Explicating Implicit Surfaces

Brian Wyvill

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail addresses: blob@cpsc.ucalgary.ca

ABSTRACT

In this paper I present a review of implicit surface techniques along with a summary of current problems and possible solutions. Implicit surface models can be made to blend and change their shape. Such properties as this make these models very useful for animation. I describe various interactive modelling and animation techniques such as the use of metamorphosis and warping. Implicit surfaces have been used to represent natural phenomena such as biological shapes and in this paper a new technique is presented for simulating glowing objects. Finally some thoughts are given to the future direction of implicit surface research.

KEYWORDS: modelling, implicit surfaces, skeletons, animation.

INTRODUCTION

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) [18] and *MetaBalls* (Nishimura) [12]. 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]. In this paper I have concentrated on implicit surfaces built from skeletons. A skeleton is composed of a number of skeletal elements. A scalar field is defined around a skeletal element. The shape and a variety of properties of a skeletal element are discussed below. 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 this paper I have outlined the major trends and techniques using skeletal implicit surfaces, various problems of the technique and areas of future research are also discussed. Following this introduction the paper is organised as follows; section one deals with modelling techniques, section two with rendering and section three with animation. In each section a brief overview of the subject is presented followed by details of areas where active research is ongoing.

SKELETAL IMPLICIT SURFACE MODELS

The basic idea is that a model can be built from a primitive skeleton by combining elements such as points, lines, polygons, circles and splines. A surface representing a blended offset from the skeletal elements, is calculated and visualized. The skeletal elements are linked hierarchically. At each frame an implicit surface encloses the skeleton using the techniques described in [4]. In general, any three dimensional object can be a part of the skeleton, as long as it is possible to determine the distance from a given point in space to the object. Skeletons are useful for several reasons:

- Skeletons provide intuitive representation for many natural objects.
- Skeletons themselves are easily manipulated and displayed.
- Skeletons provide a more concise representation than parametric surfaces.
- As in Constructive Solid Geometry (CSG), complex shapes can be modeled with few elements. But unlike CSG it is much easier to add new primitive types. (CSG requires $O(n^2)$ intersection algorithms for n primitive types, implicit surfaces require $O(n)$ distance algorithms).

The skeleton is surrounded by a scalar field $F_{total}(P)$ (equation 1). The intensity of the field being the highest on the skeleton, and decreasing with distance from the skeleton. The function $F_{total}(P)$ relates the field value (intensity) to distance from the skeleton has an impact on the shape of the surface, and determines how separate surfaces blend together (see [8]). 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 are added to find the potential at some chosen point. (Values can be negative or positive). The value at some point in space is calculated as follows:

$$F_{total}(P) = \sum_{i=1}^{i=n} c_i F_i(r_i) \tag{1}$$

where P is a point in space

$F_{total}(P)$ is the value of the field at P

n is the number of skeletal elements

c_i is a scalar value (used for positive or negative elements)

F_i is the blending function of the i_{th} element

r_i is the distance from P to the nearest point Q_i on the i_{th} element.

The evaluation of $F_{total}(P)$ has two steps. The first step involves finding the nearest point Q_i on the skeletal element to the given query point P and calculating the distance between them. This procedure depends on the geometry of the skeletal element and can be very simple (trivial in the case of a point skeleton), or quite complex in the case of spline curves and patches, when an iterative or numerical method is necessary. The second step involves evaluation of the blending function which may be modified by noise or other perturbation function as described in [8]. The warping described below is separate from these modification functions. The surface is controlled by applying local or global transformations, such as scaling, translation, and rotation, to the elements of the skeleton, and by changing the blending functions.

