

Mixing Triangle Meshes and Implicit Surfaces in Character Animation

Antoine Leclercq[†], Samir Akkouche[‡] and Eric Galin^{††}

[†]Infogrames
82-84 rue du 1 Mars 1943
69628 Villeurbanne Cedex
aleclercq@fr.infogrames.com

[‡]L.I.G.I.M
Ecole Centrale de Lyon
B.P. 163, 69131 Ecully Cedex
samir@ec-lyon.fr

^{††}L.I.G.I.M
Université Claude Bernard Lyon 1
69622 Villeurbanne Cedex
egaline@ligim.univ-lyon1.fr

Abstract. This paper presents a technique for mixing implicit surfaces and mesh models in character modeling and animation. Implicit surfaces provide an organic aspect to standard triangle meshes and are used to add specific features. We propose a method for generating a smooth mesh from both a coarse triangle mesh and implicit primitives. The final model may be animated and displayed in real time.

Keywords: mesh models, implicit surfaces, character animation.

1 Introduction

Implicit surfaces form a very interesting model for modeling and animating organic shapes. They have been successfully used in the design of complex sea shell models [9], and for animating soft substances [4] and virtual human characters [16, 19, 12].

Let us recall that an implicit surface is mathematically defined as a set of points in space \mathbf{x} that satisfy the equation $f(\mathbf{x}) = 0$. Therefore, visualizing implicit surfaces typically consists in finding the zero-set of f , which is difficult to perform.

Polygonization algorithms provide a fast representation of the surface that may be rendered with the now democratized graphic hardware. Polygonal meshes tend to become the standard representation for surface geometry in many computer graphics applications. Although many accelerated polygonization schemes using spatial and temporal coherence [10] have been proposed, experience shows that visualising complex implicit models in real time is still difficult.

Another advantage of polygonal models is their ability to be smoothed using subdivision techniques [7, 8, 22, 3]. Subdivision algorithms proceed in two steps. The control mesh is first refined to create new vertices using a dyadic scheme, i.e. new vertices are inserted at the center of the edges. The vertices are then elevated by weighting the influence of their neighbors. The overall process is recursively applied to generate the final mesh.

Other smoothing techniques perform an iterative N-adic decomposition of the triangles [18, 17] that split triangles into n^2 sub-triangles. In contrast to subdivision techniques, those methods need not explore the neighborhood of the vertices of the mesh to compute the elevation map but rely on the normals at the control vertices only. The results are visually as convincing as subdivisions.

Tentatives to mix implicit surfaces and the polygonal model have been performed. The underlying idea is to keep the best of both worlds. Implicit surfaces may provide a smooth organic look to mesh models, accelerate collision detection and fake physically based deformations. Triangle meshes are versatile and can model any kind of shape at

different levels of detail. Moreover, triangle meshes can be rendered in real time using the graphic hardware. Existing mixed models focus on the following two objectives: accelerating and improving collisions, and skinning skeletal models.

Parent [13] has proposed to embed a meshed object into a set of implicit primitives approximating the general shape of the object to perform collision detection and propagate better deformations over the mesh. The mesh is deformed according to the modification of the potential field defined by the implicit primitives. This method relies on the implicit contact model proposed by Gascuel [11].

Shen [14] has proposed to attach skeletal implicit primitives to an animation skeleton to create virtual actors. Implicit surfaces are used to skin a polygonal skeleton. The surface is created by finding the polygonal intersection between the implicit surface and cross sections located along the skeleton. The vertices of the resulting polygonal contours are used as control points to define B-Spline patches that skin the model.

In this paper, we use implicit primitives to create and animate organic features on animated character models that are represented by a triangle mesh. The original mesh model comes from an animation modeler providing an animation skeleton and a coarse mesh. Implicit primitives are inserted only in some areas of interest where we aim at adding features. The other are left free of implicit primitives if they already provide sufficiently fine detail or out of efficiency.

