Combining implicit surfaces with soft blending in a CSG tree

L. Barthe V. Gaildrat R. Caubet

Abstract

In this paper, we show an extension of the model proposed by Sabourdy, combining various implicit surfaces with soft blending capacities in a CSG tree. This extension allows integration of plane surfaces, skeletons and many other types of implicit surfaces. It particularly develops the blending possibilities offered by this approach. The model is then able to generate a wide variety of shapes on which the user may act through a limited number of parameters.

Introduction

Modelling complex shapes with CSG trees is an efficient and natural method. Indeed, union, intersection and difference are Boolean composition operators which represent a sort of 'manual' way of manipulating. But, used alone, this approach fails to obtain soft transitions between the different primitives composing the object.

For each function f, an implicit surface is defined by its equation f(x,y,z) = k. Many different implicit models exist, each based on specific implicit functions generating particular forms. Some of them blend [3, 4, 8], others twist and sweep [5], control which elements blend together using a graph [6, 18], but all are isolated implicit models. In composing these models, we considerably enrich the forms we can produce. Even though the first models combining implicit surfaces in CSG trees (using *min* and *max* functions) [11] did not give soft transitions, most composition models now allow the introduction of blend in the transitions. Pasko [10] uses the Boolean operators (union, intersection and difference) to compose R-functions which is a very general approach. Wyvill [16, 17] limits his approach to skeleton-based implicit surfaces but treats blending and space warping in the same way as union, difference and intersection. These two models, are very efficient but Pasko's model suffers from a lack of blend variety and Wyvill's model from lack of shape variety (which stays rounded). It would also be interesting to introduce, in this type of model, the notion of influence radius to maintain blends where they are wanted. The model developed by Sabourdy [13] uses CSG binary trees to combine any implicit surfaces defined by an equation $f(x,y,z) = 0$ (if $f(x,y,z) < 0$ then the point (x,y,z) is inside the object bound by the surface; if $f(x,y,z) > 0$ then the point (x,y,z) is outside). Furthermore, these are combined with exponential blending functions.

$O_1 \cup O_2$ is defined by :

$$-\exp(-b.f_1(x,y,z)) - \exp(-b.f_2(x,y,z)) + 1 = 0.$$

$O_1 \setminus O_2$ is defined by :

$$-\exp(-b.f_1(x,y,z)) + \exp(-b.f_2(x,y,z)) + 1 = 0.$$

$O_1 \cap O_2$ is defined by :

$$\exp(b.f_1(x,y,z)) + \exp(b.f_2(x,y,z)) - 1 = 0.$$

b is a positive constant.

This model allows the introduction of many different types of implicit surfaces, including plane surfaces, and thus a larger number of different objects and shapes. But three main inconveniences remain :

- The use of binary CSG when n-ary CSG would be preferable.
- The results of the difference is only an approximation to the Boolean operator.
- The use of exponential functions to perform the blending.

In spite of those inconveniences the model is sufficiently interesting to be studied and improved, which is the object of this paper.

Extending the binary into a n-ary CSG model

The binary CSG model combines two different objects, creating a new equation that defines a new object. That is why the use of the same blending function at different tree levels gives a different result, whereas one might require the same transition between each primitive: for example when the final object should be the union of n different objects (as in the 'Blobby Model' [1, 2, 9]).

Similarities between the structures of global blending based on skeletons [4, 3] (that is

$$\sum_{S_i} g_i(r_i) = C$$

where S_i is the skeleton, r_i is the distance from P to S_i and g_i is a blending function, positive and decreasing)

and those used by the union in Sabourdy's CSG model, of the form

$$-\exp(-b.f_1(x,y,z)) - \exp(-b.f_2(x,y,z)) + 1 = 0,$$

suggested the n-ary union structure i.e. :

$$\bigcup_{i=1}^{n} O_i \text{ is defined by } \quad 1 - \sum_i g_i(f_i(P)) = 0,$$

with f_i defining the implicit surface and g_i the associated blending function. The properties of these two functions remain to be specified.