MODELLING PROBLEMS

Before good interactive design tools can be built, solutions need to be found to a number of outstanding problems. Firstly, the method is not localised sufficiently. Moving a skeletal element (or primitive) has a global effect on the field. Some progress towards building a hierarchical scheme, where primitives may be subdivided and their children have limited effect has been made ([7]) but as yet, the design tools do not have the intuitive feel of Forsey's hierarchical parametric surface editor ([5]). A second problem is known as the *bulging problem*. fields from neighbouring skeletal elements are summed and produce a bulge where their combined values are equal to the iso-value. Jules Bloomenthal and Ken Shoemake offered a solution to this problem ([3]), however their convolution surface technique is relatively slow and involves a lengthy pre-processing step at a discrete resolution. Another possibility would be to use some other technique as an alternative to taking the sum of the implicit values. For example given two primitives, A and B, for some point P the values could be combined thus: $V_{total} = (V_A + V_B)/(V_A^2 + V_B^2)$ The bulge is reduced using this technique, although not eliminated. Another source of further research is the unwanted blending problem. A good example is a human hand. The fingers blend at their roots but not along their length. This has been partially solved in the *Meta-Editor* by introducing the concept of a primitive which has an asymmetric effect on the surrounding field. Another approach is to find the implicit value by taking the maximum of the contributions of the surrounding primitives. This technique was first introduced by Thad Bier see [17]. This produces no blending between primitives. Blending can thus be designated between groups of primitives in a graph like structure [16]. Figure 1 demonstrates this technique, as can be seen, collision detection between the fingers has not been applied, blending is minimised but after polygonization the fingers are interpenetrating in places.

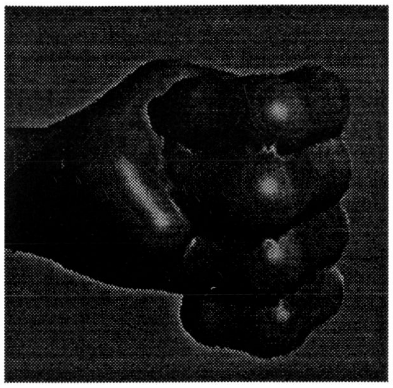


Figure 1: Unblending Primitives (Courtesy Andrew Guy University of Calgary

RENDERING

Skeletal Implicit surfaces are defined by *black box functions* which given a point provide a value and in the case of points on the surface, a surface normal. These surfaces can be visualized by ray tracing, finding the ray surface intersection by one of a number of numerical techniques. A good survey of these techniques is given in [17]. A popular method of rendering implicit surfaces is to first convert the surface to a polygonal approximation. A survey of methods of doing this *polygonization* is detailed in [11]. In an interactive environment, the surface must be visualized as fast as possible, to keep up with changes to a model entered interactively by the user. An additional advantage of the polygonization approach is the availability of a full 3-D approximation of the implicit surface which allows for fast viewing from arbitrary directions.

Most currently existing techniques for the polygonization of implicit surfaces are based on data structures that allow spatial indexing: either a voxel-based structure ([2]) or the hash-table structure of ([18]) may be used. Some inherent disadvantages of these data structures exist. Firstly, the data structure comprises a partitioning of the space rather than a tesselation of the surfaces to be polygonized. Especially in the case of animation (e.g. in the computer animation "The great train rubbery", [15]), this is likely to cause geometric artifacts that are fixed with respect to space, thus moving in an incoherent way over every moving surface.

Second, there is an apparent mismatch between the number of triangles that is generated by these algorithms and the complexity of the surface that is approximated: even relatively smooth and flat segments of an implicit surface usually result in large amounts of facets. Bloomental ([2]) uses an adaptive version of the spatial indexing data structures, an octree in order to reduce the amount of polygons produced in tesselating an implicit surface. This indeed reduces the amount of polygons generated, but full advantage of large cells can only be taken if the flat regions of the surface happen to fall entirely within the appropriate octants. The algorithm proves in practice to be considerably slower than the uniform voxel algorithm of [18], and is very complicated to implement.

SHRINKWRAP

A new, fast, adaptive algorithm called; *ShrinkWrap*, was offered by [13]. This algorithm took the following approach. First, when constructing an adaptive tesselation for a curved surface, the most obvious parameter to use as an indicator for the local tesselation-resolution is the local curvature of the surface. In many cases, a measure for this local curvature (the Gaussian curvature), can be computed analytically. This is also true for implicit (or equi-potential) surfaces in the case of $1/r$-type potentials. When tesselating the surface, however, i.e. approximating the surface by a discrete set of samples plus some connecting topology, the interpretation of the value of this curvature in the sampled points is not at all clear. For instance, the surface may be highly curved between two adjacent sample vertices, but if it happens to be flat in these vertices we won't know and the tesselation is likely to miss this curved feature. In this algorithm the tesselation consists of a mesh of triangles, but the error analysis takes place on the edges of the triangles (the chords) rather than on the triangles themselves. Chords are considered as approximations of segments of curves in the implicit surface.

