TEXTURE MAPPING THE BLOBTREE

Mark Tigges *Brian Wyvill*

Dept. of Computing Science
University of Calgary, Calgary, Alberta, Canada
mtigges@cpsc.ucalgary.ca blob@cpsc.ucalgary.ca

ABSTRACT

Recently an automatic solution to the problem of applying 2D textures to an implicit surface has been introduced [13]. The method derives a uv-texture coordinate system by tracing particles from the implicit surface to a *support surface*. In this paper we describe the adaptation of this algorithm to the *BlobTree* data structure. The *BlobTree*provides an implicit modelling system through a hierarchy of blends, CSG and warping applied to primitives. There are several problems involved in applying textures in such a general modeling system, particularly across CSG junctions. In this paper we identify these problems and introduce some possible solutions.

1. INTRODUCTION

Implicit surfaces, those defined by an iso-contour of an implicit function [3], have enjoyed an active research community for over a decade. The use of this class of surfaces outside of academia has been restricted to only special case examples, partly due to the lack of both good interactive modeling systems and the inability to apply 2D textures.

The use of implicit surfaces in creation of cartoon like characters [8] has been outlined and extension of implicit surface modeling to include CSG operations have made description of solid models for engineering possible [11]. These steps are progress towards fulfilling the needs for commercial use. One open problem which has only partially been resolved, is the derivation of a 2D texture coordinate system directly from the definition of the skeletal implicit model.

Parametric modeling techniques have the advantage over implicit surface techniques, that a natural 2D texture coordinate system can be easily derived from parametric space. Implicit surfaces do not yield a two dimensional parameterization easily. A computed parameterization would need to be suitable for use in application of texture maps, and for use in displacement mapping [2]. Solid texturing techniques have been applied to implicit surfaces [12], but these methods do not allow an animator to apply a bi-dimensional texture onto an implicit surface. For example, the application of a textured face to an animated figure. Ability to generate two dimensional mappings for the surface of an implicit model is required for iso-surface techniques to enter into mainstream modeling and animation use in computer graphics.

1.1. The BlobTree

The *BlobTree* [9] provides a hierarchical data structure for the definition of complex models built from implicit surfaces, CSG Boolean operations and field warping functions. Models built with this system are generally considered as the hierarchical composition of multiple objects rather than a complete object (see figure 1). Typically these components are defined with differing surface attributes: specularity, reflectance, transmittance etc; accordingly we wish to include 2D surface parameterizations for application of texture maps. In this work we present extensions to the *BlobTree* that allow the definition of such parameterizations and their use for texture mapping.

1.2. 2D Texturing of Implicit Surfaces

An algorithm to map a point on an iso-surface to a point in a defined texture space was introduced by Zonenschein et al. [13]. The method bounds the iso-surface to another surface (known as the *support surface*) which

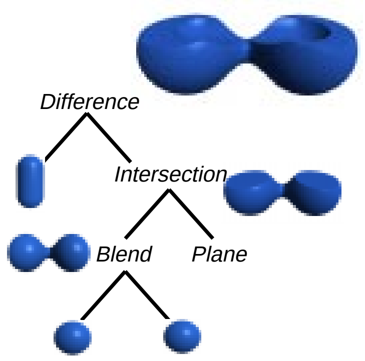


Figure 1: Example of a model built up from a hierarchy of primitives and operations.

is easily parameterizable in two dimensions yielding a 2D texture coordinate system. A system of particles are traced from the iso-surface (x, y, z) to the parameterized surface, the particle intersection with the parameterized surfaces gives the (u, v) coordinates which become the texture coordinates of (x, y, z). The support surface is any surface on which exists a known texture mapping. Zonenschein et al. used a cylindrical support surface to contain the texture space. Spherical and cylindrical bounding surfaces have been tested but in principal any other appropriate parameterizable surface can also be used. Smets-Solanes [6] also uses particle systems but in his work it is used to transform a mapping to minimize inconsistencies due to changes to the surface over time.

