide means
for the direct drawing of surfaces [Schkolne et al. 2001] or
for space painting and carving [Galvean and Hughes 1991].
An alternative to these shape creation techniques is the
warping or deformation of existing shapes. Various
methods and interaction paradigms have been developed for
this purpose. Sederberg and Parry [1986] introduced the
free-form deformation (FFD), based on lattices of control
points and trivariate Bernstein polynomials. Hsu et al.
[1992] developed a version of FFD that allows direct
manipulation, while Coquillart [1990] and MacCracken and
Joy [1996] extended FFD to support more general lattices.
The technique described in this paper belongs to this group.
It is based on a grab-and-drag shape deforming operator,
allowing the direct manipulation of shape. It does not limit
the user's interaction to control points and does not restrict
the operations to be axial deformations.

Based on a designer's natural knowledge of the physical
world, we strive to approximate material properties such as
elasticity or plasticity. See [Metaxas 1996], [Gibson and
Mricht 1997] and [Gain 2000] for reviews. However,
simulated physical realism is generally too expensive for
real-time feedback. We have thus opted for a compromise,
which offers a simple and intuitive map between hand-
gestures and space warps that is independent of the
manipulated surface. The cost of computing the warp
parameters is negligible and its effect appears physically
plausible and quite predictable. Space warping and
morphing techniques are thoroughly reviewed by [Gomes et
al. 1999].

In the spirit of Forsey and Bartels' Dragon editor [1988],
Zorin et al. [1997] presented a system for multiresolution
mesh editing in which vertices at different levels of
subdivision can preserve details by using adjustment
vectors defined in local frames. Other approaches are based
on the idea of space warping, which is independent of the
representation of the underlying geometry. Barn [1984]
introduced the general space deformations twist, bend and
taper. Chang and Rockwood [1994] used a generalized de
Casteljau approach to extend Barr's technique.

Allan et al. [1989] and Bill [1994] developed systems
that displaced a selected vertex and its neighbors by a set of
decay functions. Modern software packages, such as
Discreet 3D Studio Max 4 and 5 [Discreet 2002], also allow
weighted manipulation of vertices with an adjustable decay
function. Twister [Llamas et al. 2003] uses a pair of 3D
trackers to grab two points on or near a surface and to warp
space with a weighting function that decays with increasing
range from the trackers. The work described in the present
paper can be viewed as an extension of this approach. It is
particularly useful for bending long shapes and for
operating on elongated regions of influence.

Borrel and Bechmann [1991] and Borrel and Rappoport
[1994] developed real-time techniques for computing space
warps that simultaneously interpolate several point-
displacement constraints. Previous work by Fowler [1992],
Gain [2000] and Llamas et al. [2003] support not only point
displacement constraints, but also orientation constraints on points. Milliron et al. [2002] recently introduced a general framework for geometric warps.

The Axial Deformations of Lazarus et al. [1994] used piecewise linear curves of any shape as the axis for a generalized cylinder with variable radii and local frames at key points. Wires, by Singh and Fiume [1998], takes curve based deformation techniques further, but at a higher computational cost. Balakrishnan et al’s ShapeTape [1999] uses B-spline curves to create surfaces and Wires to deform shapes using a 3D tracked and instrumented flexible rubber tape.

Turk and O’Brien [2002] approach shape modeling by constructing an implicit surface from scattered data points and normals. Several authors have developed techniques for computing piecewise polynomial surfaces that interpolate points and curves in position and possibly orientation [Hoppe et al. 1992], [Carr et al. 1997], [Bajaj et al. 1995].