The following criteria lead to a definition of an acceptable surface. α, $\mathcal{L}$ and L_{max} are real-valued parameters that are used to characterize the algorithm.

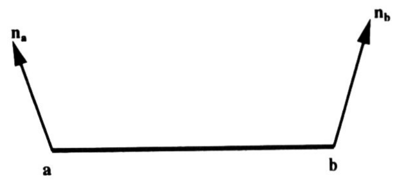


Figure 2: chord criteria

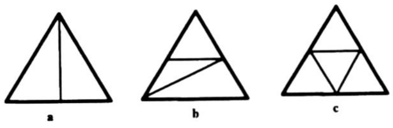


Figure 3: triangle subdivision

- the surface is given by $f(r) = 0$ where $r \in \mathbf{R}^3$;
- a chord is a tuple $(a, b, n_a, n_b) \in \mathbf{R}^3 \times \mathbf{R}^3 \times \mathbf{R}^3 \times \mathbf{R}^3$ where $f(a) = f(b) = 0$ and $n_a = \nabla f(a)$ and $n_b = \nabla f(b)$;
- the surface is called acceptable iff for every chord on the surface $\angle(n_a, n_b) \leq \beta|a - b|$ where $\angle(p, q)$ is the angle between p and q;
- a chord is called acceptable iff $\angle(n_a, n_b) \leq \alpha$ and $|a - b| \leq L_{max}$;
- a triangle, consisting of three chords is acceptable iff all three chords are acceptable.

The parameter $\mathcal{L}$ is essentially the Lipschitz parameter of the surface to be tessalated. The Lipschitz criterion bounds the variation of the derivative of a function between neighbour points in the domain of that function. The Lipschitz criterion has been used in 1989 by Kalra and Barr ([9]) to compute ray-intersections with implicit surfaces with a guaranteed accuracy. Before that date, B. von Herzen ([14]) studied its application in computer graphics.

The algorithm attempts to create a tesselation which consists of acceptable triangles. The simplest closed polygon mesh is a tetrahedron, consisting of 4 vertices, 4 triangles and 6 chords. If one or more chords are unacceptable, they have to be split. Figure 3 shows a splitting scheme which illustrates how a triangle can be subdivided into smaller triangles. In *a* one of the chords is subdivided; in *c* three chords are subdivided and in *b*, one of the two possible ways of subdividing two chords. This subdivision scheme can be used to assign surface coordinates to all newly created vertices to be the average of the surface coordinates in the two extreme vertices of the split edge. The new midpoint is then forced onto the surface by an iterative method. (see [13]).

One of the problems is that if the surface is too convoluted, the iterative algorithm mentioned above will not be able to converge onto the surface. However we observe that equipotential surfaces

$$\{r \in \mathbf{R}^3 | \sum_i \frac{\rho_i}{|r - R_i|} = V_0\}$$

with $V_0 < 1$ have a shape which is less involved, whereas equi-potential surfaces with $V_0 > 1$ are more involved. An extreme case of the first example is $V_0 = 0$, which produces a sphere with an infinitely large radius. We take advantage of this observation and start with a tetrahedron which is compared to a surface with a low value of V_0. The triangles are then split according to the acceptability criteria and the resulting mesh then compared with a surface where the value V_0 is increased. Thus we have an iterative process which creates gradually more involved surfaces. (See Plate 1).