Incorporating the particle texturing technique into the *BlobTree* structure introduces several problems and considerations. Most notably the introduction of CSG can cause a discontinuity in the mapping. Also it is important to consider desired effects in the combination of mappings and textures at the implicit surface blends. In section 2 we discuss our computation for the trajectory of the particles, section 3 discusses the integration of this computation into the *BlobTree* and conclusions are drawn in section 4.

2. TEXTURING ALGORITHM

Texturing implicit surfaces using particle systems depends on computing the trajectory of each particle. The origin of a particle $\mathbf{M}_0$ is provided by sampling methods for implicit surfaces [7], or one can use the vertices from a standard polygonization, [3]. To adequately deal with problems introduced by the *BlobTree* data structure, the algorithm presented here

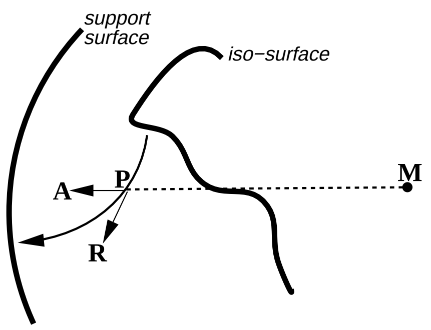


Figure 2: Top view of particle trajectory from an iso-surface to a cylindrical bounding surface showing the components of particle direction.

differs somewhat from the Zonenschein et al. algorithm (see [13]). More control over the trajectory is required than that afforded by the differential equation used in that work in order to adequately deal with CSG junctions and other *BlobTree* features. As will be seen in section 3 the trajectory depends on the type of nodes in the tree for which we are computing a mapping. This section will outline the basic calculation.

We need to compute the direction of a particle for any time t during its trajectory. The particle direction, $\mathbf{T}$, is computed as the linear combination of two normalized vectors: the repulsion force $\mathbf{R}$ that controls the direction of the particle away from the iso-surface and, the attraction force towards the support surface, $\mathbf{A}$. Figure 2 shows the vector forces on a particle during its trajectory. The attraction vector $\mathbf{A}$, equation 1 is computed as the normalized vector lieing in the shortest path to the support surface ($\mathbf{M} - \mathbf{C}$). The gradient of the field function for the surface, $\nabla F(\mathbf{M})$, is used for the repulsion vector $\mathbf{R}$ as shown in equation 2.

$$\mathbf{A}_t = \frac{\mathbf{M}_t - \mathbf{C}}{\|\mathbf{M}_t - \mathbf{C}\|} \quad (1)$$

$$\mathbf{R}_t = \nabla F(\mathbf{M}_t) \quad (2)$$

It should be noted that in the case that the parameterized bounding surface is a cylinder, $\mathbf{C}$ is the point on the axis of the cylinder for which $\mathbf{A}$ is orthogonal to the cylinders axis. This means that $\mathbf{C}$ will change during the particles trajectory. This is not the case if the parameterized bounding surface is a sphere, where $\mathbf{C}$ is the center of the sphere.

The formulation for the direction $\mathbf{T}$ of a particle at a time t is given in equation 5. The weights for the linear combination are shown as K_0 and K_1, in equations 3 and 4, and are dependent on the value of the field function

$F(\mathbf{M})$ for the particle position. K_0 is the weight for the repulsion vector, we use $F(\mathbf{M})$. The weight of the attraction vector K_1 is then computed as $1 - K_0$ to ensure that $\mathbf{T}$ is also a normalized vector.

$$\begin{aligned} K_0 &= \text{clamp}(\|F(\mathbf{M}_t)\|, 0, 1) & (3)\\ K_1 &= 1 - K_0 & (4)\\ \mathbf{T}_t &= K_0 \cdot \mathbf{R}_t + K_1 \cdot \mathbf{A}_t & (5)\end{aligned}$$

The calculation of the particle position over the trajectory is given by equation 6.

$$\mathbf{M}_t = \mathbf{M}_{t-dt} + \mathbf{T}_{t-dt} \qquad (6)$$

The calculation of the complete path of the trajectory of $\mathbf{M}_t$ is computed by stepwise solution of $\mathbf{M}_t$, equation 6.

