Interactive Techniques for Implicit Modeling

Jules Bloomenthal
Brian Wyvill[1]
Xerox PARC
Palo Alto, California

Abstract: Recent research has demonstrated the usefulness of implicit surfaces for modeling geometric objects. The interactive design of such surfaces has not, however, received the same attention as has the design of parametric surfaces. Principally this is due to the difficulty of controlling the shape of implicit surfaces while displaying the changes quickly enough for use within an interactive design environment. This paper describes progress towards interactive control of implicit surfaces and introduces new techniques useful to the designer.

CR Categories and Subject Descriptors: I.3.5 **[Computer Graphics]:** Computational Geometry and Object Modeling - Geometric algorithms; I.3.6 **[Computer Graphics]:** Methodology and Techniques - Interaction techniques.

Additional Keywords and Phrases: Modeling, Design, Implicit Surface, Animation.

Introduction

In computer graphics, a designer must design and possibly animate three dimensional models. The most popular primitives for designing models are parametric surfaces such as polygon meshes and bicubic patches [Foley 82]. Operations such as *extrusion*, *surfaces of revolution*, and *control point manipulation* aid in the construction of three-dimensional models, but do not, in general, lend themselves to the creation of blended, offset, branching, or algebraic surfaces, or surfaces that obey some constraint based on distance [Bloomenthal 88][2].

An implicit surface is a surface consisting of those points $\boldsymbol{p}$ that satisfy the arbitrary implicit function, f, $f(\boldsymbol{p}) = 0$. Because they possess blending and constraint properties, implicit surfaces have received recent attention in the design of three dimensional objects [Sederberg 85, Rockwood 89]. In many cases, models defined implicitly are difficult to build with conventional, parametric methods. There are aspects of implicit surfaces, however, that complicate their use in the design process. Current tools for interacting with implicit surfaces lack the display speeds and control flexibility necessary for a truly interactive design environment.

In particular, the ability to display a surface in real or near real time is crucial to the design process and, in general, implicit surfaces are more time consuming to display than parametric surfaces. Also, a design system must provide for a high degree of user control of the surface; a number of techniques and tools have been developed that permit global and local control of conventional parametric surfaces, but such methods await development for implicit surfaces.

In this paper we describe techniques to improve the efficiency of current display methods for implicit surfaces and introduce alternative display methods requiring substantially less time. We also investigate methods for the control of implicit surfaces during the design process with the goal of improving the interaction of the designer with the surface being designed.

Previous Work

Both Ricci and Blinn demonstrated that implicit functions are well suited to represent blended surfaces [Ricci 73, Blinn, 82]. Blinn demonstrated the value of this technique for both scientific visualization and for modeling. Subsequently, several researchers in the United States [Sederberg 85], in Japan [Nishimura 85] and in Canada [Wyvill 86a] developed techniques for implicit surface modeling.

Common methods for the display of implicit surfaces include ray tracing, scan conversion, and polygonization.

---

[1] On leave from the University of Calgary, Calgary, Alberta.

[2] With the exception of refinement methods, such as recursive polyhedral subdivision, which can produce smooth, branching shapes [Nasri 87].

Blinn employed a ray tracer optimized for a molecular surface; Hanrahan developed a ray tracer for general algebraic surfaces [Hanrahan 85]. Ray tracing arbitrary implicit surfaces, however, is slow. Recently, Sederberg developed a scan line method for the shaded display of implicit surfaces limited to polynomials [Sederberg 89].

Wyvill introduced a spatial partitioning polygonization algorithm, facilitating the rendering of implicit surfaces with conventional polygon techniques. Bloomenthal demonstrated the advantage of an octree space partitioning and provided a polygonization method that adapts to surface curvature [Bloomenthal 88]. Polygonization of the implicit surface permits the relatively rapid creation of a prototype, which is particularly important in an interactive design environment.