The Breakfast Algorithm

The above process does not solve all the problems. As the values of V_0 are increased the surface can break off into separate parts. The above algorithm cannot cope with holes (as in a toroidal model) or separate manifolds. The *Breakfast Algorithm* partially solves the problem by using the voxel algorithm of [18] to identify separate surfaces and then applies *Shrinkwrap* to each of these. Voxels are classified as *inside*, all vertices have an implicit value > the iso-value, *outside*, all vertices have an implicit value < the iso-value, or *containing* in which at least one vertex is the opposite sign. A search algorithm is applied to produce groups of voxels that have neighbours in the *containing* category. These groups of voxels are polygonized and form the starting mesh for applying the *Shrinkwrap Algorithm.* At first sight it seems that the *Breakfast Algorithm* combines the best of both the previous approaches. However toroidal objects are still a problem and the number of manifold surfaces emerging from the first pass is dependent on the size of the voxels. Further experiments have to be done to establish the efficiency of these algorithms.

RAY TRACING GLOWING OBJECTS

Making an object glow is a useful effect in computer animation. This can be done relatively easily with implicit surface models. A glow should be seen surrounding an object and fade to zero at some distance away. A value for the brightness of the glow can be obtained as a function of the field in which the model is defined. This has been implemented as a part of a ray tracer. The nearest distance between each ray to each primitive ellipsoid is calculated. The point on the ray is shown as P in figure 4. One way to calculate the glow is to take the value of P corresponding to the shortest of these distances and pass it to equation 1. The implicit value, v, can

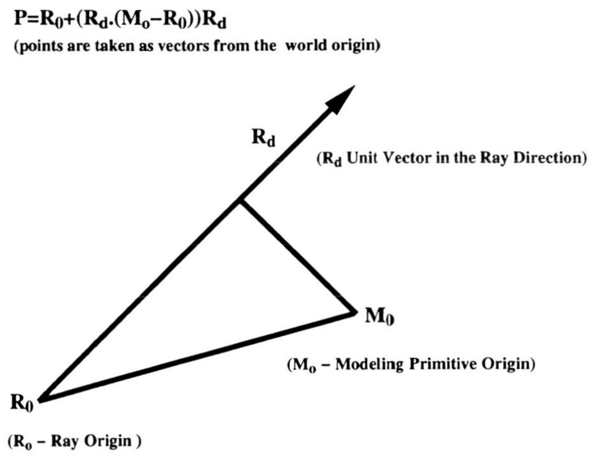

P=R0+(Rd.(Mo−R0))Rd
(points are taken as vectors from the world origin)
Rd
(Rd Unit Vector in the Ray Direction)
M0
(Mo – Modeling Primitive Origin)
R0
(Ro – Ray Origin)

Figure 4: Finding the nearest distance from a ray to an implicit surface primitive

be calculated and used to obtain the brightness of the glow as follows:

$$G_\lambda = m_\lambda v' \qquad (2)$$

where

$$v' = \begin{cases} 0 & \text{if } v < 0 \\ 0.5 & \text{if } v > 0.5 \\ v & \text{otherwise} \end{cases} \qquad (3)$$

and $m_\lambda \epsilon [0, 1]$ are scalars controlling the intensity of independent wavelengths.

There is a problem with this procedure consider two neighbouring rays, R_a and R_b. The nearest primitive to R_a may be different from the nearest primitive to R_b, thus returning two different points with correspondingly different implicit values. Thus causing a discontinuity in the glow. An obvious method around this problem would be to find the closest distance for each primitive and sum the implicit values contributed by each primitive for each ray. Since each primitive contributes a value between 0 and 1, the sum can be normalized by simply dividing by the number of primitives. This procedure causes the glow to be very dense near to areas where there are lots of primitives grouped together, but far too sparse in areas where there are few primitives. For example plate 3 shows a dinosaur model next to its glowing counterpart. The tail section received no-glow by this method. To achieve the even glow in plate 3 the points along the ray representing the closest point to each primitive are selected as above. The implicit values are calculated as before but