The field function is assumed to have a range between 1 and 0, where the field value $F(\mathbf{M}) = 1$ for a point on the skeleton and decreases to $F(\mathbf{M}) = 0$ with increase in distance from the skeleton. The trajectory computation terminate when the particle leaves the bounding volume of the field. The reason for this is the first derivative of the trajectory does not change when the field is constant. Therefore the texture coordinates for the particle are already determined when the particle leaves the field. This point is important as it implies that the support surface geometry can be inferred by the bounding volume of the scalar field. This guarantees that there are no discontinuities in the computed 2D mapping due to the ends of a support cylinder. There is an assumption underlying this simplification: that the bounding volume of the implicit surface, around which the support surface is defined, is in fact a bounding volume of the non-zero portion of the scalar field used to define the surface (see figure 3). This places a restriction on the type of scalar

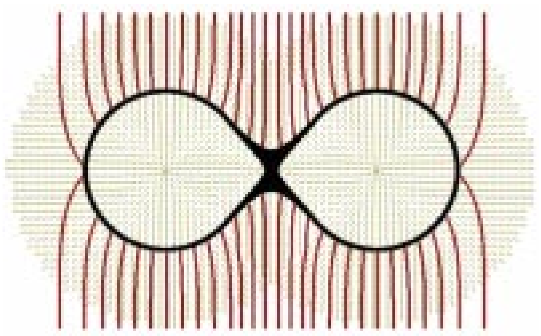


Figure 3: 2D illustration of the trajectories moving towards a support surface that bounds the non-zero area of the scalar field.

field used, namely that it must vanish at some distance from the skeleton. This is not a severe restriction as most systems use one of the fields defined in [5, 10] which are polynomials which vanish at a determined constant (see [4] for an in depth discussion of field functions for procedurally defined surfaces).

The method as described, computes a mapping which contracts at concave areas of a surface and expands at convex areas (see figure 4). This effect mimics our interpretation of the blend in a pliable surface, the blend can be viewed as the stretch between two solids as they break apart. However this contraction-expansion effect may not be the generally desired result. We can partially

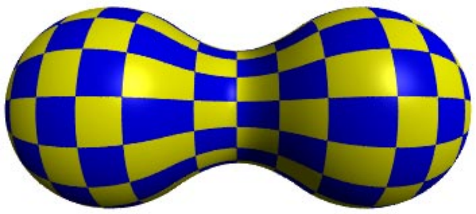


Figure 4: Texture mapping applied to a simple model.

control the effect by scaling the value of K_0 so that instead of ranging from L (the iso-value for the level surface) to 0 during the particles trajectory it ranges from 1 to 0. This range amplifies the effect by placing more emphasis on the gradient in the calculation of T, the particle direction. To reduce the contraction and expansion of the mapping, scale the range of K_0 to reduce the emphasis on the gradient. This simple trick to

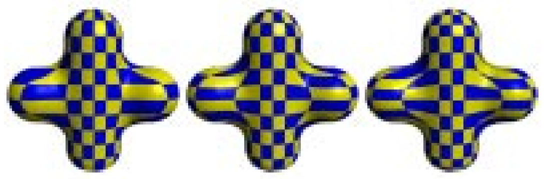


Figure 5: Demonstration of the affect of the influence of the gradient, ranges of K_0 from left to right: $[0.0 \ldots 0.0]$ $[0.5 \ldots 0.0]$ $[1.0 \ldots 0.0]$.

control the contraction and expansion of the mapping is not altogether successfull (see figure 5). The stretch of the texture laterally in figure 5 is controlled however we would like to be able to compute a mapping where the checkers are more closely uniform in size over the entire surface. Clearly more research is needed in this area.

