Laboratório VISGRAF

Instituto de Matemática Pura e Aplicada

Global 2-D Texture Mapping for Implicit Models

L.H. de Figueiredo, J. Gomes, M. Tigges, L. Velho, B. Wyvill, R. Zonenschein

Technical Report TR-02-09 Relatório Técnico

June - 2002 - Junho

Global 2D Texture Mapping for Implicit Models

L.H. de Figueiredo[1], J. Gomes[1], M. Tigges[2], L. Velho[1], B. Wyvill[2] and R. Zonenschein[1]

[1]Instituto de Matematica Pura e Aplicada, Rio de Janeiro, Brazil
[2]University of Calgary, Alberta, Canada

Abstract
An investigation on a method for applying 2D textures onto implicit models is presented. The method uses a dynamical system associated with the gradient vector field of an implicit function to acquire 2D texture coordinates on a support object. A complete set of extensions to the basic method is presented in order to integrate it on a complete system for modeling and rendering implicit surfaces.

1. Introduction

In this paper we discuss a method to apply 2D textures onto implicit models. We investigate its potential, its implications on modeling, visualization and user control of textures on implicit models.

Our goal is to devise a way to apply 2D textures onto implicit models so that it can be implemented in a complete system for implicit surfaces, with tools for texture placement, control and manipulation.

1.1. Problem and Motivation

Computer graphics can generally be subdivided in the areas of Modeling and Visualization, including techniques that enable one to create graphical objects in the computer and convert them into images.

Texture mapping is a successful area of research that aims to increase detail of an object associating attributes to its surface while unchanging its geometry. It is difficult to determine into which area, Modeling or Visualization, texture mapping should be classified. In fact, it embraces both areas, using tools and concepts related to each one (such as surface normals), making a liaison between the two main areas of Computer Graphics.

One of the open problems in computer graphics is the problem of applying a 2D texture onto implicit models. This is mainly due to the way an implicit model is described. Since the resulting model does not have a coordinate system defined on its surface, usual parametric texture mapping techniques can not be applied. Implicit models have then relied on 3D texture mapping or on parameterizing methods.

1.2. Related Work

This paper covers three distinct research areas of computer graphics: texture mapping, implicit surfaces and particle systems.

Texture mapping was first introduced by Ed Catmull[7]. The success and originality of the technique followed a series of proposals of uses and improovements, becaming a standard tool in computer graphics research and commercial projetcs.

The use of implicit functions in computer graphics was introduced by Jim Blinn[2] in his quest for a convinced framework to model molucules blending together.

Particle systems[13, 14] have been used in many ways in computer graphics, from modeling, visualization and animation.

1.3. Contribution

The few attempts to texture map implicit models available in the literature highlights how unsuitable those models are to 2D texture mapping.

Recently, we have published some papers [21, 22, 23, 24, 25] where we describe a dynamic system method to apply a 2D texture onto implicit surfaces.

On them we tackled some different difficulties of both implicit surfaces and texture mapping, discussing how our method could be extended and be useful to associate 2D textures to implicitly defined models. Our objective in this paper is to present a broader view of the method, particularly interested in integrating it to a complete system for modeling and visualization of implicit models.

2. Implicit models

In this section we present the basic concepts related to implicit surfaces in a general system for manipulating implicit models. This will be important when deriving a framework to implement the texture method in such system.

2.1. Definition

An implicit surface is defined as the set of points x in 3D space that satisfy an equation $F(x) = c$, where $F\colon \mathbf{R}^3 \to \mathbf{R}$ and $c \in \mathbf{R}$.

2.2. Operations

It is important in any good modeling system a consistent set of tools for manipulating the graphical object. In the case of implicitly defined models, a rich set of operations are possible that may change their geometry and topology. These operations can be classified in those that act on the range of the implicit function and those that act on its domain:

Domain of F: To apply a transformation to an implicit object, the argument x of its implicit function should be first transformed and then the implicit function must be evaluated.

$$F(x) -> F(T(x))$$

Range of F: One of the important advantages of implicit functions for computer graphics is that they allow a geometric operation by changing the level set of the implicit function.