Our method generates a global smoothed mesh on the fly during the animation of the character in real time. In section 2, we present an overview of the edition of the implicit surface and control mesh models. In section 3, we address the smoothing algorithm used to create the final mesh. We show how to obtain smooth blending between regions where implicit primitives deform the control mesh and ordinary mesh areas. Section 4 presents some results and timings.

2 Overview

In our animation system, the designer first creates an esquisse of a coarse mesh attached to an animation skeleton. In general, the mesh is characterized by many small triangles in the regions of interest of the shape, and may also feature large triangles in flat or slightly curved regions. For instance, video-game designers often create fine details for the head of animated character, whereas the body is characterized by a coarser mesh. The overall model is split into different regions that are in general the principal parts of the animated character.

The second step of the method aims at adding or modifying the shape of the model in some specific regions. Features are added by inserting skeletal implicit primitives wherever needed. Implicit surfaces help adding an organic look to the mesh model in some regions. For instance, muscles lend themselves to implicit modeling. Implicit surfaces are created by blending skeletal primitives. In our implementation, we use ellipsoidal primitives characterized by a center and three axes out of efficiency. The skeletal primitives are also attached to the animation skeleton so that they may be animated later. During this modeling step, the control mesh may be smoothed at each step so that the designer can modify the model interactively.

Eventually, the animation of the model only starts when editing has been finished. At each step of the animation, the position and orientation of the primitives of the animation skeleton are updated. The control mesh follows the animation skeleton accordingly as in standard animation systems. The implicit primitives attached to the skeleton also follow the animation skeleton. The control mesh is refined and smoothed on the fly during the animation as the implicit primitives constantly move which results in some

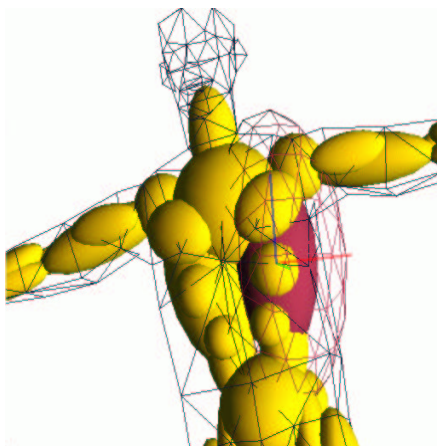


Fig. 1. Inserting implicit primitives in the control mesh model.

deformations of the surface in their region of influence.

Smoothing is performed in two steps. First, the mesh is refined using a N-adic refinement scheme. Then, the elevations of vertices are computed as follows. The new vertices of triangles outside regions of influence are moved using a N-adic smoothing algorithm [17, 18]. If a triangle fully lies inside the region of influence of f , the vertices are relocated on the implicit surface using a projection scheme that resembles the seed migration proposed by Desbrun [5]. The vertices inside triangles that straddle the regions of influence are moved using a weighted combination of both previous techniques. We address the details of the smoothing process in the next section.

3 Smoothing process

In this section, we address the computation of a smooth triangle mesh generated by the control mesh and the implicit primitives. Our algorithm proceed in two steps. First, we refine the whole control mesh using an N-adic scheme. Then, we move the vertices of the subdivided mesh to create a smooth model. The overall algorithm may be outlined as follows.

1. Refine all the triangles of mesh \mathbf{T} using the N-adic subdivision scheme.
2. For each vertex \mathbf{p} of the refined mesh, let \mathbf{T} its parent triangle in the control mesh :
 - 2.1 If \mathbf{T} lies in the region of influence of the field function f , evaluate the direction of projection, project \mathbf{p} onto the implicit surface and compute its normal as the normalized gradient of f .
 - 2.2 If \mathbf{T} lies outside the region of influence of the field function f , evaluate the new position of \mathbf{p} and its normal following a smoothing technique.
 - 2.3 Otherwise, \mathbf{T} straddles the region of influence, relocate \mathbf{p} and compute its normal by weighting the results obtained in the two previous cases.

The first step aims at creating a refined mesh. The refinement is controlled by a refinement-depth parameter that balances the level of detail and the number of generated

triangles which directly affects the overall performance. We use the same refinement depth for all the triangles of the control mesh which guarantees a consistent topology all over the refined mesh.