3. A BLOBTREE MAPPING NODE

The texturing algorithm described in the previous section applies texture globally onto the surface. In the case of the *BlobTree* the total surface is the result of the composition of the child nodes of the hierarchy. In order to incorporate the texturing algorithm into the *BlobTree* a new node type is defined. This node is identical to the standard *blending group*, figure 3, with the exception that it provides a 2D parameterization for the surface defined by its children. This node must take into account the sibling relationship of its child nodes in the hierarchy. This means that the texturing algorithm must be modified to provide appropriate mapping across the blends of the nodes for which the mapping is being computed. There are three cases to consider: blending, warping and CSG. We include an overview of the *BlobTree* traversal algorithm for completeness, refer to [9] for a full discussion.

Function $F(\mathcal{N}, \mathbf{M})$:

1. Primitive: $F(\mathbf{M})$.
2. Warp: $F(\mathcal{L}(\mathcal{N}), w(\mathbf{M}))$.
3. Blend: $F(\mathcal{L}(\mathcal{N}), \mathbf{M}) + F(\mathcal{R}(\mathcal{N}), \mathbf{M}))$
4. Union: max$(F(\mathcal{L}(\mathcal{N}), \mathbf{M}), F(\mathcal{R}(\mathcal{N}), \mathbf{M}))$
5. Intersection: min$(F(\mathcal{L}(\mathcal{N}), \mathbf{M}), F(\mathcal{R}(\mathcal{N}), \mathbf{M}))$
6. Difference: min$(F(\mathcal{L}(\mathcal{N}), \mathbf{M}), -F(\mathcal{R}(\mathcal{N}), \mathbf{M}))$

Figure 6: Traversal algorithm for the *BlobTree* data structure.

Where L, R return the left and right nodes of N respectively. Note that in our actual implementation nodes are *n-ary* rather than binary.

3.1. Blending

The surface attributes for a point on the surface which results from a blend node, are derived from the surface attributes of the child nodes. Typically the contribution from a single child node N_i to the surface at some point $\mathbf{M}$, is weighted by the field value for the child at that point $F(N_i, \mathbf{M})$.

We can use the same method for blending of texture values. We trace a particle from the point with respect to each primitive to a support surface which bounds that primitive. This will produce a texture space coordinate and colour for that primitive which can be blended with the sibling values. As pointed out in [14] this can cause possibly undesirable artifacts, where textures are layered on top of each other (see figure 7 left). Instead it was suggested in their paper to blend the support surfaces and trace one particle in the blended area to that *average* support surface. The two methods are shown in figure 7,

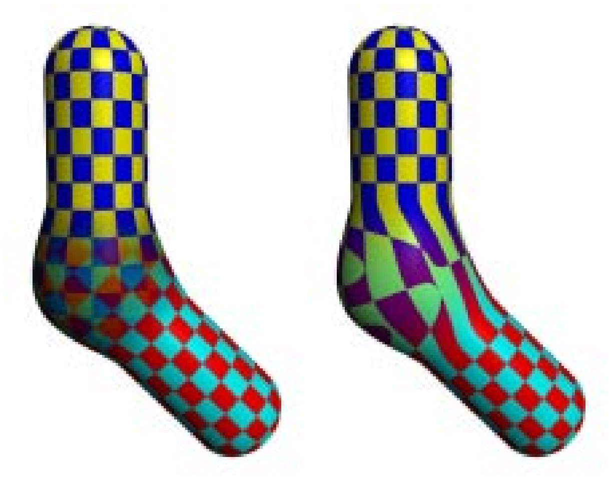


Figure 7: Comparison of texture combination over implicit blends; left: standard weighting, right: support surface blending.

and discussed fully in [14], that work should be consulted for complete detail. In the case where the support surface is blended, the texturing in the blended area becomes very distorted. It would be possible to use the application of a sigmoid function to the weighting value to reduce the area of texture distortion.