2.3. Visualization

Visualizing an implicit model may be done in three different ways. Scan conversion is possible only for analytical descriptions of implicit models. Polygonization needs a decomposition of the space in cells. Ray tracing implicit surfaces may take advantage of.

3. Texture mapping

Texture mapping a graphical object increases detail associating attributes to a surface without changing its geometry.

3.1. Basic concepts

The problem of texture mapping may be defined as $m = g \cdot t$; the composition of two functions, g and t, as shown in the Figure 1. The mapping function g defines a mapping between two spaces; V, the object space and U, the texture space. The texture function t associates the texture space to an attribute space, such as color. Combined together these two functions allow one to texture map an object.

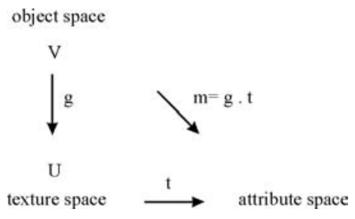

Figure 1: *Texture mapping.*

There are many ways to associate the texture space to the attribute space. Generally, the attribute space is a vector space and the association of texture space to it is achieved via an attribute function t. At the implementation stage this could be a look up table or a procedural function. The function t will not be discussed in this paper, and it is included here for the sake of completeness; it is important to stress, though, that the attribute itself is not an intrinsic part of the texture space. It is acquired at an attribute space through t.

The main difficulty of texture mapping is how the function g associates the object and texture spaces. This amounts to a deformation of the texture space onto the object sapce, which points to a warping problem. It also depends on how the geometric support of the graphical object and how the texture space are defined.

The easiest case is when the graphical object is a planar image; its geometric support is a square, and the mapping function g is the identity. See Figure 2.

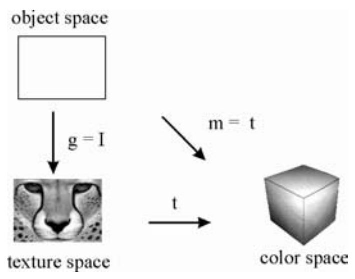

Figure 2: *Dissociating the geometric support and the attribute of the graphical object*

When the graphical object is a surface embedded in the 3D space we can achieve a similar identity mapping using a 3D texture space[11, 10]. In this case, the mapping function g is the restriction of the 3D texture to the point sets of the object. See Figure 3. This is a powerfull method that works well for any kind of surface, independently of its definition. The intuition is that we would be carving the surface out of a 3D block of textured material. This method is very usefull when texture mapping with materials that have a 3D structure in nature, such as wood and marble.

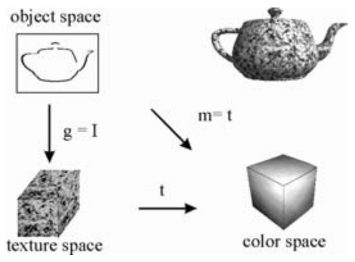

Figure 3: *3D Texture mapping.*

In this paper we are particularly interested in the use of 2D textures. This kind of mapping gives the idea of placing a label on an object, wraping it with a plastic sheet etc. If we can achieve such mapping we are able to mimmic some everyday real situations, such as placing wall paper in a room, sticking a label onto a bottle etc.

The association of such 2D texture space to the object space has a "natural" solution when the surface to be textured is parametricaly defined. In this case the inverse of the parametrization is used to associate both spaces. See Figure 4.

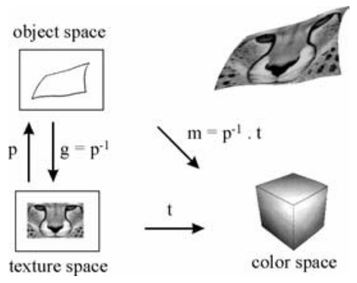

Figure 4: *2D Texture mapping.*

Although the use of 2D texture mapping onto parametric surfaces is largely used, there are some situations where things can get complex. This is the case when we do not have a global parametrization of the surface. Using a piecewise parametrization, we are still able to texture map each separate patch, but our problem becomes on how to maintain continuity between patches. This problem may be overcomed introducing the idea of projection mapping[1].