The ray tracing and polygonization methods are capable of displaying arbitrary implicit surfaces, but do not, unfortunately, operate in real or near real time on today's workstations. This is because surface vertices are obtained through an iterative search mechanism; usually this is a convergence along a segment that spans a root of the implicit function (whether this be a portion of an eye to object ray or an edge of a spatial partitioning cell). This requires considerably more processing than for a comparable parametric surface.[3]

Characteristics of Implicit Surface Modeling

Algebraic surfaces constitute the best known subset of implicit surfaces. Their shape, however, is difficult to control interactively; a designer often has no intuitive understanding of the effect of altering polynomial coefficients or exponents.

Our research has focused on the use of implicit surfaces defined by *skeletons* [Wyvill 86b, Bloomenthal 89]. By skeleton we mean a tree structured set of skeletal elements, such as points, curves or polygons. The designer creates a shape by interactively defining the skeleton and various parameters that control how that skeleton becomes a polygonized surface. Although many skeletally defined surfaces may be expressed analytically, we prefer to treat our implicit functions as *procedural*, *i.e.* defined by procedures that return a scalar value given a three-dimensional point. Such procedures may contain conditionals and other operations difficult or impossible to express analytically.

[3] Parametric functions readily generate surface vertices by evaluating the function over its parametric range. Some algebraic surfaces are readily solved analytically, but we do not limit our consideration to this subset of implicit surfaces.

We confine our representation of implicit surfaces to polygons [Wyvill 86a, Bloomenthal 88], since they are drawn quickly and are the most generally used graphic primitive for interactive modeling. A polygonized implicit surface can not contain intersecting polygons, a characteristic distinguishing it from a parametric surface.

Software required to convert a high level description of a model to a lower level polygon description is usually more complex for parametric surfaces; the software must explicitly accommodate the topological complexities of the model. For example, consider connecting two cylinders ends; the parametric definition must explicitly account for the connection; the implicit definition need not.

Shape Control

This section summarizes existing techniques for creating shapes from skeletons and describes new ways in which a user may interactively control the surface shape.

Skeletons

Various skeletally defined shapes are described in the literature [Burtnyk 76]. We define a skeleton to consist of:

- **points**: degenerate skeletons that serve as centers for simple quadrics (spheres, ellipsoids) or superquadrics.
- **splines**: a set of central axes for generalized cylinders with possibly varying radii or cross sections.
- **polygons**: a mesh of polygons and splines used to produce an offset surface.

Each skeletal element is associated with a locally defined implicit function; individual functions are blended using a polynomial weighting function that can be controlled by the user. Figure 1 illustrates in two dimensions the relation between the skeleton and its surface.

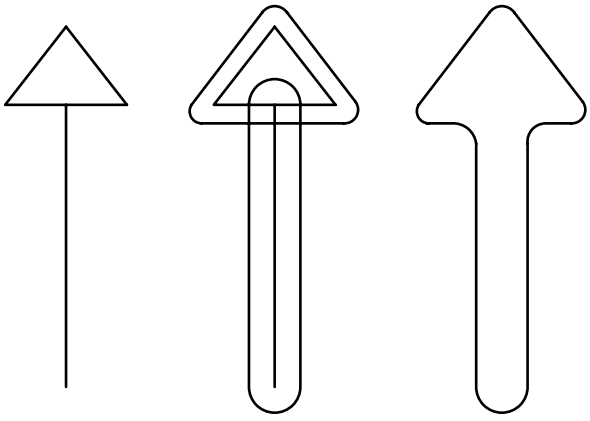

Figure 1: **A skeleton (left) defines two implicit primitives (center), resulting in a blended surface (right).**

Well established techniques exist for manipulating and editing skeletons in real time [Sutherland 63]. In using a skeleton to define an implicit surface, the designer may exercise global or local control of an implicit model in three separate ways:

- definition/manipulation of the skeleton.
- definition/adjustment of those implicit functions defined for each skeletal element.
- definition of a blending function to weight the individual implicit functions.

The skeleton is usually much simpler than its corresponding surface and yet can be quite instructive to the designer. In Figure 2 the branching generalized cylinder (a surface offset from a skeleton consisting of space curves) is shown as a skeleton and as a surface.