From this structure the shape of intersection and of difference can be easily deduced. The complement operator is still -f if the surface of the object is defined by f :

$$\bigcap_i O_i = \neg \bigcup_i \neg O_i,$$

so $\bigcap_i O_i$ is defined by $-1 + \sum_i g_i(-f_i(P)) = 0$.

$$O_1 \setminus O_2 = O_1 \cap \neg O_2,$$

so $O_1 \setminus O_2$ is defined by $-1 + g_1(-f_1(P)) + g_2(f_2(P)) = 0$.

We can then extend this structure to :

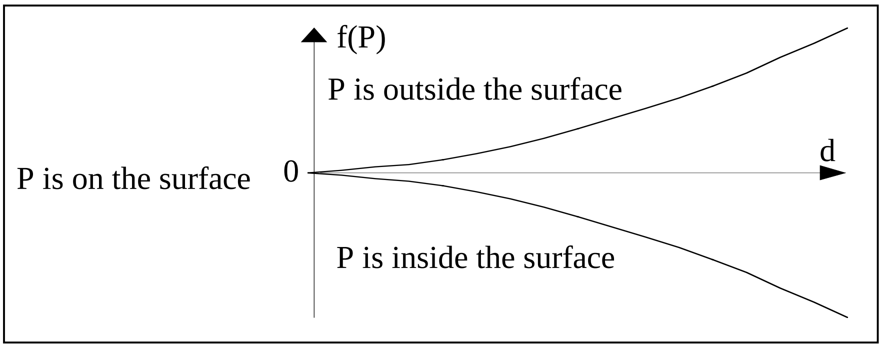
$$\bigcap_{i=1}^{l} O_i \bigcap_{j=l+1}^{n} \neg O_j \text{ is defined by } -1 + \sum_{i=1}^{l} g_i(-f_i(P)) + \sum_{j=l+1}^{n} g_j(f_j(P)) = 0.$$

The results obtained with Boolean algebra are exact, including the difference.

Implicit surfaces

Our model requires certain properties we will now investigate, and then show how the integration of some types of functions can be carried out as basic primitives, knowing that, in order to have an homogeneous model, the functions at each level of the CSG tree must keep those same properties.

Implicit surface properties



d is the distance between point P and the surface

Figure 1.

In order to have a general model that may use as many types of implicit surfaces as possible, the properties of the functions have to be specified according to certain requirements :

- If P is a point in space, the shape of the function must be of the type f(P) = 0 with :

$f(P) > 0$ P is outside the volume limited by the surface.

$f(P) < 0$ P is inside the volume limited by the surface.

- For the points located outside the surface, the function f must increase with the distance between the point P and the surface. For the points located inside the surface, the function f must decrease with the increasing of the distance between the point P and the surface.
- The function f must be differentiable in all points P of space in order to allow computation of the surface normal for lighting

purposes. If that is not the case, the function must at least be continuous to avoid discontinuities in the surface. The surface normal is then computed using the display model (voxels, polygons,...) or by sampling the field value in small increments in each direction.

Such a function has an overall aspect as shown in Figure 1.

Integration of algebraic functions

Algebraic functions [7, 12] are defined with a polynomial function containing tree spatial variables (x,y,z). They can be shown in a normal form :

$$\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} a_{ijk} x^i y^j z^k = 0 .$$

For example, the equation of a sphere could be :

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - R^2 = 0 .$$

The properties described above have to be checked, particularly concerning the notions of ‘inside’ and ‘outside’ the surface.

A particularly interesting algebraic surface is the plane surface (defined by an equation : $ax + by + cz + d = 0$) because it allows the creation of a greater variety of shapes.

Integration of skeletons

Implicit surfaces defined by skeletons are of the following type :

$$d(P,S)^2 = C^2$$

where d represents the distance between the point P and the skeleton S and C represents the distance between the skeleton the ‘iso-surface’.