3.2. Projection mapping

Projection mapping is a two-step texture mapping technique[26]. Initially it applies the 2D texture onto a simple auxiliary object such as a cylinder or a sphere. We call it the support object for the texture (In fact, this concept is already "hidden" in basic 2D texture mapping, where the support surface for a 2D texture is commonly a square). The surface to be textured is then placed inside the support surface and the projection mapping takes place tracing rays from the support object until they reach the surface. This proved to be a really usefull technique since most commercial programans (such as Maya) have this feature implemented. An example of a cylindrical projection is shown in Figure 5.

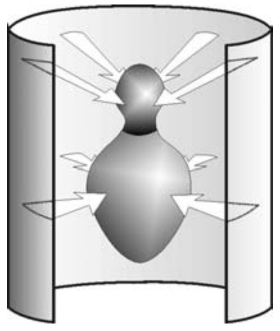

Figure 5: *Cylindrical projection.*

Although the result may be successfull, sometimes it is what has been desired. The reason is mainly because the projection method uses only information from the support surface, whithout taking any consideration to the surface properties, such as curvature.

In the next section we shall see that the method for applying 2D textures onto implicit surfaces described in this paper gets its inspiration from this observation.

3.3. Implicit surfaces

As described in section 2, an implicit function describes a family of level surfaces, one for each isovalue c. We have observed that the gradient vector field ∇F has a close relationship with the level surfaces, which follow the gradient orthogonally, as can be seen in Figure 6.

This observation gives an intuitive sense that an implicit surface may evolve smoothly into its neighbors

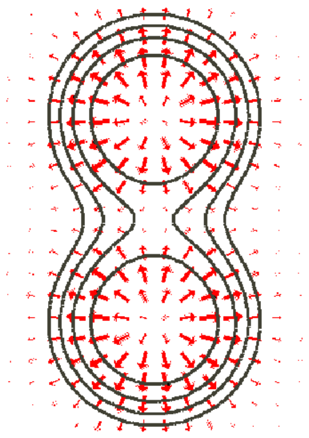

Figure 6: *Level surfaces follow gradient vector field.*

level sets. In other words, an implicit surface may be projected to a neighbor level surface through its gradient vector field. This projective characteristic is the starting point of the texture method described in the next section.

4. Projection Mapping for Implicit Models

4.1. Overview

Let S be one of the level surfaces defined by the implicit function F. We may assume that the isovalue corresponding to S is $c = 0$, i.e., that $S = F^{-1}(0)$. In the same manner, let T be the level surface defined by the implicit function G when its isovalue is $d = 0$. We call T the support surface for the texture, as we will initially apply the 2D texture on it.

Geometrically, we set T in such a way that it surrounds S. Our goal is to generate a force field defined in the ambient space so that an association of points on S and points on T, where a 2D texture is already defined, can be achieved. We employ the gradient vector fields ∇F and $-\nabla G$ to generate such force field.

In Figure 7 we can visualize a combination of ∇F and $-\nabla G$ associating S and T.

The support surface T should be simple enough so that it is very easy to define texture coordinates on it (e.g., a cylinder, sphere etc.). For this reason, we select a surface that has both a parametric and an implicit description. The parametric description allows us to define a texture on T. The implicit description is useful for detecting where should the force field end.

The two steps of this texture mapping method is illustraded in Figure 8.

In summary, the method comprehends:

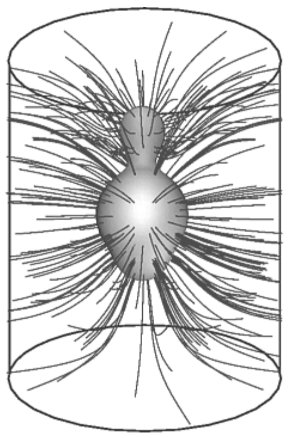

Figure 7: *An implicit model surrounded by a support surface and the force field generated by a combination of the* ∇F *and* -∇G.