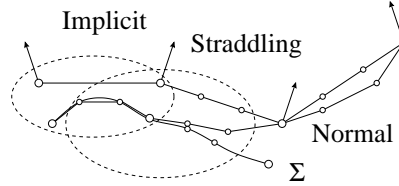


Fig. 2. Smoothing process used for implicit, straddling and normal triangles

The second step moves the vertices of the refined mesh in order to smooth the general shape of the character. As outlined in the algorithm, three cases arise, depending whether a triangle is fully inside, outside or straddling the region of influence of the implicit primitives. Those triangles will be referred to as *implicit*, *normal* and *straddling* triangles respectively. Figure 2 illustrates the smoothing process invoked for neighboring implicit, normal and straddling triangles. We address the three specific smoothing algorithms in the next paragraphs.

3.1 Smoothing implicit triangles

Let us recall that given vertices \mathbf{p} on a control mesh, we aim at moving those vertices to an implicit surface created by blending implicit primitives. Let us first detail the implicit primitives we use. In our implementation, the implicit surface is created by blending several skeletal primitives organized in a blending graph. The blending graph [21, 4] enables us to avoid unwanted blending between parts of the character model that should not melt, for instance the forearms and the torso. The final field function f is obtained by computing the sum and sometimes the maximum of field functions f_i of the primitives.

The evaluation of the field functions f_i is critical for real-time animation. Many evaluations of f_i are often needed to project a point in space onto an implicit surface. Field functions f_i are defined as $f_i = g_i \circ d_i$ where g_i denotes the potential function and d_i the distance to a skeleton. In our implementation we use spheres and ellipsoids which are very efficient to compute and produce convincing organic shapes. Spheres are the fastest to compute, whereas ellipsoids are implemented as spheres transformed by an affine transformation matrix. The ellipsoids are not only aligned to the animation skeleton axis, but may have any orientation. We use the following simple quadratic potential function, in normalized form $g(r) = (1 - r^2)^2$ which is the fastest to compute [15].

Let \mathbf{T} be a triangle inside an implicit region. The vertices of the refined triangles are to be projected onto the implicit surface. Our projection algorithm is inspired by different methods presented in the literature. Desbrun [4] has proposed to project seeds that are defined on the bounding box of each primitive along a predefined direction. The main advantage of this method is its speed, while its main drawback lies in the fact that the mesh is built separately for each primitive. The piecewise generated mesh can't be easily closed in areas where many primitives blend which results in cracks in the polygonization. Although Crespín [2] has presented a method for closing the mesh, too many triangles are required for performing this crack-fixing step, which makes this

approach too expensive in our application which should perform on the fly and in real-time.

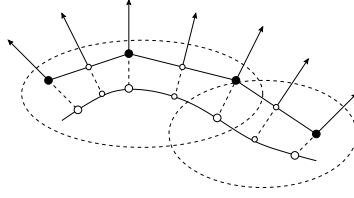


Fig. 3. Mesh smoothing for implicit triangles. The regions of influence of primitives have been dashed.

Our algorithm will take the good part of Desbrun’s method. We project the vertices of the refined mesh along predefined direction onto the implicit surface. New vertices and normal are computed as a barycentric combination of the three vertices and vertex normals of their parent triangle.

The projection of \mathbf{p} is defined as the first intersection between the implicit surface and a ray whose origin is the vertex \mathbf{p} , and whose direction follows the vertex normal if $f(\mathbf{p}) > 0$ and the opposite direction of the normal if $f(\mathbf{p}) < 0$. This projection step is crucial as it is invoked for the vertices of each implicit and straddling triangle. Implementation details are addressed in section 4.

3.2 Normal triangles

Triangles of the control mesh that do not intersect the region of influence of f also need to be smoothed. Let us recall that those triangles have been refined during the refinement step, so we need to compute a displacement map for the vertices of the refined triangles. It is interesting to point out that any smoothing algorithm may be used at this point. In our implementation, we use the N-Patch smoothing process [17]. This algorithm will be implemented in hardware in the next generation of 3D accelerators chipsets on graphic cards. Although we haven’t been able to take advantage of such hardware acceleration up to now, we expect to increase the performance of our method soon.