In order to get a function having the required properties, one only needs to write the equation as follows :

$$d(P,S)^2 - C^2 = 0 .$$

Integration of a complex object defined by implicit surface

It may be interesting to use an object created with a different composition model, using its defining function as a basic primitive in our model, in order to add elements, to make improvements that the original composition model could not realise. The function defining this object must nevertheless comply with the properties of our model. For example, an object created with a global blending based on skeletons [4, 3] can easily be used if written as :

$$C - \sum_{S_i} g_i(r_i) = 0\,.$$

Models using *min* and *max* functions require care with the differential calculus, to make sure that the correct function is chosen at each point.

Blending functions

The blending function is designed to model the influence that one surface exerts on its surroundings. This function defines the contribution of one object to the blend; it makes the soft transition and allows the control of its smoothness. One blending function will give different results depending on the implicit function to which it is associated, owing to the wide variety of types of implicit surfaces that can be used in our model. Furthermore, the abruptness of the transition between the primitives being mixed should be controlled. Thus it is necessary to have a wide set of different blending functions available. Knowing the properties of the implicit, one can describe the properties blending functions should have.

Properties of blending functions

In Sabourdy’s model, blending is performed with an exponential function. It is combined with the implicit surface function which is a 0 ‘iso-surface’ to produce a 1 ‘iso-surface’ (exp(0) = 1). The new surface is identical to the initial one, except that it now includes the blending influence of the surrounding objects. This influence declines

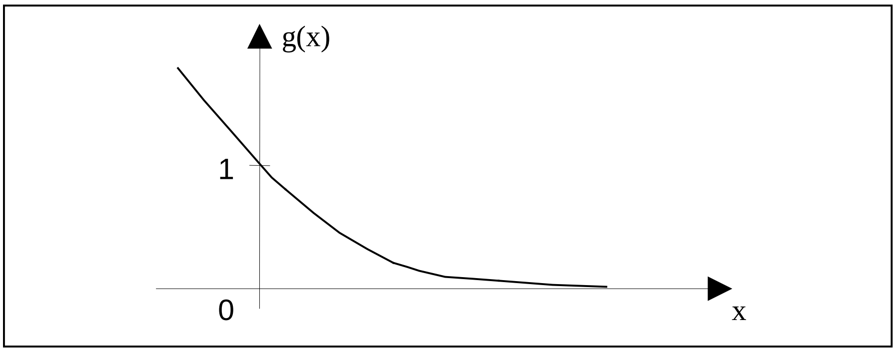


Figure 2.

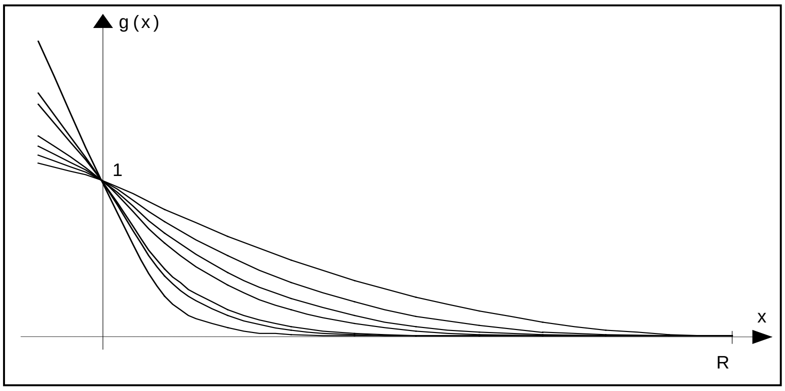
when point P is further away from the surface, due to the decreasing exponential function (exp (-x) is a decreasing function). The composition is then carried out by adding (union) the 1 'iso-surfaces' and, in order to work on CSG trees, the function obtained after composition must have the same properties as that used for the composition, which is obtained by subtracting 1 to the resulting function. Thus the *g* blending function must have certain properties, described as follows :

- *g* is a function defined in R.
- *g* must be monotonous, continuous and decreasing in R.
- *g*(0) = 1.
- $\lim_{x \to -\infty} g(x) = +\infty$ and $\lim_{x \to +\infty} g(x) = 0$.
- In order to compute the surface normal resulting from the blend, g must be differentiable in R. The derivative should be continuous to avoid discontinuity on the surface.