3.2. Warping

Warping of the field function in the *BlobTree* is accomplished through the use of the Barr warps [1, 9] (although other warp functions have also been used). These functions $w : \mathbf{R} \to \mathbf{R}$ provide a mapping which can cause twist, taper and bend of the surface. Figure 8 illustrates how the computed 2D mapping is affected by the spatial warp applied to the field. The warps we currently use do not cause a discontinuity in the gradient of the field defining the surface so there is no discontinuity in the 2D mapping. However the gradient is subject to the spatial warp so that particle trajectories following this field cause the 2D mapping to follow the surface. The effect is that the mapping appears as if

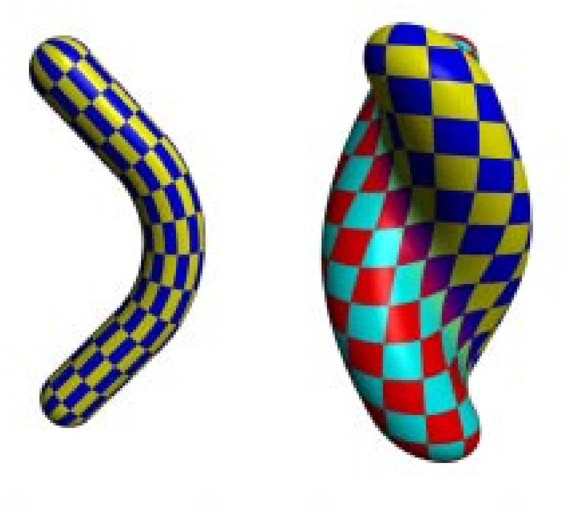


Figure 8: Response of a texture map to warped space.

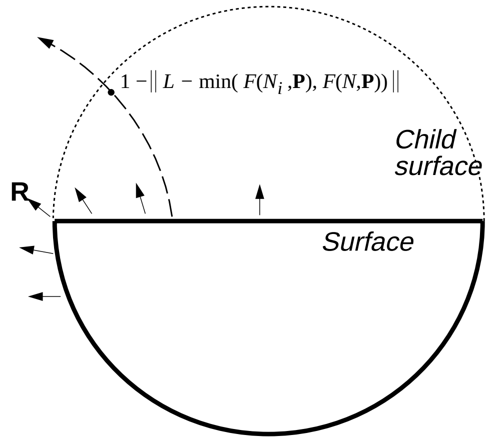
it was applied to the surface before warp was applied. This is partly a desireable effect but it may be that the preference is for the texture to not react to the warp. This is an as yet unexplored area. The most logical solution for future work would be to apply the inverse of the warp function after the trajectory calculation is complete and use the unwarped point to compute the mapping.

3.3. *CSG Boolean Operations*

Texture mapping across CSG junctions with the algorithm described in section 2 presents a special problem. The algorithm is heavily reliant on the gradient of the scalar field defining the surface. This causes the algorithm to fail when the gradient is not defined or contains a discontinuity. Boolean operations can introduce discontinuities in the gradient. The net result is that closely spaced particles on the surface are traced in different directions and hence yield texture coordinates which cause a discontinuity in the mapping. The problem is illustrated in figure 9, at the junction in the bottom image $\mathbf{R}$ is not defined and particles could go in one of two orthogonal directions. One solution is to avoid the problem by not assigning the gradient of the scalar field to the repulsion force $\mathbf{R}$. Instead we assign to $\mathbf{R}$ a normalized direction vector which is dependent on the type of the node, $\mathcal{N}$, as

$$\mathbf{R} = \begin{cases} \mathbf{\Psi}(\mathcal{N}_i) & \text{if } \mathcal{N} \text{ is blend} \\ \sum \mathbf{R}_i \cdot \hat{F}(\mathcal{N}_i, \mathbf{M}_t) & \text{if } \mathcal{N} \text{ is csg} \\ \nabla F(\mathcal{N}, \mathbf{M}_t) & \text{if } \mathcal{N} \text{ o/w} \end{cases}$$

$$\hat{F}(\mathcal{N}_i, \mathbf{M}_t) = 1 - \|L - \min(F(\mathcal{N}_i, \mathbf{M}_t), F(\mathcal{N}, \mathbf{M}_t))\|$$