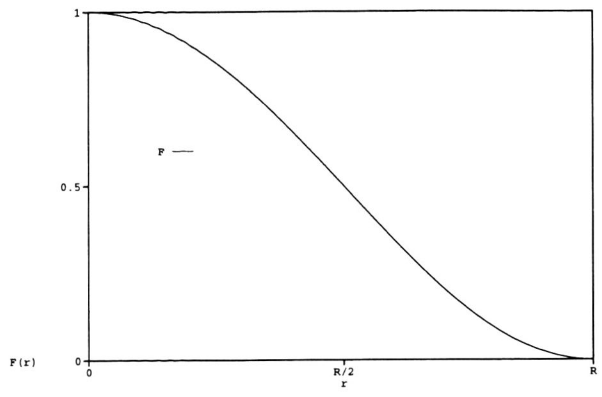


Figure 5: Cubic Blending Function $F_{cub}(r) = -\frac{4}{9}\frac{r^6}{R^6} + \frac{17}{9}\frac{r^4}{R^4} - \frac{22}{9}\frac{r^2}{R^2} + 1$

only the N highest values are used for the glow calulation. It has been found empirically for our models that $N = 3$ is a reasonable number to choose. With N<3, discontinuities appear, with N>3, no glow is seen where there are few primitives.

Plate 4 shows two rows of merging sphere primitives. The top row has been raytraced and a texture applied. The bottom row is the same set of textured primitives with a glow added. The advantage of using implicit surface models is that the glow can be calculated directly from the field as shown above. The shape of the glow follows the zero contour. Values of the field greater than zero are brighter than the background. It is also possible to change the shape of the glow, by altering the blending function used in the expression for $F_{total}(P)$. For the dinosaurs we use the cubic from [18] see 5. The glow field can be increased by using a function that falls to zero more slowly than the blending function for the train model itself.

ANIMATION

Various animation techniques have been devised to make use of the blending and other properties of implicit surface models. For example:

- Path Following
- Negative Primitives
- Metamorphosis
- Collision Detection
- Warping

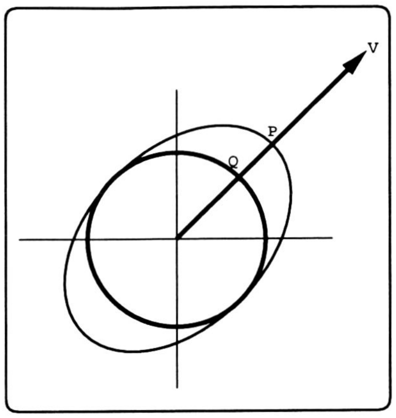


Figure 6: P is warped to Q. The original sphere is warped to an ellipsoid.

The first three of these techniques are detailed in [17] and the 4th in [6]

A useful tool in our system is the ability to distort the shape of a surface, by warping space around it. 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 animations.

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

$$F_{total}(P) = \sum_{i=1}^{i=n} c_i F_i(r_i)$$

$$r_i = f_i(P) = dist(w_i(P), Q_i)$$

where $w_i(P)$ is the position of the point 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, P is first warped to the appropriate position before evaluating the distance function.

As a first example, we study a warp function $w_i(P)$, which warps a point P to a point Q along a given vector v; it may be given by the vector equation:

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

where p is $P - S_{0,i}$.

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

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

To understand how this affects the iso-surface consider 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 Q which is on the iso-surface. The value returned for P by the implicit function is the value that would have been returned for Q if warping were not in effect. In this case that value is the iso-value. Thus P becomes a point on the surface and the sphere is warped to an ellipsoid. (See Figure 6)

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. John Lasseter notes [10] 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 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 shown in the example below. Plate 2 shows some frames selected from an animation showing the putty like ball bouncing. This is implemented in the warp function in the following manner:

Let P be a point in space at which the implicit function is to be evaluated. For simplicity, assume the collision plane to be the plane $y = 0$.

$$w(P) = \begin{cases} P - \beta\hat{v}(v.p) - p\downarrow\gamma S & \text{if } P.y > 0 \\ (P.x, \infty, P.z) & \text{otherwise} \end{cases}$$

Here

$p = P - S_0$

S_0 = the origin of the skeleton

$p\downarrow = (P.x, 0, P.z)$

$\gamma = h\left(\frac{S_0.y}{y_0}\right) h\left(\frac{P.y}{y_0}\right)$

$h(t)$ = A decreasing differentiable function (e.g.a cubic polynomial) such that:

$$\begin{aligned} h(t) &= \begin{cases} 1 & \text{for } t \leq 0 \\ 0 & \text{for } t \geq 1 \end{cases} \\ \beta &= \mathit{clamp}(1 - 2.0\gamma) \\ S &= a \text{ parameter, typically around } 0.5 \end{aligned}$$

The interpretation of the terms is as follows:

$(P.x, \infty, P.z)$
guarantees that no part of the object protrudes below the collision plane.