This smoothing algorithm deals with each triangle of the control mesh independently. Each vertex inserted in the refined triangle is elevated according to the vertices position and normal of the refined triangle.

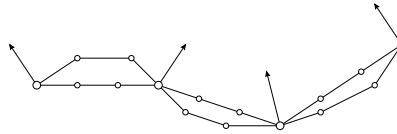


Fig. 4. Triadic smoothing using N-Patches

The topological refinement involved in this algorithm is a simple n^2 decomposition. Let us recall that we use the same refinement rule, with the same depth, for the implicit

regions and the straddling regions so as to obtain a mesh with a consistent topology all over the character.

3.3 Straddling triangles

We invoke a specific process to obtain a smooth transition between the refined implicit triangles and normal triangles. Straddling triangles are characterized automatically in a pre-processing step, and defined as triangles that share at least one edge with an implicit triangles and a normal triangles.

Straddling triangles are processed by combining the previous two methods. Let ABC a refined triangle, we assign weight to its vertices, referred to as ω_A , ω_B and ω_C , and defined as 1 if the vertex is inside the region of influence and 0 otherwise. Let (α, β, γ) the barycentric coordinates of a vertex \mathbf{p} inserted in the triangle ABC . We compute a final parameter denoted as ω that weights the influence of the implicit region and the mesh only region.

$$\omega = \alpha\omega_A + \beta\omega_B + \gamma\omega_C$$

Let $\mathbf{p}_{\text{Implicit}}$ and $\mathbf{p}_{\text{Normal}}$ denote the vertices obtained with the implicit projection and the N-Patch elevation algorithms respectively. The final vertex position is defined as follows :

$$\mathbf{p}' = \omega\mathbf{p}_{\text{Implicit}} + (1 - \omega)\mathbf{p}_{\text{Normal}}$$

The same process is applied to the normals that need to be eventually normalized.

4 Implementation details

Let us recall that we need to compute the smooth triangle mesh skinning the animated skeletal model on the fly in real time. Although the topological refinement of the coarse mesh may be performed as a pre-processing step, the computation of the elevations of the inserted vertices is to be performed at each time step. Normal triangles are smoothed into N-Patches in real time [17]. Straddling and implicit triangles require more computations as the inserted vertices need to be projected onto the implicit surface.

Projecting a vertex \mathbf{p} onto the implicit surface can be thought of as computing the first intersection between a ray with origin \mathbf{p} and a given normal. We have implemented and compared two methods to project vertices on the implicit surface. The first one uses an analytic ray intersection scheme presented derived from [20] whereas the second one uses a binary search inside an interval.

4.1 Analytic scheme

Wyvill [20] first proposed analytic techniques for ray tracing blobs created with sphere and ellipsoid primitives. Let $\mathbf{x}(t) = \mathbf{p} + t\mathbf{\bar{n}}$ the parametric equation of the ray where $\mathbf{\bar{n}}$ refers to the mesh normal at vertex \mathbf{p} . The squared Euclidean distance of a point along the ray to the center of the sphere may be written as a second degree polynomial of variable t . As we use quartic potential functions, the field function along the ray $f(t)$ is a piecewise quartic polynomial whose roots may be computed analytically. The general algorithm may be written as follows :

1. For each primitive i , compute its field contribution $f_i(t)$ along the ray over the interval $[t_i^-, t_i^+]$ if the ray intersects the region of influence of primitive i , otherwise set $f_i(t)$ to null.

2. Compute the overall field function along the ray $f(t)$, which is a piecewise polynomial in t formed after substituting the ray equation into the polynomial equations of the different regions of space the ray traverses.
3. Search the roots of the polynomial equation $f(t) = T$ over the sub-intervals, where T is fixed threshold.

In practice, this method requires the quite expensive computation of the closed form expression of $f(t)$ before any root finding process may be invoked.