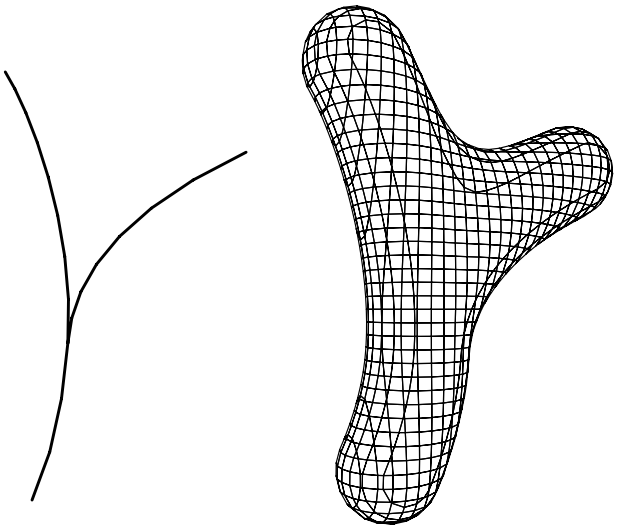

Figure 2: A skeleton and corresponding surface.

Blending of primitives

Each implicit primitive associated with a skeletal element may be blended using established implicit blending techniques [Wyvill 86a, Bloomenthal 88, Rockwood 89]. Woodwark offers a survey of blending techniques [Woodwark 86].

A weighting function that blends individual implicit primitives is shown in Figure 3 [Wyvill 86a]. For a three-dimensional point **p** the distance *r* is found to each skeletal element and weighted by *f*; if *r* >= *R*, its contribution to the blend is zero. By adjusting *f* and *R*, the designer defines the comparative influences and final blend of the primitives. This function is based on radius-squared, eliminating the need for a square root when computing distance. Note that *f* can be negative, providing subtractive as well as additive primitives.

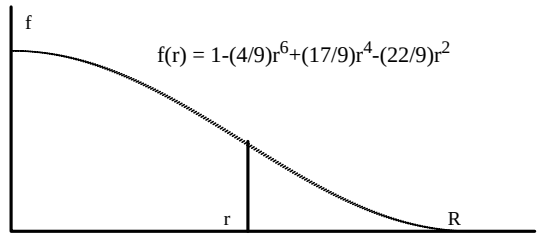

Figure 3: Weighting function for blending implicit primitives.

Offsets to the function

The user may define an offset (sometimes called the *iso-value* or *contour level*) to be added to the blend of implicit primitives, resulting in a dilation or contraction of the entire object; this is demonstrated in Figure 9 where different surfaces are obtained for different offset values.

Manipulation of individual functions

Certain implicit primitives are easily and directly controlled by the designer. For example, the *Graphicsland* animation system at the University of Calgary allows the user to interactively alter ellipsoidal primitives defined as quadrics or superquadrics [Wyvill 86c]. In Figure 4, the axes of the ellipsoids are represented skeletally by prisms.

By manipulating the skeletal ellipsoids, the user can produce complex models. For example, the train's engine shown in Figure 5 requires only twenty primitives. Six ellipsoids form the boiler, which sits atop a base of six smaller ellipsoids. Other small ellipsoids provide detail, such as the chimney. The tops of the cab and the chimney have been flattened using primitives with large radii and negative weights. In Figure 4 negatively weighted ellipsoids are shown as four sided prisms; positive ellipsoids are shown as eight sided. The train was the subject of a short animation [Wyvill 88].

Figure 4: Skeletal elements for the train.

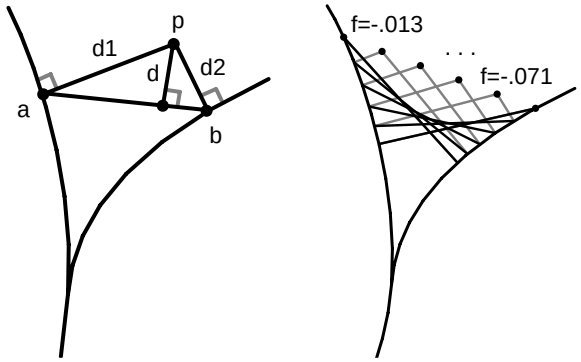

Figure 6: A procedural implicit function and corresponding diagram.

