Computer Graphics Techniques for Realistic Modeling, Rendering and Animation of Water. Part II: 1989-1997

A. Iglesias[*]

Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, s/n, E-39005, Santander, Spain
iglesias@unican.es
http://personales.unican.es/iglesias

Abstract. This work concerns the realistic modeling, rendering and animation of water. Specifically, the paper surveys many of the most outstanding references on computer graphics techniques regarding this problem developed during the period 1989-97. The paper, which has been organized for clarity into two different periods, follows on from a previous one dedicated to the same topic during the period 1980-88 [11].

1 Early nineties: 1989-92

Although the previous works [11] provided adequate models for waves hitting the beach, to quote just one example, a significant range of physical phenomena still remained unexplored. A major question was the accurate description of fluid dynamics. Great interest was also placed upon several physical phenomena related to water such as:

- state of matter changes such as melting and freezing,
- complex natural phenomena such as wetting and drying,
- mass transport or flow including meandering,
- the behavior of individual droplets and their streams,
- interaction with static and dynamic buoyant obstacles, etc.

On the other hand, a number of interesting rendering effects, such as the simulation of reflected waves, the interaction between light and water, the analysis of caustics, etc. had not yet been considered by the previous computer graphics techniques. Finally, water realistic animation was still in its infancy and it had to be greatly improved. To overcome these limitations, several new models were proposed. Roughly, they can be grouped into two different alternatives which are analyzed in the following sections.

[*] This research was supported by the CICYT of the Spanish Ministry of Education (projects TAP1998-0640 and DPI2001-1288).

1.1 Interaction of a Large Number of Particles

The first alternative tried to simulate the fluid dynamics by the interaction of a large number of particles [7, 17, 26, 27]. Thus, in [17] and [27] the authors studied the attraction and repulsion forces between particles to simulate various degrees of fluid viscosity and state of matter change such as melting. For example, the approach in [17] presented a particle based model for fluids, powders and gelatinous solids. This model was based on what the authors called *globules*, a term intended to avoid connotations with words such as particle or blob, and used to designate the elements of the connected particle system. Globules can be applied for the detection of soft collisions between the particles and obstacles, involving forces which vary gradually with distance, thus allowing globules to flow over one another. The forces acting on the particles depend on two scaling factors, one for attraction/repulsion, s_r, and another one, s_d, (drag) to attenuate the inter-globule force based on distance. For powders and fluids the globules interact with a short range repulsive force and drag. In the solid state the attraction term is also considered, so globules interact with a short range repulsion, a medium range attraction and long range indifference. The temperature values for a given pair of globules can be used to simulate changes in their interaction behavior, thus allowing the melting of solids and the freezing of liquids. This temperature effect was later analyzed by Tonnesen in [27], through a model describing the changes in geometry and movement of elemental volumes as a consequence of thermal energy and external forces. This model simulates both the liquid and solid states by varying the shape of the potential energy curve as a function of temperature. Objects at hot temperatures behave like fluids, with rapidly varying geometry. At cold temperatures they resemble solids, with a stable shape which can be modified, however, under the influence of appropriate external forces. Changes in topology are modelled by using particles to represent elemental volumes with potential energies between pairs of particles. Finally, another remarkable reference using this particle interaction scheme is due to Sims [26]. In his work, a waterfall was simulated by applying gravity to thousands of non-interacting water droplet particles and bouncing them off obstacles made of planes and spheres. The water droplets were created at the top of the waterfall, flowed over the last edge at the bottom and then recycled back to the top of the waterfall. These particles of different shapes, sizes, colors and transparencies were rendered by using a parallel rendering particle system incorporating some techniques for increasing the image quality, such as anti-aliasing, hidden surfaces and motion-blur. The combination of some *ad hoc* tricks (such as the combination of white and blue particles to avoid lighting calculations and exaggerated motion blur to give the flow a smoother look) allow realistic pictures of the waterfall to be obtained. Additional effects such as the deformations produced on a falling droplet as it enters in contact with a surface are also analyzed in [20].