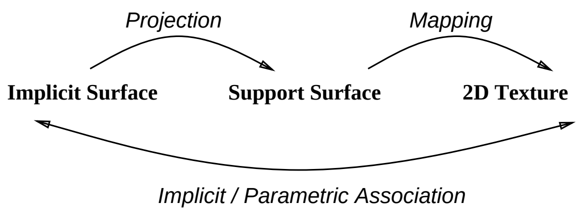

Figure 8: *The method and its steps.*

- *Projection step*: The first step deals with the construction of a force field that associates an implicitly defined model to a support object for the texture. For that, we use the gradient vector field of the implicit functions that define S and T.
- *Mapping step*: The second step refers to the mapping of a 2D texture source onto a parametric description of the support object T.

In combination, these two steps produce the final association of points in an implicit model and points in a 2D texture source.

4.2. Numerical simulation

The method establishes a correspondence between the implicit model and the support surface creating a particle system. This dynamic system runs along the force field created from the implicit and support surfaces gradient vector fields. The motion of a particle is governed by the differential equation

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \alpha\nabla F - (1-\alpha)\nabla G = 0,$$

where x is the position of the particle, t is time, γ is a viscosity constant and $\alpha \in [0,1]$ is a user-controllable relative weight.

5. Composite objects

As discussed in section 2, it is common in modeling systems to create a complex model through the composition of hierarchies of simpler primitives. In the case of implicit models, such composition is achieved through a combination operator C applied to the implicit primitives. The result is also an implicit function:

$$F_c = C(F_1, F_2, \ldots, F_n).$$

In fact, implicit models were first introduced to the computer graphics community in a work by Jim Blinn[2] where he searches for a reliable description of a combination of molecules blending together.

If the result of a combination operator C is a differentiable implicit function F_c, the method described in section 4 is suitable to assign 2D textures onto it. Actually, all examples we have shown so far were done using a composite of implicit primitives.

There are some situations, though, when the method described is not suitable to texture map composite implicit objects:

1. The implicit model may be extremely complex, and a simple support surface with a geometric similarity with the model can not be found.
2. We may want to use the characteristic of composite implicit objects whose shape may change in time. Either an amorphous or an articulated object will show a sliding texture if a unique and static support surface is used.

To deal with these two problems, we may extend our method setting a particular support surface for each primitive of the composite model. Each support surface is bound to its primitive affine transformations (see Figure 9).

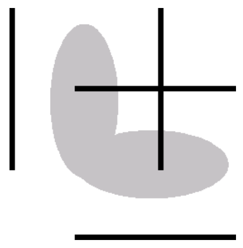

Figure 9: *Two primitives and their support surfaces.*

In the following subsections we discuss how the extended method is applyed in two kinds of composite implicit models: those created by booleans and those created with blends.

5.1. CSG Operations

The combination operators of union and intersection for implicit primitives are described in the following equations:

$$C_\cup = \max(F_1, F_2, \ldots, F_n).$$

$$C_\cap = \min(F_1, F_2, \ldots, F_n).$$

Setting a separate support surface for each primitive is sufficient to assign 2D textures to composite CSG implicit models. When simulating the dynamic system, a particle takes its texture coordinate value from the intersection with its support surface 9

Even when the composite object changes its shape via affine transformations applied to its primitives, texture consistency is maintained 10.