It is an open question as to how such definitions may be described without resorting to a programming language. A promising approach is to define these procedures *by example*. Various two-dimensional systems (*e.g.* MetaMouse, [Maulsby 89]) provide for programming by example, and other graphical systems (like Juno [Nelson 85]) permit definition of graphical operations through example. To our knowledge no such definitions by example exist for three dimensional models.

Figure 5: Surface of the train.

Procedural methods

The blending, offset, and manipulation techniques just described depend on simple metrics (such as distance to a curve) or simple algebra (such as an ellipsoidal function). Many implicit surfaces are not so easily defined by so few parameters. A fourth technique available to the designer is the *procedural* implicit function; this is an arbitrary function whose input is a point in space and whose output is a real value. This permits arbitrarily complex geometric operations to occur during the computation of the function value [Bloomenthal 89]. For example, in Figure 6 (left) the value at the point **p** depends on the distance *d* to the base of the triangle formed by **p**, **a**, and **b**. We call this a *compound metric*, as opposed to the *simple metric* that would sum functions of *d1* and *d2*.

A procedural implicit function permits a greater degree of localized control as compared to a simple blend of implicit primitives in which each primitive potentially has a global affect on the surface. We consider a procedural function to define a class of objects; parameters of the procedure permit the definition of individual objects within the class.

Interaction with complex procedures is somewhat awkward; usually changes to the procedure must be compiled and the procedure module reloaded. Alternatively, the procedure can be interpreted, eliminating compilation at the expense of performance. A technique we suggest to assist the designer with procedural definitions is the *interactive diagramming* of the operation of the procedure. By allowing the designer to interactively query and diagram the function at arbitrary points in space, it becomes easier to understand and modify the procedural function. Figure 6 (right) is an example of such diagramming.

Shape Display

With implicit modeling, a designer can control complex shapes using a few high level primitives. For this to be interactive, alterations in the primitives must be reflected in the corresponding surface as quickly as possible. Rapid feedback is necessary to convince the user that a surface is changing shape smoothly as it is edited. In reality, immediate feedback is difficult to achieve for complex surfaces using current workstation technology. In this section we describe existing display techniques and suggest improvements that make progress towards real time response.

Established Display Techniques

In [Blinn 1982], surfaces representing electron density functions were rendered directly, a technique that does not lend itself to interactive work on current hardware. In an attempt to accelerate the display of implicit surfaces, a polygonization algorithm using spatial partitioning was developed [Wyvill 86a]. With spatial partitioning, surface accuracy can be exchanged for speed by altering the size of the partitions (*cells*), essentially changing the sampling rate of the implicit function. This allows for fast prototyping.

As previously mentioned, the skeleton of a model is composed of various elements. To find the value of the implicit function at a point in space, the contributions from all nearby skeletal elements are summed. To locate points on the surface, the function evaluation is repeated for many points in space. Efficiency can be improved by limiting the radius of influence of each skeletal element and sorting

them into the cells they influence. Thus, to evaluate the function at a point within a given cell, only those primitives influencing the cell need be considered.

The uniform space partitioning introduced in [Wyvill 86a] was improved by using an octree [Bloomenthal 88]; an adaptive algorithm was also introduced that samples the function more often where the curvature is greater. For a given maximum depth of the octree, fewer polygons are produced for the same quality of display, thus reducing display time.

In [Jevans 87] the use of frame coherence is suggested to improve feedback when local portions of the implicit surface are modified. Only those cells of the octree for which the influencing skeletal elements are modified need be re-computed and polygonized. This achieves considerable savings when changes are local, since much of the model does not require re-computation.

Improved Surface Display Techniques

A number of methods exist that represent the surface at varying levels of detail; each method, to some degree, exchanges polygonization accuracy for speed of display. We now describe several techniques to reduce display time. We propose that a design system permit the user to select the display technique most appropriate for the required surface detail and display speed.

· **Octree display**

Rather than display surface polygons, the octree is displayed with back faces of cells removed, something appropriate for volume rendering hardware. This is a coarse representation of the surface, but does not require polygonization.