1.2 Partial Differential Equations for the Fluid Dynamics

The second approach is to directly solve a partial differential equation (PDE) system describing the fluid dynamics [14]. Of course, this alternative is much more reliable and realistic in terms of physical simulation. The main problem is that a truly accurate simulation of fluid dynamics requires the computation of the fluid motion throughout a volume. This implies that the computation time for each iteration is at least proportional to the cube of the resolution, making the computation prohibitively expensive. Fortunately, for rendering and animation purposes, to obtain very accurate results is less important than other factors, such as the speed of the simulation and the stability of the numerical methods involved within. With this idea in mind, Kass and Miller [14] considered a very simplified subset of water flow where:

– the water surface can be represented by a height field,
– the vertical component of the velocity of the water particles can be ignored (so motion is uniform through a vertical column) and
– the horizontal component of the velocity of the water in a vertical column is approximately constant.

With these simplifications, the motion equations of the water were approximated in terms of a grid of points on the height field, obtaining a wave equation in which the wave velocity is proportional to the square root of the depth of the water. Thus, this method can generate wave refraction with depth and take into account other features, such as wave reflections, net transport of water and situations in which the boundary conditions change through time altering the topology of the water. Due the height-field representation of the water surface, the two-dimensional motion equations were numerically integrated by using a finite-difference technique. This technique converts the PDE into ODEs, which can be integrated through a first-order implicit method. The algorithm is very easy to implement and gives rise to tridiagonal linear systems. For the three-dimensional case, one alternating-direction method was applied, allowing the 3D iteration to be divided into two 2D sub-iterations, so complexity does not radically increase. Additional simplifications were added for rendering, making high-resolution simulations possible. The water was rendered using *caustic shading* that simulates the refraction of illuminating rays at the water surface. The work also incorporated realistic appearance for water in sand through a *wetness map* that computes the wetting and drying of sand as the water passes over it. However, this wetness map had to be filtered to avoid aliasing artifacts in the boundary between wet and dry areas before using it for shading.

Another interesting reference using caustics for water rendering is that in [29]. This work, cited here for its applications to realistic water rendering, presented a new method (the so-called *backward beam tracing*) for computing specular to diffuse transfer of light. This problem appears when light reflecting from, or refracting through, one surface (the specular surface) hits a diffuse surface where, by definition, it is emitted equally in all directions. Early 90s' traditional global illumination techniques [9], such as those based on ray tracing and radiosity, did not handle it. This fact was not so surprising when using ray tracing, best suited to solving the diffuse and specular to specular mechanisms of light transport. After all, the specular to diffuse transfer is view independent and hence, more suitable for radiosity. But even radiosity techniques generally ignored this problem. Remarkable exceptions were given by the *hybrid techniques* [25, 28], where a combination of ray tracing and radiosity is considered. The hybrid method uses two passes for illumination calculation: a view independent pass, similar to diffuse radiosity, computing the diffuse illumination of a scene, and a view dependent pass, similar to distributed ray tracing, that uses the previous one to provide global illumination information and subsequently the specular components. Unfortunately, these methods only captured the softest of specular and diffuse effects varying slowly over pixels, and they became impractical for capturing the (more interesting) higher frequency effects.

The approach in [29] is also a two-pass algorithm, the first pass being a variation of the backward ray tracing [1] called backward beam tracing, then combined with a rendering phase (the second pass) incorporating a test for caustic polygons associated with diffuse polygons. The method was applied to simulate the light-water interaction where the water surface acts as the specular surface. Its performance was illustrated through some frames from an animated underwater caustic sequence, showing the familiar sinuous shifting patterns of light on objects underwater as they intersect the caustic. As these patterns are driven directly by the water surface (described by a polygonal mesh of triangles displaced by a height field), animating them is merely a question of animating the water surface. Therefore, this animation process consists of deciding upon an appropriate time interval and frequency/amplitude rates. On the other hand, if the triangles in the polygonal mesh are small enough, the beams can be represented as polygonal illumination volumes, rendered using a modified version of a light volume rendering technique proposed by Nishita et al. [21]. The results show that this technique is particularly effective to describe the light-water interaction.

2 1993-97

During these years, the two approaches introduced in the previous years (namely, the interaction of a large number of particles and the system of PDEs to describe the fluid dynamics) were extended and improved. In addition, several interesting effects for substantially improving both rendering and animation were described.