Since designers are naturally capable of operating in 3D space, and since 3D surfaces are to be manipulated, we chose to explore a shape operator that provides a natural control of position and orientation of selected regions of space. We justified this decision on the basis of a well-understood interaction style [Shaw and Green 1997], [Hinckley et al 1994] and readily-available hardware [Polhemus 2002]. Using two hands allows the user to adopt both asymmetric [Guirard 1987] and symmetric operations with both hands on the surface being edited. Asymmetric operations allow the dominant hand to adjust fine detail while the non-dominant hand sets up context (position and orientation of the workpiece). Symmetric operations allow each hand to create shapes with its 6 DoF cursor. Offering natural control over six degrees of freedom per hand simplifies the design of complex warps, which will otherwise require a laborious series of 2 DoF or 1 DoF operations if only a mouse is available. Other user interface issues with high degree-of-freedom input devices are explored in Grossman et al. [2003].

3. Implementation details

In this Section, we describe how a central line of the ribbon is computed to interpolate the position and tangent directions at the end-points. Instead of using a cubic parametric curve to solve this Hermite interpolation problem, we use a biarc curve [Rossignac and Requicha 1987] made of two smoothly joined circular arcs. Then, we explain how the additional twist imposed by the two trackers is interpolated along the central line and how it affects P. We first compute the projections Q_i of P onto the initial wire. These projections are points on the wire that we call a ribbon. At every point P_i of the wire, we have one degree of freedom (which we call twist) for rotating the normal N_i to the ribbon’s surface around the wire’s tangent T_S with respect to the local Frenet coordinate system. This twist is designed to provide a smooth field of normal directions as a linear interpolation between the user-controlled twists at the two ends of the wire. The point P_S and the two unit vectors, T_S and N_S, suffice to define a local coordinate system C_S at P_S that follows the ribbon in position and orientation as s varies from 0 to 1.

Figure 3: By specifying the six degrees of freedom at each coordinate system at the end of the wire, (P0, T0, N0) and (P1, T1, N1), the user controls the shape of the wire (cyan) and the orientation (twist) of the ribbon around it. A parameter s defines a point PS on the wire and two orthogonal vectors, TS and NS.

Hence, for each vertex P of a triangle mesh we compute how the warp affects P. We first compute the projections Q_i of P onto the initial wire. These projections are points on the wire at which the distance to P goes through a local minimum. For all Q_i that are closer to P than a user-prescribed threshold, we compute a displacement vector W_i. The displacement vector W_i is the result of moving P by a fraction f_i of a screw motion M_i. f_i is computed as a function of the distance ||PQ||.

The screw motion M_i is computed as follows. From the position of Q_i along the wire, we compute the corresponding parameter s. Then, we compute the corresponding coordinate systems C_s and C’s on the initial and final wire. M_i is defined as the unique minimal screw motion interpolating between them. We apply a fraction f_i of M_i to P and compute the displacement vector W_i.

3.2. Wire construction

The wire is defined by the positions and tangent directions of its two ends (P_0, T_0) and (P_1, T_1). We wish to create a smooth 3D curve that interpolates the end-conditions and is formed by two circular arcs that are smoothly joined at some point J. The wire is completely defined by computing two scalars, a and b, which define
the four normals, \( N_0, N'_0, N_1, \) and \( N'_1 \) are coplanar and
in the 3D situation, simply fold the paper along the \( I_0I_1 \).

Two triangles are not coplanar. To obtain an example of a
plane, the construction holds in three dimensions, when the
tangent. Although for clarity Figure 4 was drawn in the
plane of the second arc and \( N_1 \). Let \( e \) denote the angle
between \( N_0 \) and \( N'_1 \). If we rotate \( P_0 \) to follow the first arc,
Similarly for \( N_1 \) and \( N'_1 \). In this final configuration,
the point \( J=(bI_0+aI_1)/(|bI_0+aI_1|) \). The triangle \( (J, I_1, P_1) \)
is isosceles and inscribes a second circular
arc that starts at \( J \) where it is tangent to \( I_0I_1 \) and ends at \( P_1 \).
Both arcs meet at \( J \) with a common
tangent. Although for clarity Figure 4 was drawn in the
plane, the construction holds in three dimensions, when
the two triangles are not coplanar. To obtain an example of a
3D situation, simply fold the paper along the \( I_0I_1 \).