· **Polygon vertex computation**

Once the implicit function is spatially partitioned, polygon vertices are computed from those partition edges that penetrate the surface. Again, accuracy may be exchanged for speed. Binary subdivision or iterated *regula falsi* along the partition edge is highly accurate [Bloomenthal 88], but requires repeated evaluation of the implicit function. Alternatively, a single step of *regula falsi*, *i.e.*, simple linear interpolation (Figure 7), may be employed, offering considerable savings in computation [Wyvill 86a]. This presumes the function to be continuous and locally linear. Linear interpolation of nonlinear functions will be inaccurate, resulting in artifacts such as rippling surfaces in motion. All of these methods are mathematically simpler if the edge is aligned with a major axis.

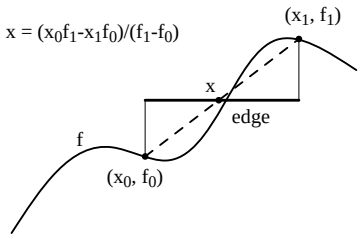

Figure 7: Linear interpolation along an edge.

· **Improved partitioning strategies**

As previously mentioned, adaptive subdivision produces a surface of higher complexity for the same number of polygons as produced by uniform subdivision. Adaptive subdivision requires computation of curvature, however, which is often compute intensive. Thus, the workstation's ability to display polygons must be balanced against its ability to create the octree. For those surfaces where curvature is computationally inexpensive, adaptive subdivision will improve design interaction.

A reduction in tree traversal time can improve design interaction. In a variant octree data structure, the number of child nodes, rather than a constant eight, is a function of surface shape, resulting in fewer empty nodes. Memory savings of 15% and traversal time savings of 10% were achieved [Jevans 88b]. Other partitionings, such as the KD tree. may converge to the surface more quickly than does the octree [Samet 84], again resulting in improved design interaction. These marginal performance improvements may not merit the additional implementation complexity.

· **Sub-division ''paint brush''**

Rather than subdivide according to computed curvature, an octree could be subdivided interactively. Here, the user indicates those portions of the octree that should be subdivided to provide greater surface detail. The intersection of the ''user ray'' (a line from the eye through a point on the screen) with an octree is a relatively fast operation [Glassner 84], capable of interactive speeds.

· **Function simplicity**

In designing an implicit surface, most processing is devoted to evaluating the implicit function. It is critical this be done efficiently. Significant improvement often can be achieved with an approximation to the function. For example, the generalized cylinder in [Bloomenthal 88] requires the solution of a fifth degree polynomial; in [Wyvill 86a] a set of spheres approximates a cylinder, requiring a simpler solution (Figure 8). These *soft objects* (*a.k.a. metaballs*; see [Nishimura 85]) provide a large performance improvement. The approximation may be replaced with the exact solution once the prototype is complete.

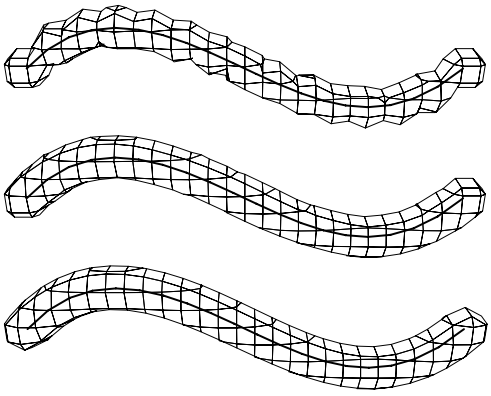

Figure 8: Approximation with 15 spheres (top), 30 spheres (center), the exact solution (bottom).

· 2d slices

Rather than display a surface, it is possible to display the implicit function directly, at a considerable savings of time. A two-dimensional window into the function can be rendered on a display, as shown in Figure 9. This requires interpretation skills on the part of the designer, and does not necessarily provide a good impression of the object's shape. However, certain qualities, such as surface continuity, may be readily detected by direct inspection.