4.2 Ray sampling

In our case, we can easily find an interval where the ray does intersect the implicit surface. The direction of the ray no longer follows the vertex normal $\hat{\mathbf{n}}$. Instead, we use a precomputed direction. For each vertex \mathbf{p} of the refined mesh, we can compute a corresponding anchor point denoted as \mathbf{q} on the animation skeleton. The anchor points are computed only once during initialisation. They are defined as points on the animation skeleton the closest to the mesh vertices we want to project. The anchor point is animated using the animation skeleton so that it should always lie on the animation skeleton. Since the body of the character is built around the animation skeleton, the potential of the anchor point is always positive.

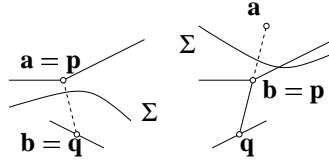


Fig. 5. Computation of the line segment intersecting the implicit surface

The algorithm proceeds in two steps. First, we define a line segment $[\mathbf{ab}]$ intersecting the implicit surface, with $f(\mathbf{a}) < 0$ and $f(\mathbf{b}) > 0$ respectively. Two cases appear. If the mesh vertex \mathbf{p} is outside the implicit surface, then set \mathbf{a} to \mathbf{p} and \mathbf{b} to its corresponding anchor point \mathbf{q} . Otherwise, set \mathbf{b} to \mathbf{p} and locate \mathbf{a} as shown in Figure 5.

The second step performs a simple iterative binary search over the line segment $[\mathbf{ab}]$. Experiments show that less than eight steps are necessary to approximate the intersection. In order to speed up the computation of $f(\mathbf{x})$ during the binary search, we first cull primitives that do not intersect $[\mathbf{ab}]$.

4.3 Back face culling

In general, back-facing triangles need not be displayed. Therefore, another improvement consists in eliminating back facing triangles before performing the projection of the vertices of the refined triangle mesh. A classic backface culling algorithm is not satisfying though, as the triangles of the control mesh are refined and deformed during the smoothing process. Thus, refined triangles may appear at the silhouette edges even if their parent triangle were facing backwards.

In our implementation, we only eliminate triangles whose three vertex normals are pointing backwards. As the vertex normals represent the way a triangle deforms, this criterion enables us to preserve triangles that may modify silhouette edges. This results in significant accelerations as show in Table 1.

5 Results

The refinement and smoothing algorithms have been implemented and linked with the animation system developed at Infogrames. The following timings have been performed on a Pentium III-733 with a GeForce-1 graphic card.

Technique	Analytic ray tracing		Binary search	
	Normal	Face culling	Normal	Face culling
2^2	104	123	141	165
3^2	54	67	76	92
4^2	33	43	47	59
5^2	22	30	31	40
6^2	15	21	22	29
7^2	11	16	17	22

Table 1. Frame rates obtained using analytic ray tracing and binary search

Table 1 compares the frame rates obtained using analytic ray tracing and binary search. In both cases, we have reported the frame rate with and without back face culling. Timings show that the binary search technique performs significantly faster than the analytic ray tracing method. One reason for this is that we take advantage of the known location of primitives around the animation skeleton. Therefore, the interval where the ray surface intersection is searched is very small, and only a few iterations are needed to converge to the surface.

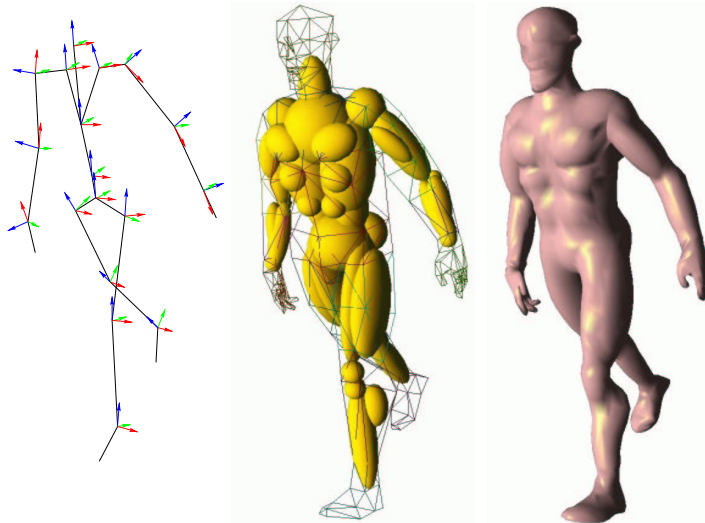


Fig. 6. Walking humanoid model, from left to right, animation skeleton, mesh model and implicit primitives, final smooth mesh model