$-p\downarrow\gamma S$
accounts for spreading out the lower part of the object (squashing). The vector $p\downarrow$ is parallel to the collision plane indicating that squashing should be a horizontally directed effect. The factor γ assures that squashing increases near the collision plane. No squash is applied if the center of the objects is higher than y_0 or if P is higher than y_0. The parameter S controls the amount of squashing.

$-\beta\hat{v}(v.p)$
is a modified version of the simple linear velocity warping as discussed in section 4. The factor β is introduced to quench the velocity warping near the impact point, to avoid discontinuities in the warp function.
Indeed: at the point of impact, v undergoes a discontinuous change,
$(v.x, v.y, v.z) \rightarrow (v.x, -v.y, v.z)$. This would cause a discontinuous warp function if it was not compensated. The factor 2.0 in the expression for β has been found experimentally to be a reasonable value. The function *clamp(..)* clamps its argument value between 0 and 1.

This approach initially aligns the warp vector with the velocity vector of the ball in the bouncing ball example. When the impact occurs the ball will start to deform in a non- linear oblate fashion according to the $-p\downarrow\gamma S$ term. At the same time the linear prolate deformation (due to $-\beta\hat{v}(.p)$) subsides. Our implicit surface models can be a collection of skeletal elements, rather than a single spherical element, such as the ball. Thus we can easily apply this non-linear warp to a complex shape. In Plate 2 some frames are shown from an animation which applies this method to a bouncing ball (one soft primitive $S = 0.25$). The ball is distorted into an ellipsoid (Plate 2a) whose long axis grows as the vertical velocity increases (Plate 2b). On impact the ball warps (Plate 2c/d) into a bulging shape. As the ball bounces it regains its ellipsoid shape and as its vertical velocity decreases the ball remains stretched along the horizontal axis by an amount corresponding to its horizontal velocity. The technique easily extends to skeletons consisting of many primitives. In Plate 2, the same method is shown to work for Nelson, the jumping bear (consisting of 25 soft primitives). The slug in Plate 2 is in fact a warp applied to three ellipsoid primitives. Warping can be applied in space or time and may be non-linear, for example, a warp can be applied:

- To the space in which a model exists, then move the model.
- To the space over time, the model will change with time.

We have tried several different types of warp and the oustanding problem is to present warping to the animator with a consistent user interface, so that custom warps may be designed.

CONCLUSIONS

In this paper some techniques for modelling, rendering and animating implicit surfaces have been presented. Attention has been focused on previously unpublished techniques such as glowing objects and the *breakfast algorithm.* Skeletal Implicit Surface techniques have proved useful for designing models but still await widespread use in the design community. Before this can happen, better interactive tools need to be built for the designer and animator.

ACKNOWLEDGMENTS

I would like to thank the many students who have contributed so greatly to this research. I would also like to thank Jules Bloomenthal for his encouragement and ideas over the years and to Kees van Overveld, who is my co-worker in developing the *Shrinkwrap* and *Breakfast* algorithms. I am particularly in debt to my brother, friend and colleague, Geoff Wyvill, who started the whole thing off by solving a problem in scientific visualization, the solution to which turned out to be so useful for building models.

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada in the form of a research grant and equipment grants.

References

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

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

[3] Jules Bloomenthal and Ken Shoemake. Convolution Surfaces. volume 25, pages 251–256, August 1991.

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

[5] David R. Forsey and Richard H.Bartels. Hierarchical B-spline refinement. *Computer Graphics (Proc. SIGGRAPH 88)*, 22(4):205–212, August 1988.