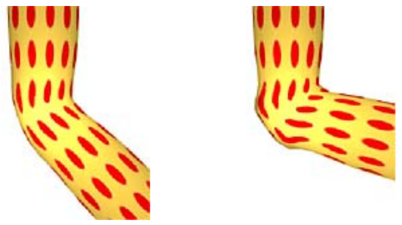

Figure 10: *A textured CSG implicit object in two different shapes.*

5.2. Blends

Care should be taken when applying 2D textures onto blended composite implicit objects. A blending combinator operator C performs smooth transitions between implicit primitives. A linear blend, for example, is given by

$$C = \sum_{i=1}^{n} F_i.$$

It is clear that particles that are outside the blending area should behave as in the previous subsection: it should acquire texture coordinates at the primitive's support surface. The problem resides at particles at the blending area. Figure 11 illustrates the area where the support surfaces overlaps, indicating the associated primitives' blending area.

To tackle this problem, we look at the ideal (but rather difficult) solution as an intuitive guide: if we

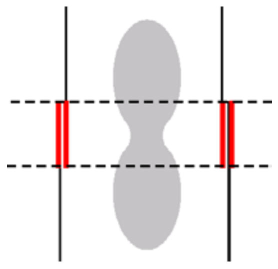

Figure 11: *The support surfaces associated to the primitives' blending area.*

were able to blend the support surfaces that contribute to a particle at the object's blending area, the particle trajectory would have a natural intersection point at it. See Figure 12.

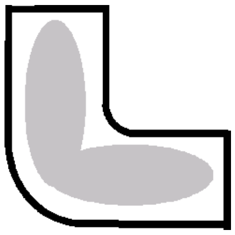

Figure 12: *The ideal unique blended support surfaces.*

Even if this solution was feasible, we had to assume that the texture on each support surface area that would be blended had to be the same. This assumption is important so that ghosting effects due to overlapping textures would not occur.

Although this solution is a difficult task, we are able to perform a similar approach performing a blend of support surface transformations. This is achieved using the contribution attributes α_i of each implicit function F_i to the composite implicit object F_c. For each particle at the blending area, these weighting attributes are used to perform a linear combination of the affine transformation applied to each support surface. The simulation is then run once for each particle, each of which associated with a specific support surface T transformed by a combined transformation M_c.

The method consists of the following steps:

1. For each particle at rest at the surface, compute the contribution of each primitive to F_c: $\alpha_1, \ldots, \alpha_n$.
2. Simulate the motion of each particle and test its trajectory against a support surface T transformed by M_c, which is positioned using the proportion α_i of each primitive to F_c.
3. Texture coordinates are read at the intersection with the transformed support object M_c.

Figure 13 illustrates the method. If a particle is at a surface point where only one primitive contributes to the implicit surface iso-value, the trajectory is computed with respect to the transformation for that primitive. Particles which originate in a blend area of the surface use the linear combination of the transformations of the components contributing to the field value of the surface location. Note that the support surfaces T have to be the same (geometry and texture) in the blending parts. Also note the similarity of this approach to the ideal support surfaces blend in Figure 12.

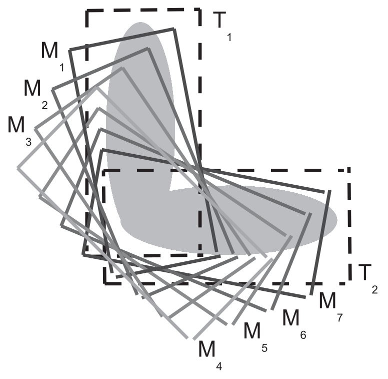

Figure 13: *Transformations:* M_1*:* $\alpha_1 = 0.9, \alpha_2 = 0.1$*;* M_4*:* $\alpha_1 = 0.5, \alpha_2 = 0.5$*;* M_7*:* $\alpha_1 = 0.1, \alpha_2 = 0.9$.

6. Control

The main contribution of the method is the possibility of global placement of 2D textures onto implicit models. Although this can be considered a good step forward to the problem, the practical world of texture mapping demands more ellaborated tools. To effectively take advantage of a texture mapping technique, it is important to be able to control how the texture is mapped onto the surface. In traditional parametric texture mapping, a user may be given tools to place a texture map at any desired position and orientation on a surface, scale the map, tile it along the surface, stretch it etc. In this section we describe how the dynamic system method may be used to give similar tools for controlling the 2D texture placement onto implicit models.