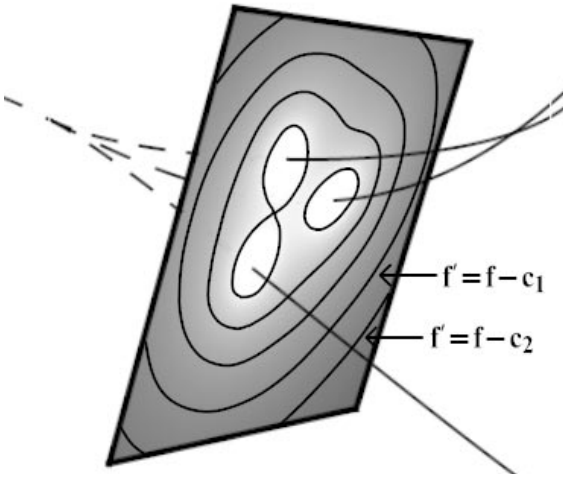

Figure 9: Slice of surface and varying contours.

The object shown in Figure 2 required an estimated 44,000 evaluations of the implicit function; the slice shown in Figure 10 required about 14,000 evaluations, for a performance improvement of approximately three to one. This improvement depends on the shape of the object and the resolutions used (the size of the partitioning cubes and the resolution of the slice); a coarser sampling for the slice may be acceptable to the designer.

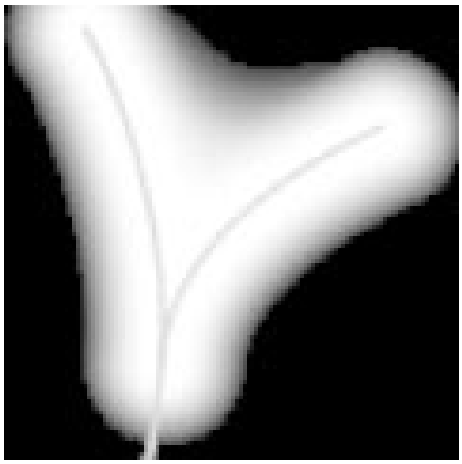

Figure 10: Slice for object shown in Figure 2.

· Image thresholding

If an implicit function is rendered as a two-dimensional slice, or as a volume [Drebin 88], manipulation of the display's video look-up table (*i.e.*, color map) may yield a highly interactive study of the object. For example, it is possible to approximate the effects of altering offsets to implicit functions by modifying the tables. This technique can also accentuate discontinuities in the implicit function.

· Scattering

Another technique that dramatically improves design performance is *scattering*, in which seed points, presumed close to the surface, migrate to the surface. This does not require polygonization, which is relatively compute intensive. The seed points are readily established given a skeleton and its corresponding cross-sections.[4] For each point the gradient of the implicit function is computed, providing a direction for the points to migrate. The points continue to migrate until reaching the surface, Figure 11. No polygonal data is available with this technique and fine detail will be absent, but the technique offers greatly improved speed. For example, 1,300 function evaluations were required for the object in Figure 11; the same object shown in Figure 2 required an estimated 44,000 evaluations, yielding an approximate performance improvement of 33 to 1.

Figure 11 lacks apparent depth since a major depth cue is given by parallax, which is readily available with an interactive display. The use of points to represent a surface is not new [Connolly 83], but the migration of points is a novel technique for the visualization of implicit models. An additional benefit is that point density is higher along silhouettes, aiding the designer in interpreting the display.

[4] This requires a vector orthogonal skeleton's tangent at a cross-section, which is simpler than computing a reference frame using parallel transport [Shani 84].