Following [Rossignac and Requicha 1987], we chose
\( a=b \). This choice leads to an efficient calculation and yields
excellent shapes for the wire. In fact, in most situations the biarc
is very close to a cubic parametric curve with the
same end-conditions.

To compute the parameter \( a \), we must solve \( ||(P_0+aT_0)-(P_1-aT_1)||=2a \), which yields a second degree equation in \( a \):
\[
S(S-2a(T^2)+a^2(T^4-4)=0),
\]
where \( S=P_1-P_0 \) and \( T=T_0+T_1 \).

In the general case, when \( T^2\neq 4 \), we use
\[
a=(\sqrt{(S^2(T^2)+S(S)(4-T^4)-(S^2(T)^2)})/(4-T^2),
\]
which produces arcs of less than 180 degrees. In the special
case where \( T^2=4 \) and \( T_1=T_2 \), we use two semi-circles, as
discussed in [Rossignac and Requicha 1987].

3.3. Distributing the twist of the ribbon along the biarc

Each arc lies in a plane. The left-hand tracker defines the
normal \( N_0 \) to the ribbon at \( P_0 \). We record the angle \( \alpha_0 \)
between \( N_0 \) and the normal \( N'_0 \) to the plane of the first arc.
Similarly, we record the angle \( \alpha_1 \) between the normal \( N'_1 \)
to the plane of the second arc and \( N_1 \). Let \( e \) denote the angle
between \( N'_0 \) and \( N'_1 \). If we rotate \( P_0 \) to follow the first arc,
and wished to keep the associated normals \( N_0 \) and \( N'_0 \) in
constant orientation with respect to the local Frenet
triadhedron of the first arc, we would arrive at \( J \) with both
normals parallel to the plane through \( J \) and orthogonal to
\( I_0I_1 \). Similarly for \( N_1 \) and \( N'_1 \). In this final configuration,
the four normals, \( N_0, N'_0, N_1, \) and \( N'_1 \) are coplanar and
their relative orientations are given by the three angles \( \alpha_0, \alpha_1, \) and \( e \). In particular, the angle between \( N_0 \) and \( N_1 \) is then
\( \alpha_0+e+\alpha_1 \). Because we wish to obtain a smooth field of
normals starting at \( N_0 \) and finishing at \( N_1 \), we distribute the
difference linearly, and twist the local coordinate system \( C_0 \)
along the tangent \( T_i \) by an angle equal to \( s(\alpha_0+\alpha_1+e) \), where
the parameter \( s \) is equal to 0 at \( C_0 \) and equal to 1 at \( C_1 \). In
practice, for points on the first arc, we twist \( C_0 \) by
\( s(\alpha_0+\alpha_1+e) \) and then rotate it around the axis of the first arc.

For points on the second arc, we rotate \( C_1 \) by \( (1-s)(\alpha_0+\alpha_1+e) \)
and then rotate it backward around the axis of the second arc.
Figure 5 shows the ribbon for the same wire with and
without twisting.

The choice of the \( s \) parameterization affects not only the
twist of the ribbon around its wire, but also the
correspondence between two wires, and hence the warp.
We have explored two parameterizations. The first one is
an arc-length parameterization for the whole biarc. The
second one uses an arc-length parameterization for each
arc, forcing \( s=0.5 \) at the junction.

A parameterization that maps the first arc of the initial
wire onto the first arc of the final wire is slightly simpler to
compute. However it tends to produce a non-uniform
stretching of space when the arc-length ratios between the
first and the second arc differ significantly in each wire. We
have therefore opted to use the global arc-length
parameterization.

3.4. Projection of points onto a biarc

Consider the biarc of the initial ribbon and a point \( P \). As
argued above, we want to compute all points \( Q \), on the biarc
where the distance between \( P \) and the biarc goes through a
local minimum. We will call them the projections of \( P \). We
explain in this section how to compute these projections
quickly and prove that when \( P \) is closer to the wire than the
minimum of the bi-arc radii, at most two such projections
exist.