An example of such a function *g* is shown in Figure 2.

Group of functions of a 'Wyvill' blending type

Wyvill proposes a polynomial weighting function [15, 4] that allows the blend of implicit surfaces defined by skeletons. With this function, the contribution to the blend of the object vanishes, beyond a distance R. Our aim being to combine different type of implicit surfaces, this



From left to right : a=1 n=12, a=1 n=8, a=0.01 n=8, a=1 n=4, a=0.01 n=4, a=1 n=2, a=0.01 n=2.

Figure 3.

parameter may not always be a distance (we will investigate this point in the section *'Control of the influence radius'*).

$$G(r) = -\frac{4}{9.R^6}.\left(r^2 - R^2\right)^2.\left(r^2 - \left(\frac{3}{2}.R\right)^2\right) \quad \text{if r} \leq \text{R}$$

$$G(r) = 0 \quad \text{otherwise.}$$

This function is based on squared distance, which actually represents the equation of the skeleton implicit surface. From this observation it was possible to adapt this blending function so that it would be compatible with our model :

$$g(x) = \frac{4}{9.R^3}.(x - R)^2.\left(\frac{9.R}{4} - x\right) \quad \text{if x} \leq \text{R}$$

$$g(x) = 0 \quad \text{otherwise.}$$

It is now necessary to introduce parameters into this function so as to obtain a group of functions allowing the control of the smoothness of the transitions with a given influence radius. The parameters are **n** and **a,** where **n** adjusts the function by steps and **a** realizes a fine tuning in-between the steps. Figure 3 shows some examples of blending functions with different values of **n** and **a**.

$\forall$ a $\in$]0,1] and $\forall$ n $\in$ N^*, n pair

$$g_{an}(x) = \frac{a}{R^{n+1}}.(x-R)^n.\left(\frac{R}{a}-x\right) \quad \text{if x} \le \text{R}$$

$$g_{an}(x) = 0 \ \text{otherwise.}$$

A curved curb (n > 8 for example) gives a abrupt transition whereas a straight curb (n = 2) gives a soft transition. Thus, if we are not limited by the abruptness of the transition, we are restricted by the minimum value of *n*, and thus the softness of the transition. We are then led to find a new group of blending functions that could allow a softer transition (Images 1,2,3 represent the intersection of six plane surfaces with different value of *a* and *n*).

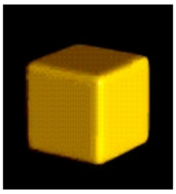


Image 1 (a=1, n=8).

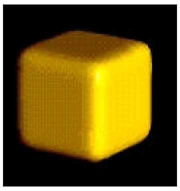


Image 2 (a=1, n=4).

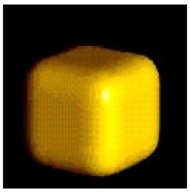


Image 3 (a=1, n=2).

Group of functions of a 'Stolte' blending type

Stolte also proposes a polynomial weighting function [14] that allows the blend of skeleton-based surfaces, with a given influence radius :