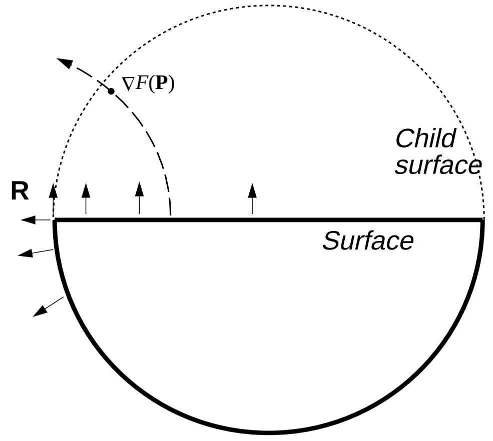


Figure 9: Computation of $\mathbf{R}$ with CSG junctions; top: $\mathbf{R} = \nabla F(\mathcal{N}, \mathbf{M})$, bottom: $\hat{F}$ used to weight the child $\mathbf{R}_i$.

Where $\mathbf{\Psi}$ returns the weighted average of the direction vectors of the child nodes, $\mathcal{N}_i$, in the same manner as used for computing normals of blended nodes [9]. $\mathcal{N}$ is the current node, $\mathbf{R}_i$ is the direction vector for the i^{th} primitive, $F(\mathcal{N}, \mathbf{M})$ is the field value for node $\mathcal{N}$ at point $\mathbf{M}$ and L is the iso-value for the level surface being used.

When a node is a Boolean operation, $\mathbf{R}$ is computed as the normalization of the weighted sum of the child $\mathbf{R}_i$ vectors with $\hat{F}$ used for the weight. This function is meant to produce a weighted average of the child $\mathbf{R}_i$ vectors, however we must ensure that we do not contribute more from a child than is being contributed by the actual CSG surface. Consider figure 9, these diagrams show a particle leaving the flat surface produced by the intersection of a plane and a sphere. In the top figure $\hat{F}$ is used to weight the contribution of the $\mathbf{R}_i$ of the child nodes. The definition of this function ensures that the child surfaces do not contribute

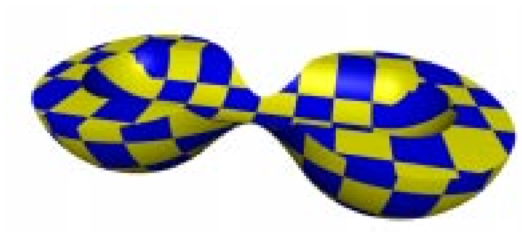


Figure 10: Texture space discontinuity across a CSG intersection.

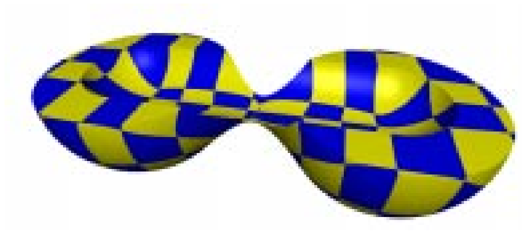


Figure 11: Continuous texture mapping over a CSG intersection.

more than the value of the field function for the entire surface at the current point. In the bottom figure, $\mathbf{R}$ is computed directly as the gradient for the whole surface $\nabla F(\mathcal{N}, \mathbf{M})$, as shown in the figure the discontinuity causes very different trajectories. $\hat{F}$ ensures that the change in the initial direction of particles leaving the surface is continuous. Figure 11 illustrates the use of the new $\mathbf{R}$ in computing particle trajectories for the mapping. In figure 10, the misalignment in the checks across the gradient discontinuity (CSG junction) shows the discontinuity in the 2D mapping. In comparison of the two figures it is easy to notice the improvement in the alignment of the checkers across the CSG junctions. It should be noted that the same function for the computation of $\mathbf{R}$ is used for all Boolean operations: *union*, *difference* and *intersection*.