Consider a circle with center \( O \), radius \( r \), and normal \( N \).
Let \( Q \) be the point on the circle that is closest to \( P \). We
compute \( Q \) by first computing the normal projection
\( R=P+\alpha r \) of \( P \) onto the plane of the arc. Then, \( Q \) is
obtained by displacing \( O \) by \( r \) towards \( R \). Hence
\( Q=O+\alpha r OR/||OR|| \). If \( Q \) lies inside the arc, it is a projection
of \( P \). Note that if such a normal projection exists on the arc,
then all other points of the arc lie further away from \( P \),
including the endpoints of the arc. When \( Q \) is not on the
arc, we consider the free end of the arc, the end of the biarc,
as a candidate projection of \( P \). If, at that free end, the biarc
moves away from \( P \), then it is a projection of \( P \), i.e., a local
minimum of the distance. Notice that if the other end of the
arc were closer to \( P \), it would not be the local minimum for
the biarc, since by sliding by an infinitely small amount
onto the other arc, it would approach \( P \).

An example where \( P \) has a normal projection inside one
arc and on the endpoint of the other arc is shown in Figure 6.
A closest projection $Q_0$ of $P$ lies inside the first arc. A second closest projection $Q_1$ lies at the tip of the second arc.

When two projections are returned and both are further away from $P$ than the radius of the region of influence of the warp, $P$ is not affected by the warp. When a single projection is close enough to $P$, we compute its parameter $s$ on the initial ribbon’s biarc and use $s$ and $||PQ||$ to compute a warp. In the cases when the projections $Q_0$ and $Q_1$ reported for both arcs are within tolerance from $P$, we compute two warps and blend them. The merit of this solution is discussed below.

3.5. Deforming a point using screw motion

Given the projection $Q$ of $P$ onto the first arc of the initial wire, we compute its parameter $s$ using the ratio of angles $(O_0P_0,O_0Q)$ and $(O_0P_0,O_0J)$ and the ratio of the arc-length of both arcs. A similar approach is used when $Q$ lies on the second arc. We then compute the two coordinates systems, $C_s$ on the initial wire and $C_s'$ on the final wire. They are used as input to compute a fixed point $A$, an axis direction $K$, a total rotation angle $\beta$, and a total displacement $d$. These four parameters define a screw motion that transforms $C_s$ into $C_s'$ by performing a translation by $fdK$ and a rotation by angle $f\beta$ around the axis of the screw that has direction $K$ and passes through $A$.

3.6. Preventing the tearing of space

When there are two projections $Q_0$ and $Q_1$ on the arc where the distance to $P$ is locally minimal and when both fall within the Region of Influence of the initial wire, we must take them both into account. Otherwise, a tearing of space may occur. To explain the tearing, suppose that points $P$ and $P'$ are the endpoints of an edge of the mesh. Suppose that the $s$ parameter of the closest projection $Q$ of $P$ is very different from the $s'$ parameter of the closest projection $Q'$ of $P'$. If we were to use the screw associated with a single projection, we would use similar fractions (decay weights) $f_0$ for both $P$ and $P'$, but their screws could be very different and may pull them away if, for example, the final wire increases the distance between $Q$ and $Q'$.

3.7. Choosing decay functions

Depending on the type of deformation we want to achieve, different decay functions $F$ may be preferred. Following [Jin 2000], [Lazarus et al. 1994], we let the user switch between a bell-shaped curve and a plateau function (Figure 8), which permits to preserve the shape inside a tube around the wire when the relation between the corresponding portion of the initial and final ribbons is a rigid body transformation. Such relations are maintained when performing warps that achieve rigid bending operation of limbs or tubes.