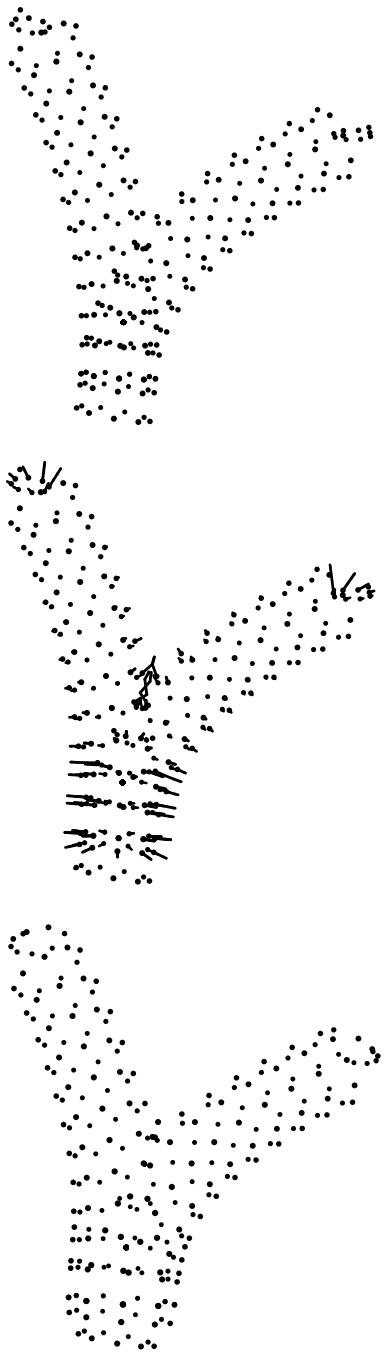

Figure 11: Original seed points (top), migration paths (center), and resting points (bottom).

· **Idle moments**

The user should be allowed to enable the system to refine the surface during otherwise idle moments, similar to the adaptive refinement of images [Forrest 85, Bergman 86]. These refinements progress naturally from:

- a simple skeleton,
- cubes representing the terminal nodes of an octree,
- the polygonized octree,
- a rendered surface.

Conclusions and Future Work

We have considered performance issues in the interactive design of implicit surfaces, surveying prior implicit surface research and noting its relevance to interactive design. We have proposed new design methods that potentially operate at interactive speeds. These include:

- the use of alternative data structures.
- the display of a slice of the implicit function.
- the scattering/converging of points to approximate the implicit surface.
- the interactive diagramming of procedural implicit surfaces.
- the user control of adaptive subdivision.

We have discussed ways in which the user can alter the shape of an implicit surface, such as offsets to the implicit function, adjusting the blending functions, and interactively defining skeletons. It remains an open question, however, whether users of the future will be able to construct sophisticated, procedural shapes without the need to program. Here the prospects of indirect programming by example seem promising, although distant.

Our present work is such that the novel techniques described have been implemented and tested in isolation. We hope to build an interactive test bed where tools for implicit surface design may be evaluated in a design environment. We also anticipate development of a set of design criteria to assist us in comparing implicit and parametric methods in model design.

Acknowledgements

We thank a number of individuals and organizations for their assistance. Pat Hanrahan, Paul Heckbert, Dave Jevans, and Geoff Wyvill all offered valuable technical insight during the past several years, The Natural Sciences and Engineering Research Council of Canada and the Xerox Palo Alto Research Center provided funding for this research. Finally, we thank Rich Riesenfeld who succinctly noted, more than a decade ago, that surface design was an interesting problem.

References

Bergman L., Fuchs H., Grant E., Spach S. Image Rendering by Adaptive Refinement. Proceedings of SIGGRAPH'86 (Dallas, Texas). In *Computer Graphics 20*, 4 (August 1986), 29-38.

Blinn, J.F. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics, 1*, 3 (July 1982), 235-256.

Bloomenthal, J. Polygonization of Implicit Surfaces. *Computer Aided Geometric Design, 5*, 4 (November 1988).

Bloomenthal, J. Techniques for Implicit Modeling. *Xerox PARC Technical Report*, P89-00106 (January 1989).

Burtnyk N. and Wein M. Interactive Skeleton Techniques for Enhancing Motion Dynamics in key Frame Animation. *Communications of the ACM, 19*, 10 (October 1976), 564-584.

Connolly, M.L. Solvent-accessible Surfaces of Proteins and Nucleic Acids. *Science, 221* (August 1983), 709-713.

Drebin, R.A., Carpenter, L., and Hanrahan, P. Volume Rendering. Proceedings of SIGGRAPH'88 (Atlanta, Georgia). In *Computer Graphics 22*, 4 (August 1988), 65-74.