$$G(r) = \frac{1}{R^8}.\left(r^2 - R^2\right)^4 \quad \text{if r} \le \text{R}$$

$$G(r) = 0 \quad \text{otherwise.}$$

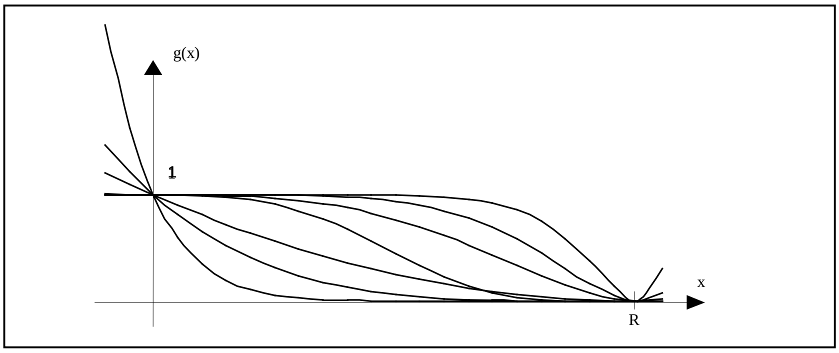
This function can also be adapted to our requirements with **m** and **n** parameters :

$\forall$ m $\in$ N^*, m impair and $\forall$ n $\in$ N^*, n pair

$$g_{nm}(x) = \frac{1}{R^{m.n}}.\left(R^m - x^m\right)^n \quad \text{if x} \le \text{R}$$

$$g_{nm}(x) = 0 \ \text{otherwise.}$$

Figure 4 shows that the new group represents a wider family of functions, allowing control of abruptness and softness of transitions with a C^1 continuity. Furthermore, when controlling softness, the



From left to right : m=1 n=10, m=1 n=4, m=1 n=2, m=3 n=6, m=3 n=2, m=5 n=2, m=9 n=2.

Figure 4.

object created can be inflated at transition level, and in this case, the continuity of contact between the primitives and the blends stays C^1. Classical blending processes now seem to be easily mastered with the help of *m* and *n* parameters modifying the family of blending functions (Images 4,5,6 represent the intersection of six plane surfaces with different value of m and n).

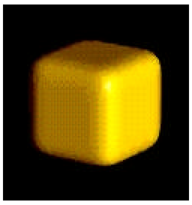


Image 4 (m=1, n=4).

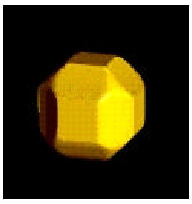


Image 5 (m=3, n=8).

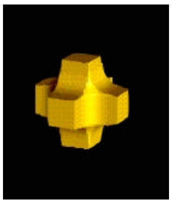


Image 6 (m=5, n=2).

Linear blending function

It is also possible, if needed, to create specific blending functions in order to get special effects. For instance, we propose a linear blending function shown in Figure 5:

$$g(x) = -\frac{1}{R}.x + 1 \text{ if } x \leq R$$

$$g(x) = 0 \quad \text{otherwise.}$$

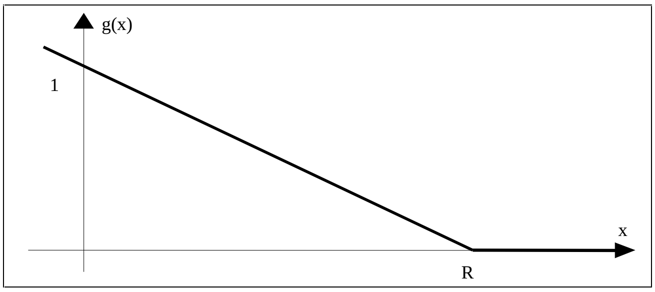
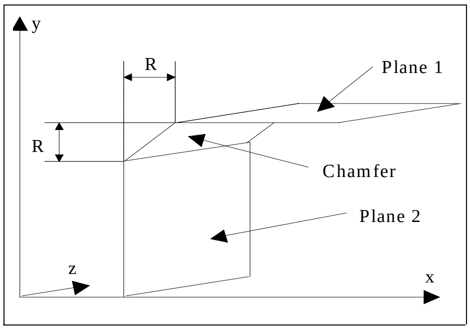
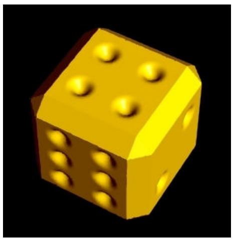


Figure 5.