6.1. Framework

In a user point of view, a texture mapping system should initially enable automatic and semi-automatic solutions. Once a texture mapping exists, refinement tools should be available for both global and local control of texture placement.

The dynamic system method is a good candidate to provide such tools. After choosing a support object for the texture and initially placing it, the user would run the system to achieve an initial texture mapping. Thereafter, while not satisfied with the result, the user may change it globally or locally using some of the tools that the dynamic system allows us to create. These are tools related to different properties of the method, named:

1. Dynamic: tools that use the time component of the dynamic system.
2. Geometric: geometric transformations applied to objects and external forces.
3. Parametric: varying attributes.

In the following sections we will describe those properties, exemplifying with control tools.

6.2. Dynamic

The main characteristic of the method is its dynamic behaviour. Any change in this system will modify the resulting texture mapping. Although the geometric and parametric characteristics that we analyse in the next sections do as well influence the dynamic system, in this section we introduce the tools that have a closer relationship with the dynamic behaviour. These are tools that introduce other elements to the system, such as attractors and repulsors, and tools that act using the time component of the dynamic system.

The extend on which each object's gradient vector field will act during the simulation have a direct inpact on the resulting texture placement. This may be controlled using the α parameter of the motion equation. Besides, α may be a function of time, enhancing the influence of the gradient vector field of each object depending on the proximity to it. The motion equation should then be changed to the following:

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \alpha(t)\nabla F - (1-\alpha(t))\nabla G = 0.$$

Local control may be achieved introducing external forces to the medium. This could be attractive and repulsive point, curve or surface sources. Figure 14 shows how an external force may act on a region of the force field.

6.3. Geometric

The relative positions of S and T and of any other external force play a major role over the global placement of the texture. Any transformation applied to them will modify the overall force field. In Figure 15 diferent relative positions of implicit and support objects generates changes in the force field (a) and in the texture placement (b).

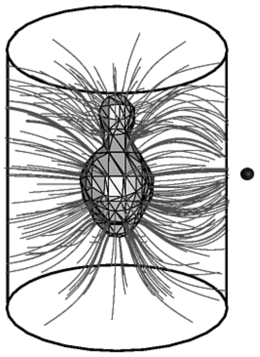

Figure 14: *An external force changing the force field locally.*

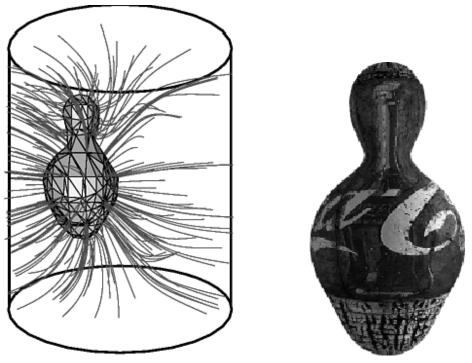

Figure 15: *Changes on the relative positions of S and T and its effect on the force field and on the texture mapping.*

In the same manner, an external force may be moved to locally change the texture mapping. See Figure 16.

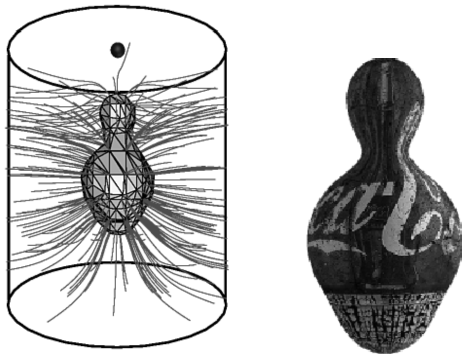

Figure 16: *Moving a repulsor to locally change its influence over the force field interfering on the texture mapping.*

6.4. Parametric

Various parametric variables are available for both global and local manipulation of the force field. The functions that generate and influence the dynamic system are some of them; by varying their parameters we can globally and locally control the texture placement. In Figure 17 we vary the area of influence of a repulsor to increase a squeezing effect comparing to Figure 16.