3.8. Maintaining continuity

In this subsection, we discuss a modification to the screw computation, which was necessary to ensure the continuity of the warp through space.

The screw motion interpolation used in [Llamas et al. 2003] always generates screws of minimal angle, which is always less than 180 degrees. Consider two points $P_s$ and $P_s'$ traveling simultaneously on the initial and final wire.
Assume that they move towards each other, then go through a singular situation where their velocities are parallel, and finally diverge. As we pass through the singular situation, the orientation of the screw axis $K$ is reversed. The displacement values and the direction of rotation are also reversed. This flip produces a discontinuity in the pencil of helix trajectories taken by points of the initial wire as they are warped (Figure 9). We detect these situations using the sign of the dot product of consecutive $K$ vectors. To prevent the discontinuity, we simply revert the flip. This correction results in rotation angles $\beta$ that may temporarily exceed 180 degrees. We compute the angle as before, and simply replace it by $(\beta - 2\pi)$. The $K$ axis is reversed and the distance $d$ negated. We do this change at each singular point.

When no correction is needed, we use the natural direction of $K$ given by the original construction in [Llamas et al. 2003]. When one or more corrections are needed, the user may press a button to toggle between the two possibilities, the one defined at $s=0$ by the original construction of $K$ and the one where all the $K$ directions are reversed.

Note however, that neither the flip of $K$ nor the blending of screws associated with two projections will solve the problem of space inversion that is inherent to all wire-based warps and may occur when the radius of the region of influence is larger than the minimum radius of curvature of the wire. Modeling the wire as a biarc makes it trivial to detect these situations because the radius of curvature is known for each arc. Thus, we have considered reversing the radius of the region of influence automatically to avoid such inversions. However, because undesirable space inversions are easy to detect visually and avoid with direct manipulation, we have opted not to perform the automatic adjustment to avoid surprising the user with the occasional incorrect choice.

3.9. Adaptive subdivision

When the mesh is stretched by a warp, the density of its tessellation may no longer be sufficient to produce a smooth warped surface (Figure 10). We use a simple and very efficient technique for adaptively subdividing the surface wherever appropriate. After each warp, when the user freezes the shape, the system starts an adaptive subdivision process and replaces the warped surface with a smoother one. Note that our subdivision simply splits some triangles into 2, 3, or 4 smaller triangles without changing the initial shape. Contrary to subdivision procedures that smoothen the shape, in our implementation, the new vertices are positioned exactly in the middle of the old edges and the old vertices are not adjusted. Tucking in of the old vertices as a Loop subdivision [Loop 1987] would do or bulging out the edges as a Butterfly subdivision [Dyn et al. 1990] would do is unnecessary, if the initial shape was sufficiently smooth. Hence, we do not have to respect restrictions on the subdivision levels between neighboring triangles.

Let the term initial mesh denote the mesh before the current warp, which deforms it into a final mesh. Note that the initial mesh may have been produced by a series of previous warps and subdivisions. Each edge of the initial mesh is tested and marked if subdivision is required. Then, each marked edge is split at its midpoint and each triangle with $m$ marked edges is subdivided into $m+1$ triangles, using a standard split. This simple approach guarantees preservation of connectivity and does not introduce T-junctions. To test whether an edge should be marked, we compute the distance between the midpoint of its warped vertices and the warped midpoint of its vertices. If that distance exceeds a threshold, we mark the edge. The process is repeated until no more edges need to split or until the user starts a new warp.

This simple approach works well in practice and is very fast. However, it does not guarantee detection of all cases where subdivision is needed. For example, a local stretch occurring inside a triangle that does not affect the edge-midpoints could remain undetected. The subdivision may also lead to overly long triangles in some areas of the model, so a criterion to deal with these could be added. The adaptive subdivision method of Kobbelt et al. [2000] deals with this, leading to an isotropic tessellation. By not dealing with these triangles we obtain an anisotropic tessellation, which may be preferred in many cases.