4. CONCLUSIONS

We have introduced some new techniques for 2D texture mapping of implicit surfaces. Our methods extend the recently introduced algorithm of Zonenschein et al. [13] to be of practical use in an implicit surface system which includes CSG operations and warping. Our method introduces new control mechanisms over the trajectory of particles which are traced from the iso-surface to a support surface.

We have implemented both a polygonizer and a ray tracer for the *BlobTree*. Although we have not yet done any exhaustive testing, we have ray traced a number of models with and without texturing. The particle tracing adds approximately %40 to the time it takes to produce a ray traced image of the untextured primitives.

One of the main advantages of using skeletal implicit models is that blending between parts of the model is automatic. The blended areas can be thought of as a distortion of the original primitives. Our texture technique also produces distorted textures in these areas which may or may not be considered desirable. Future work will address the problem of computing mappings which allow the user better control over these distortions. We will also examine the problem of designing interactive tools for positioning the support surface to produce desired texturing in an interactive implicit modeling system.

5. REFERENCES

[1] Alan H. Barr. Global and local deformations of solid primitives. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 21–30, July 1984.

[2] James F. Blinn. Texture and reflection in computer generated images. *CACM*, 19(10):542–547, October 1976.

[3] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.

[4] Zoran Kacic-Alesic and Brian Wyvill. Controlled Blending of Procedural Implicit Surfaces. *Proc. Graphics Interface 1991*, pages 236–245, 1991.

[5] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Trans. IECE Japan, Part D*, J68-D(4):718–725, 1985.

[6] Jean-Paul Smets-Solanes. Vector field based texture mapping of animated implicit objects. In *Eurographics '96 Conference Proceedings*. Eurographics, 1996.

[7] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 269–278. ACM SIGGRAPH, ACM Press, July 1994.

[8] Brian Wyvill. Animation and Special Effects. *Introduction to Implicit Surfaces*, pages 101–104, 1997. Edited by Jules Bloomenthal With Chandrajit Bajaj, Jim Blinn, Marie-Paule Cani-Gascuel, Alyn Rockwood, Brian Wyvill, and Geoff Wyvill.

[9] Brian Wyvill, Eric Galin, and Andrew Guy. The blob tree, warping, blending and boolean operations in an implicit surface modeling system. Technical Report 98/618/09, The University of Calgary, Dept. of Computer Science, 1997.

[10] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[11] Brian Wyvill and Kees van Overveld. Warping as a modelling tool for csg/implicit models. In *Shape Modelling Conference, University of Aizu, Japan*, pages 205–214. IEEE Society Press ISBN 0-8186-7867-4, March 1997. inivited.

[12] G. Wyvill, B. Wyvill, and C. McPheeters. Solid texturing of soft objects. *IEEE Computer Graphics and Applications*, 7(12):20–26, December 1987.

[13] R. Zonenschein, J. Gomes, L. Velho, and L. H. Figueiredo. Texturing implicit surfaces with particle systems. In *Technical Sketch in Visual Proceedings SIGGRAPH 1997*, page 172, August 1997.

[14] Ruben Zonenschein, Jonas Gomes, Luiz Velho, Luiz Henrique de Figueiredo, Brian Wyvill, and Mark Tigges. Texturing composite implicit objects with moving parts. Technical Report 98/617/08, University of Calgary, Dept. of Computer Science, 1998.