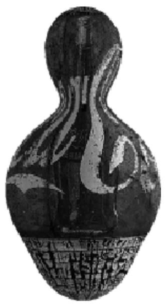

Figure 17: *Increasing the area of influence of a repulsor and its effect on the resulting texture mapping.*

As a projection mapping technique, the texture map is distorted to follow the model curvatures. Although this may be desired, it is important to have a tool to magnify the extension of such texture distortion. Such tool may be created from the parametric information that the method results: the simplicial approximation of the implicit model mapped onto a 2D texture source. See Figure 18.

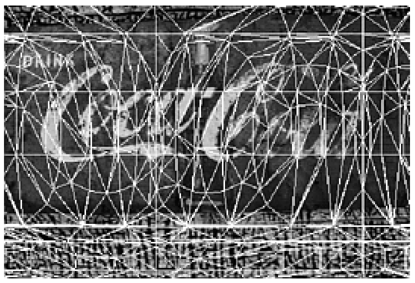

Figure 18: *A simplicial approximation of an implicit surface mapped onto a 2D texture source.*

The source of information illustrated in Figure 18 is also a rich data for local adjust of the texture mapping. For example, one can select one or more surface vertices and move their mapped counterpart at the texture source, wrapping it. This is prticullarly important when a match between model and texture features is desired. In Figures 19, 20 and 21 we see how such match may be achieved.

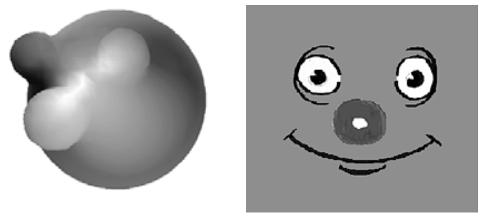

Figure 19: *The object shaded and the texture source.*

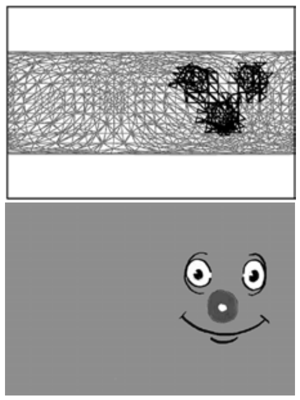

Figure 20: *The object simplicial approximation mapped onto the texture and the texture source translated and warped to match the features.*

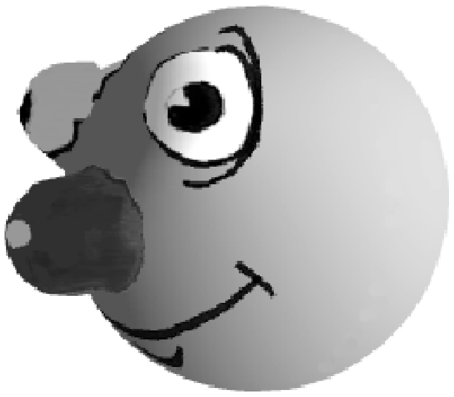

Figure 21: *The resulting textured object.*

In addition, a 3D paint tool can be created. As a mapping exists, this problem can be regarded as user interface one. With it, the user should be able to paint at the 2D geometric support of the texture while seeing the result at the implicit model.

7. Conclusion

The method we discussed in this paper has the advantage of creating a framework for both texture placement and manipulation. The use of an inherent characteristic of implicit surfaces, its gradient vector field, to generate a dinamic system, creates a rich environment for creating tools to be used in a complete system for implicit surfaces.