Foley, J.D. and Van Dam, A. *Fundamentals of Interactive Computer Graphics*. Addison-Wesley, Reading, Mass, 1982.

Forrest R. Antialiasing in Practice in Fundamental Algorithms for Computer Graphics, Ed. Earnshaw, R.A. in Proceedings of NATO ASI Series, Springer-Verlag, 1985, 113-134.

Glassner, A.S. Space Subdivision for Fast Ray Tracing. *IEEE Computer Graphics and Applications, 4*, 10 (October 1984), 15-22.

Hanrahan P. Ray Tracing Algebraic Surfaces. Proceedings of SIGGRAPH'83 (Detroit, Michigan). In *Computer Graphics 17*, 3 (July 1983), 83-90.

Jevans, D.J., Wyvill, B., and Wyvill, G. Speeding up 3D animation for simulation. *Proc. MAPCON IV (Multi and Array Processors)*, (January 1988a), 94-100.

Jevans, D.J., Wyvill, B. Ray Tracing Implicit Surfaces *Proc. Research Report, University of Calgary, Dept. of Computer Science*, 88/292/04, (January 1988b).

Maulsby, D.L., Witten, I. H., Kittlitz, K.A. Specifying Graphical Procedures by Example Proceedings of SIGGRAPH'89 (Boston, Mass.). In *Computer Graphics 23*, 3 (July 1989), 127-136.

Nasri, A.H. Polyhedral Subdivision Methods for Free-Form Surfaces. *ACM Transactions on Graphics, 6*, 1 (January 1987), 29-73.

Nelson, G. Juno, a constraint-based graphics system. Proceedings of SIGGRAPH'85 (San Francisco, California, July 22-26, 1985). In *Computer Graphics 19*, 3 (July 1985), 235-243.

Nishimura. H., Hirai, A., Kawai, T., Kawata, T., Shirakawa, I., and Omura, K. Object Modeling by distribution function and a method of image generation. *Journal of papers given at the Electronics Communications Conference 1985, J68-D(4)*, 1985 (In Japanese).

Ricci, A. A Constructive Geometry for Computer Graphics. *The Computer Journal 16*, 2 (May 1973), 157-160.

Rockwood, A. The Displacement Method for Implicit Blending Surfaces in Solid Models, *ACM Transactions on Graphics, In Press* 1989.

Samet, H. The Quadtree and Related Hierarchical Data Structures, *ACM Computing Surveys*, 16(2), (June 1984).

Sederberg, T.W. Algebraic Piecewise Algebraic Surface Patches. *Computer Aided Geometric Design, 2* (1985), 53-59.

Sederberg, T.W. and Zundel, A.K. Scan Line Display of Algebraic Surfaces. Proceedings of SIGGRAPH'89 (Boston, Mass.). In *Computer Graphics 23*, 3 (July 1989), 147-156.

Shani, U. and Ballard, D.H. Splines as Embeddings for Generalized Cylinders. *Computer Vision, Graphics, and Image Processing, 27* (1984), 129-156.

Sutherland, I.E. SKETCHPAD: A Man-Machine Graphical Communication System. *Proc. AFIPS Spring Joint Computer Conference, 23* (1863), 229-246.

Woodwark, J.R. Blends in Geometric Modeling. *Proceedings of the 2nd IMA Conference on the Mathematics of Surfaces*, (Cardiff, September 8-10, 1986).

Wyvill, G., McPheeters, C., and Wyvill, B. Data Structure for Soft Objects. *Visual Computer, 2*, 4 (August 1986a), 227-234.

Wyvill, B., McPheeters, C., and Wyvill, G. Animating Soft Objects. *Visual Computer, 2*, 4 (August 1986b), 235-242.

Wyvill, B., McPheeters, C., and Garbutt, R. The University of Calgary 3D Computer Animation System. Journal of the Society of Motion Picture and Television Engineers, 95, 6 (1986c), 629-636.

Wyvill, B. The Great Train Rubbery. SIGGRAPH '88 Electronic Theater and Video Review, Issue 26 (August 1988).