[6] Marie-Paule Gascuel. An Implicit Formulation for Precise Contact Modeling Between Flexible SOlids. *Computer Graphics (Proc. SIGGRAPH 93)*, pages 313–320, August 1993.

[7] Andrew Guy and Brian Wyvill Overveld. Hierarchical Subdivision For Implicit Surfaces. Technical report, University of Calgary, Dept. of Computer Science, 1994.

[8] 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.

[9] D. Kalra and A. Barr. Guaranteed Ray Intersections with Implicit Functions. *Computer Graphics (Proc. SIGGRAPH 89)*, 23(3):297–306, July 1989.

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

[11] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.

[12] 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.

[13] Kees van Overveld and Brian Wyvill. Potentials, Polygons and Penguins. An efficient adaptive algorithm for triangulating an equi-potential surface . *Proc. 5th Annual Western Computer Graphics Symposium (SKIGRAPH 93)*, 1992.

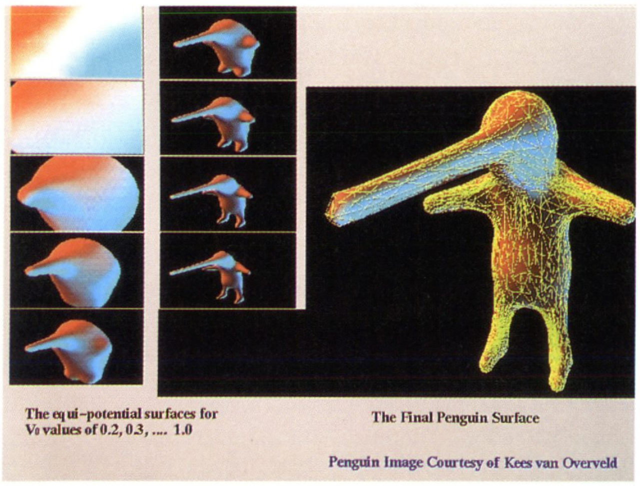
[14] B. von Herzen. *Application of Surface Networks to Sampling Problems in Computer Graphics.* PhD thesis, CalTech, Dept. of Computer Science, 1988.

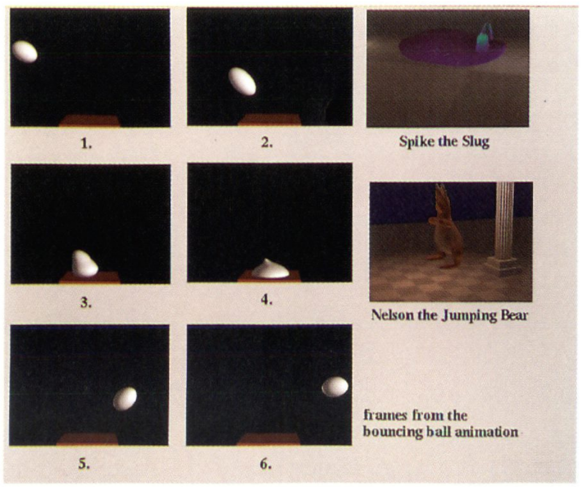
[15] Brian Wyvill. The Great Train Rubbery. *SIGGRAPH 88 Electronic Theatre and Video Review*, Issue 26, 1988.

[16] Brian Wyvill. Warping Implicit Surface for Animation Effects. *Proc. Western Computer Graphics Symposium (SKIGRAPH 92)*, pages 55–63, 1992.

[17] Brian Wyvill, Jules Bloomenthal, Geoff Wyvill, Jim Blinn, John Hart, Chandrajit Bajaj, and Thad Bier. Course Notes. *SIGGRAPH '93, Course #25, Modeling and Animating with Implicit Surfaces*, 1993.

[18] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2(4):227–234, February 1986.


The equi-potential surfaces for
V₀ values of 0.2, 0.3, ... 1.0
The Final Penguin Surface
Penguin Image Courtesy of Kees van Overveld


1.
2.
Spike the Slug
3.
4.
Nelson the Jumping Bear
5.
6.
frames from the
bouncing ball animation