Finally, a simplification procedure might be desirable in some instances, where the user desires to eliminate excessive sampling previously introduced. Such procedures could be called upon request, targeting specific regions of the mesh selected by the user, or instead, could be run after a deformation, to coarsen areas that have been flattened. This idea is integral to Gain’s adaptive refinement and decimation approach [Gain 2000].

4. Concluding remarks

By combining a biarc with the concept of a twisted ribbon around it and with a screw-based motion that interpolates corresponding portions of the initial and final ribbons, we have created a new formulation of a space warp
that is completely defined by four coordinate systems. We
have developed a prototype 3D user interface for the direct
manipulation of these coordinate systems through the use of
two Polhemus trackers. We show that the approach makes
it easy to design or bend, twist, or warp a variety of shapes.

Before opting for this approach, we have explored other
formulations for the wire and for the warp. For instance,
using a cubic curve or a helix as a wire results in a
significantly more expensive calculation of the vertex
projections on the wire and could potentially generate a
larger number of projections. Hence, we have chosen to use
a circular biarc for three reasons. First, it provides the user
with a very intuitive control of the shape of the wire.
Second, it significantly reduces the cost of computing the
projection \( Q \) of a point \( P \) and the associated coordinate
system, when compared to a helix or to a cubic polynomial
curve [Schneider 1990]. Third, our choice ensures that there
are at most two locally closest projections \( Q_0 \) and \( Q_1 \) of any
point \( P \) onto a biarc. Based on this observation, we are able
to develop a simple technique for avoiding the tearing of
space that happens when two neighboring surface points \( P \)
and \( P' \) each have locally closest projections that are distant
along the wire. For more than two projection points,
computing a blend that satisfies the constraints imposed by
the biarc is not always possible. Milliron et al. [2002]
proposed extending Wires [Singh and Fiume 1998] with a
blending approach in which the entire curve has some effect
on each deformed point, however the method is
approximating instead of interpolating, thus constraints are
not satisfied.

We have also explored using transformations that are not
screw motions for the interpolation between a coordinate
system on the initial wire and its counterpart on the final
wire. In particular, we have explored the use of a biarc-
driven trajectory. We have concluded that the combination
presented in this paper is a good compromise between
computational cost and flexibility, producing natural warps,
avoiding undesired bulges, and yielding a very fast
implementation.

There are still some ideas that may be worth examining
in the future. Bender could be extended with the ability to
snap to an arbitrary surface. The biarc curve would hardly
ever follow the surface exactly, but it would very well
facilitate the user’s task. A decay function based on
godesic distance might be better suited in this situation, for
instance when bending a curved rim of a bowl (as in Figure
11), as shown by Bendels and Klein [2003]. The biarc
curve would have to be projected and sampled onto the
surface for this computation, but the screw motion
transformations would still be defined by the starting and
ending biars.

The design choices we made lead to an intuitive and
predictable deformation, even when the changes in the
shape and twist of the initial and final ribbons are
significant. Although the individual components of the
deformation are not new, the combination we present is not
trivial and is the result of extensive research and evaluation
of the tradeoffs involved. The resultant deformation model
permits a real-time direct manipulation, even for shapes of
significant complexity. For example, our current,
unoptimized implementation produces 10 frames per
second with models of about 70,000 triangles.

Figure 11 illustrates some of the shape deformations that
may be trivially achieved by a single Bender warp.

References

Direct Manipulation of Polygon Meshes. New Advances in Computer
Graphics (Proceedings of CG International ’89) (June), 451-469.

ACM Transactions on Computer Graphics 14, 2, 103-133.

BALKRISHNAN, R., FITZMAURICE, G.W., KURTENBACH, G., SINGH, K.
1999. Exploring interactive curve and surface manipulation using a
bend and twist sensitive input strip. In Proceeding of the 1999

In Computer Graphics (Proceedings of ACM SIGGRAPH 84), ACM
SIGGRAPH, 21-30.