This function is particularly useful to get the intersection of plane surfaces. It offers the possibility to realize ‘chamfers’ (Figure 6, Image 7) between different plane surfaces. The combination of plane surface equations gives a chamfer plane surface equation through the use of this linear blending function. In the same way, if two spheres are composed with a sufficient influence radius, they are joined by a sphere (Image 8 where the middle ‘sphere’ is the result of the blend). In fact, this function reports the primitive forms in the blend.





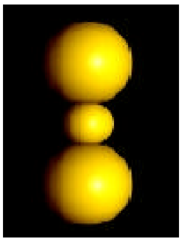


Figure 6. Image 7. Image 8.

Non-blending functions

It is also useful to be able to combine different objects without a blend. In this case, we need to create a ‘blending function’ which allows abrupt transition between non-disjoint objects or combines objects without blending them. For example, when showing two fingers of a hand, we need to be able to show them separate and not influenced one by the other.

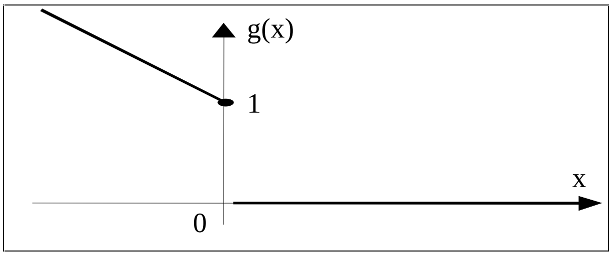


Figure 7.

$$g(x) = -x + 1 \quad \text{if } x \leq 0$$

$$g(x) = 0 \quad \text{otherwise.}$$

The main inconvenience of this function lies in the fact that it is discontinuous, owing to influence radius going to zero (as seen in Figure 7). Results obtained using this function can not be used easily in further combinations. It is then preferable (if possible) to use it last.

Control of the influence radius

A model may incorporate various types of implicit surfaces. Skeleton-based implicit surfaces have an influence radius related to distance, but that is not the case of all types of implicit surface. When combining different types at each new level of the CSG tree, the R value is no longer related to distance. This parameter has to be controlled in a specific way we will now investigate.

In our formulation, the surface is defined by points in space P (x,y,z), chosen so that f(P) = 0. That is to say, points P are such that g(f(P)) = 1. The blending functions g are designed so as g(R) = 0. The boundary of the field of influence of the object is now defined by :

$$V = \{ P' \in \text{space} \,/\, f(P') = R \}.$$

A simple way of defining R consists in choosing a point P' (x',y',z') that we want to position on the frontier V and then to take R= f(P') (as illustrated in Figure 8).

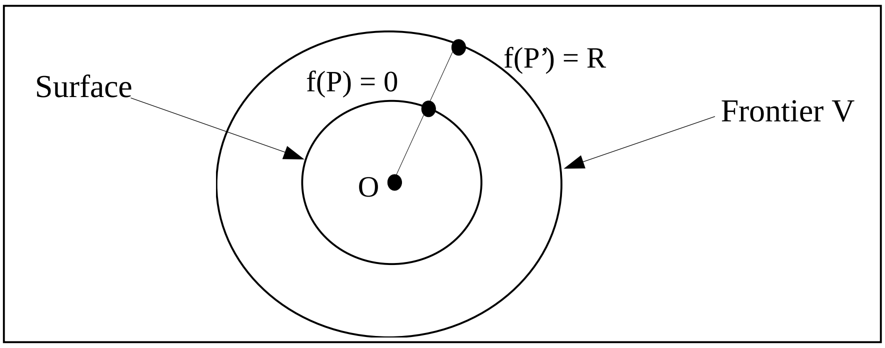


Figure 8.