Refinement	2 ² subdivision		3 ² subdivision		4 ² subdivision		5 ² subdivision	
	<i>N</i>	Fps	<i>N</i>	Fps	<i>N</i>	Fps	<i>N</i>	Fps
Humanoid	2368	165	5328	92	9472	59	14800	40
Horse	1648	197	3708	133	6592	86	10300	60

Table 2. Frame rate (labeled as Fps) and number of refined triangles (denoted as *N*) for the humanoid and horse models at increasing subdivision depths

The back face culling pre-processing step also speeds up the overall process. As one could expect, the higher the subdivision depth, the greater the accelerations.

Table 2 reports the number of frames per second obtained with the humanoid and the horse models. The control meshes involve 592 and 412 triangles respectively. 41 implicit primitives have been added to the humanoid model that are mostly located on the torso, whereas the horse model only involves 24 evenly located over the body. Without implicit primitives, both models may be rendered at 212 frames per second.

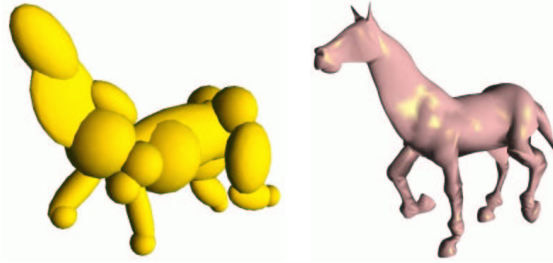


Fig. 7. Unblended implicit primitives and resulting horse model

Timings show that adding implicit primitives considerably reduces the overall rendering speed. Experiments show that using a subdivision level greater than 5 does not provide any significant visual improvement. The frame rate obtained at this depth is still real time. In general, the subdivision depth adapts to the size of the overall model on the screen so as to provide sufficient level of detail. Unless the character model fills the screen, the subdivision depth is in general set to 2 or 3 which proves to be accurate enough.

Although mapping textures onto general implicit surfaces remains a challenging problem, our method handles texture mapping in a natural way. In our technique, we rely on the coarse mesh to attach the texture map to the model. The uv-coordinates of the vertices of the refined triangles in the texture map are simply computed using their barycentric coordinates in the parent triangle (see section 3.3). Experiments performed with a GeForce-1 graphic card show that the frame rate is almost insensitive to the use of textures.

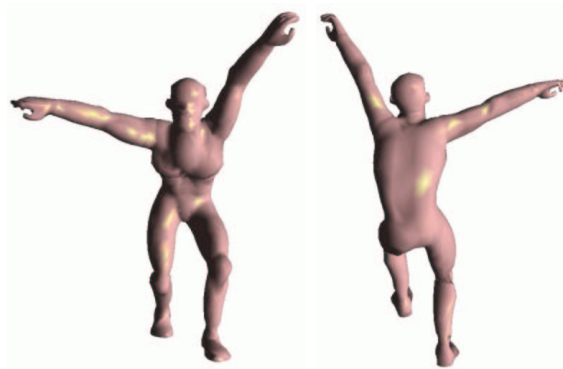


Fig. 8. Dancing character

6 Conclusion and future work

Triangle meshes provide a fast and versatile geometric representation of surfaces that can be rendered in real time with existing hardware. Implicit surfaces lend themselves for the creation of smooth organic shapes. In this paper, we have combined the best of both worlds. We have show that implicit primitives can be inserted in the character creation process to add organic features to mesh models. Our shape editor has been tested by designers at Infogrames who easily created the humanoid and the horse models.