References

1. A. Barr, *Decals.* In SIGGRAPH 83 State-of-the-Art of Image Synthesis Course Notes.
2. J. F. Blinn, *A generalization of algebraic surface drawing.* ACM Transactions on Graphics, 1(3), pp. 235–256, 1982.
3. J. Bloomenthal (ed.), *Introduction to Implicit Surfaces.* Morgan Kaufmann, 1997.
4. J. Bloomenthal, B. Wyvill, *Interactive techniques for implicit modeling.* Proceedings of 1990 Symposium on Interactive 3D Graphics, pp. 109–116, 1990.
5. L. H. de Figueiredo, J. Gomes, D. Terzopoulos, L. Velho, *Physically based methods for polygonization of implicit surfaces.* Proceedings of Graphics Interface '92, pp. 250–257.
6. L. H. de Figueiredo, J. Gomes, *Sampling implicit surfaces with physically-Based particle systems.* Computer & Graphics, 20(3), 1996, pp. 365–375.
7. E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD Thesis, University of Utah, 1974.
8. P. S. Heckbert, *Survey of Texture Mapping.* IEEE Computer Graphics and Applications, pp. 56-67, November, 1986.
9. H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, K. Omura, *Object modeling by distribution function and a method of image generation.* Trans. IECE Japan, Part D J68-D(4), pp 718–725, 1985.
10. D. Peachey, *Solid texturing of complex surfaces.* Proceedings of SIGGRAPH '85, pp. 279–286.
11. K. Perlin, *An image synthesizer.* Proceedings of SIGGRAPH '85, pp. 287–294.
12. H. K. Pedersen, *Decorating implicit surfaces.* Proceedings of the SIGGRAPH '95, pp. 291–300.
13. W. T. Reeves, *Particle Systems - A Technique for Modeling a Class of Fuzzy Objects.* Computer Graphics, vol. 17, no. 3, pp 359-376, 1983.
14. W. T. Reeves, *Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems.* Computer Graphics, vol. 19, no. 3, pp 313-322, 1985.
15. J.-P. Smets-Solanes, *Vector field based texture mapping of animated implicit objects.* Eurographics '96 Conference Proceedings, Poitier, France, 1996.
16. L. Velho, *Simple and efficient polygonization of implicit surfaces.* Journal of Graphical Tools, 1(2), 1996, pp. 5–24.
17. L. Velho, L. H. de Figueiredo and J. Gomes, *A methodology for piecewise linear approximation of surfaces.* Journal of the Brazilian Computer Society, 3(3), 1997, pp. 30–42.
18. B. Wyvill, C. McPheeters, G. Wyvill, *Data structures for soft objects.* The Visual Computer, 2(4), pp. 227–234, 1986.
19. G. Wyvill, B. Wyvill, C. McPheeters, *Solid texturing of soft objects.* IEEE Computer Graphics & Applications 7(12), 1987, pp. 20–26.
20. J. Yngvesson, I. Wallin, *SIPP – a 3D rendering library.* Available at ftp://ftp.isy.liu.se/pub/sipp.
21. R. Zonenschein, J. Gomes, L. Velho, L. H. de Figueiredo, *Textura de superfícies implícitas com sistemas de partículas.* Proceedings of SIBGRAPI '95 pp. 305–306.
22. R. Zonenschein, J. Gomes, L. Velho, L. H. de Figueiredo, *Texturing implicit surfaces with particle systems.* Visual Proceedings, SIGGRAPH '97, p. 172. http://www.visgraf.impa.br/Projects/dtexture/.
23. R. Zonenschein, J. Gomes, L. Velho, L. H. de Figueiredo, *Controlling Texture Mapping onto Implicit Surfaces with Particle Systems.* Proceedings of the Third International Workshop on Implicit Surfaces, pp.131-138.
24. R. Zonenschein, J. Gomes, L. Velho, L. H. de Figueiredo, M. Tigges, B. Wyvill, *Texturing Composite Deformable Implicit Objects.* Proceedings of SIBGRAPI '98, pp. 346–353.
25. R. Zonenschein, J. Gomes, L. Velho, N. Rodriguez, *Towards Interactivity on Texturing Implicit Surfaces: A Distributed Approach.* Proceedings of WSCG'2001 - 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 360–366.
26. E. A. Bier and K. R. Sloan Jr., *Two Part Texture Mappings.* IEEE Computer Graphics and Applications, Volume 6, Number 9, 1986, pp. 40–53.