Conclusion

Our model represents an efficient instrument for creating complex objects with implicit surfaces combined in a CSG n-ary tree. Owing to this CSG technique, we are able to include the Boolean intersection and difference operators, allowing, for instance**,** plane surfaces to be included in the primitives available to the user. The wide variety of shapes that can be produced has progressed greatly and is no longer limited to rounded shapes. The basic primitives used in our model, even when initially created through another tool, have different mathematical bases. The problems of control of smoothness of transitions implied by those differences are solved by the variety of blending functions proposed and by the easy adjustment of the influence radius from a point chosen by the user.

The result of the application of the model is an object described by a single equation that combines all the basic primitives and blending functions used in the process. This equation can be easily converted, using interval arithmetic, into voxels, directly displayable on the screen. This gives a high resolution to the images and allows interactive visualization.

Particular shapes, like sweep objects, can be used as basic primitives. It could be interesting to study the effects of our blending functions on these special implicit surfaces.

Furthermore considering the length of time needed to compute each new shape at each level of composition, it would be useful to have an interactive technique to follow, step by step, the construction of the shape being built.

References

1. H.B. Bidasaria, "Defining and rendering of textured objects through the use of exponential function", *Graphical Models and Image Processing*, 54, 2 (97-102), 1992.
2. J. Blinn, "A Generalization of algebraic surface drawing", *ACM Transaction on Graphics*, 1, 3 (235-256), July 1982.
3. J. Bloomenthal and K. Shoemake, "Convolution surfaces", *Computer Graphics*, 25, 4 (251-255), July 1991.
4. J. Bloomenthal and B. Wyvill, "Interactive techniques for implicit modeling", *Computer Graphics*, 24, 2 (109-116), March 1990.
5. B. Crespin, C. Blanc and C. Schlick, "Implicit sweep objects", *Eurographics'96*, 15, 3 (C165-C174), 1996.
6. M.P. Gascuel, "An implicit formulation for precise contact modeling between flexible solids", *Computer Graphics (Proc. SIGGRAPH 93)*, (313-320), August 1993.
7. P. Hanrahan, "Ray tracing algebraic surfaces", *Computer Graphics*, 17, 3, July 1983.
8. Z. Kasic-Alesic and B. Wyvill, "Controlled blending of procedural implicit surfaces", *Proc. Graphics Interface 1991*, (236-245), 1991.
9. S. Muraki, "Volumetric shape description of range data using *'Blobby Model'*", *Computer Graphics*, 25, 4, July 1991.
10. A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko, "Function representation in geometric modeling : concepts, implementation and applications", *The Visual Computer*, 8, 2 (429-446), 1995.
11. A. Ricci, "A constructive geometry for computer graphics ", *computer journal*, 16, 2 (157-160), May 1973.
12. T. Sederberg, "Techniques for Cubic Algebraic Surfaces", *Computer Graphics & Applications*, 1989.
13. E. Sabourdy , V. Gaildrat, N. Stolte and R. Caubet, "Constructive solid geometry using implicit functions", *Advances in Computer-Aided Design*, Proc. of CADEX'96, Hagenberg, Austria, (86-92), September 9-11 1996.
14. N. Stolte, "Espaces discrets de haute résolutions : une nouvelle approche pour la modélisation et la synthèse d'images réalistes", Thèse de doctorat, Université Paul Sabatier - Toulouse III, 1996.

15. G. Wyvill, C. McPheeters and B. Wyvill, "Data structure for soft objects", *The Visual Computer*, (227-234), August 1986.

16. B. Wyvill and E. Galin, "Extending the CSG tree", *Proc. Eighth Western Computer Graphics Symposium*, university of British Columbia, (23-32), April 1997.

17. B. Wyvill and K. van Overveld, "Warping as a modelling tool for CSG/implicit models", *Shape Modelling Conference*, University of Aizu, Japan, IEEE Society Press ISBN 0-8186-7867-4, (205-2114), March 1997.

18. B. Wyvill, "Warping implicit surface for animation effects", Proc. Western Computer Graphics Symposium (SIGGRAPH 92), (55-63), 1992.

Appendix - Images