We did not take advantage of temporal and spatial coherence to speed up computations. For instance, the projection of the vertices for implicit and straddling triangles could be improved as the previous projection point could be used to define a smaller research interval.

In the future, we plan to use more complex implicit primitives such as cone-spheres, cylinders or spline skeletal elements. Moreover, the parameters of the implicit primitives could be modified during the animation to create more realistic deformations.

References

1. J. Bloomenthal, C. Bajaj, J. Blinn, M. P. Cani-Gascuel, A. Rockwood, B. Wyvill and G. Wyvill. *Introduction to Implicit Surfaces*, Morgan Kaufmann 1997.
2. B. Crespin, P. Guitton, C. Schlick. Efficient and accurate tessellation of implicit sweeps. *Proceedings of CSG'98*, 1998.
3. T. DeRose, M. Kass, and T. Truong. Subdivision Surfaces in Character Animation. *Computer Graphics (Siggraph 1998 Proceedings)*, 85-94, July 1998.
4. M. Desbrun and M. P. Gascuel. Animating soft substances with implicit surfaces. *Computer Graphics (Siggraph 1995 Proceedings)*, 287-290, August 1995.
5. M. Desbrun, N. Tsingos and M. P. Gascuel. Adaptive Sampling of Implicit Surfaces for Interactive Modeling and Animation. *Computer Graphics Forum*, **15**(5): 319-325, December 1996.
6. M. Desbrun and M. P. Cani. Active Implicit Surface for Animation. *Graphics Interface'98*, 143-150, June 1998.
7. D. Doo and M. Sabin. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. *Computer Aided Design*, 157-175, 1978.

8. D.Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, **10**(6) : 356-360, 1978.
9. C. Galbraith, P. Prusinkiewicz and B. Wyvill. Modeling Murex Cabriti Seashell with a Structured Implicit Surface Modeler. *Proceedings of Computer Graphics International 2000*, 2000.
10. E. Galin and S. Akkouche. Incremental Polygonization of Implicit Surfaces. *Graphic Models and Image Processing*, **62** : 19-39, 2000.
11. M.P. Gascuel. An Implicit Formulation for Precise Contact Modeling. *Computer Graphics (Siggraph 1993 Proceedings)*, 313-320, August 1993.
12. P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel, D. Thalmann. Real-time Animation of Realistic Virtual Humans. *IEEE Computer Graphics and Applications*, **18**(5) : 42-55, 1998.
13. R. Parent. Implicit function based deformations of polyhedral objects. *Proceedings of Implicit Surfaces '95*, 113-128, April 1995.
14. J. Shen and D. Thalmann. Interactive shape design using metaballs and splines. *Proceedings of Implicit Surfaces '95*, 187-196, April 1995.
15. A. Sherstyuk. Kernel functions in convolution surfaces : a comparative analysis. *The Visual Computer*, **15**(4) : 171-182, 1999.
16. D. Thalmann, J. Shen, E. Chauvineau. Fast Realistic Human Body Deformation for Animation and VR Applications. *newblockComputer Graphics International*, Korea, 166-174, June 1996.
17. A. Vlachos, J. Peters, C. Boyd and J.L. Mitchell. Curved PN Triangles. *ACM Symposium on Interactive 3D Graphics*, 159-166, March 2001.
18. P. Volino and N. Magenat Thalmann. The Spherigon : a simple polygon patch for smoothing quickly your polygonal meshes. *Computer Animation 98 Proceedings*, 72-79, 1998.
19. J. Wilhelms and A. Van Gelder. Anatomically Based Modeling. *Computer Graphics (Siggraph 1997 Proceedings)*, 173-180, 1997.
20. G. Wyvill and A. Trotman. Ray Tracing Soft Objects. *Computer Graphics International '90*, 469-476, 1990.
21. A. Guy and B. Wyvill. Controlled blending for implicit surfaces using a graph. *Proceedings of Implicit Surfaces '95*, 107-112, 1995.
22. D. Zorin, P. Schröder and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. *Computer Graphics (Siggraph 1996 Proceedings)*, 189-192, 1996.