2.1 Improving Particle System Methods

The early water models developed for computer graphics and based on particle systems (see [11], Sec. 3) assumed that particles of water moved in circular and elliptical orbits around their initial positions. Such an assumption was intended to render large bodies of water such as the ocean. A very different situation is that of animating the flow of very small amounts of water, in which this particle motion model is no longer able to realistically reproduce the fluid dynamics.

Some interesting references to overcome this problem appeared in this period. For instance, in [16] the use of string textures for rendering large waterfalls was proposed. Particle systems usually require a large number of particles, thus leading to unreasonably large amounts of memory usage. This is because the position and velocity are stored explicitly for each particle. On the contrary, implicit methods are used to store parameters in solid texture, by means of addressing small lookup tables. Mallinder suggested a new modelling method which he called *string texture*, which is basically a development from particle systems and solid texture, using a method of implicitly storing particles. From this point of view, string texture is a technique to save memory for storing information on particles.

A strictly geometrical approach to calculate the rolling of a rigid convex object on a smooth biparametric surface is reported by Hégron in [10]. This author assumed that there is a single contact point (determined through a simple iterative algorithm) between the object and the surface which is maintained during the animation. Then, the motion was computed in terms of discrete displacements on the tangent plane at the contact point and projections onto the surface. Since this projection is time-consuming, an alternative prediction-correction scheme (based on projecting the motion onto the tangential plane and then making corrections to ensure that the points are on the surface) was proposed instead. Although the proposal does not take into account the physics of the object and it was not intended specifically for water animation purposes, it has a potential application to the problem of modeling the water droplets on a curved surface.

Other relevant references for rendering the flow of small amounts of water are described in [4, 15]. They included a physically based model for simulating icicle growth realistically [15] and the rendering of water currents using particle sytems [4]. Unfortunately, since these authors did not consider the interfacial dynamics, it is difficult to animate the streams of water droplets.

Another interesting alternative was given by Kaneda et al. in [12]. These authors proposed a method for animating water droplets and their streams on a glass plate taking into account the dynamics between fluid and solid. In this scheme, the shape and the motion of the water droplets depend on the gravity force, the surface tensions of the glass plate and the water and the interfacial tension between them.

To simulate the stream the surface model of the glass plate was discretized so that the water droplets travel from one mesh point to the next according to a set of rules describing their dynamics as a function of their mass, the angle of the plate's inclination, φ, and affinity. A droplet placed on a lattice point (i,j) moves to one of the three lower lattice points $(i-1,j-1)$, $(i-1,j)$ or $(i-1,j+1)$. Because of the nature of the wetting phenomena, some amount of water remains behind the route of the stream so the mass of the droplet decreases with time and finally the flow stops. The model also includes equations for the speed of the running water droplet as a function of the wetness and the angle φ and for the speed and mass of the new droplet resulting from merging of two original droplets. The final algorithm returns the positions and masses of all water droplets for every frame of the animation.

To render the water droplets and the streams the authors proposed a high-speed rendering method which takes into account the reflection and the refraction of the light. Basically, it is an extension of the environment map by Greene [8] in which objects in the scene are projected onto the planes of a cuboid, whose center is on the glass plate. The method consists of calculating background textures by projecting objects in the scene onto the faces of a cuboid whose center is on the glass plate, then calculating the directions of rays reflected or refracted by water droplets on a glass plate, and finally determining pixel colors by using the background textures and the intersection of the ray and the cuboid. The method finally generates an image with these pixel colors at these intersection points, thus avoiding calculating the intersections between the ray and the objects (undoubtely, the most time-comsuming process in ray tracing).

This model was firstly applied to animate water droplets meandering on an inclined glass plate [12] and later extended to water droplets on curved surfaces [13]. In this last case, the surface was described by Bézier patches and then converted to a discrete surface model for which each quadrilateral mesh was approximated by a plane. In [13] the applications of water droplet animation were classified into two categories: those mainly pursuing rendering speed (such as in drive simulators in which interactivity is the goal) and those pursuing photo-reality. According to this classification, two different rendering algorithms were proposed. The first one (for which the stream is modeled as a group of spheres) is essentially an extension of the simple and fast rendering method described in [12] to handle Bézier patches. The second one is a more sophisticated high-quality rendering method intended for photo-reality and based on metaballs.

The metaball technique was first introduced by Blinn in [2] and he called it *blobs*. Subsequently, this technique was improved by several authors [19, 30–34] who coined the terms *metaballs* and *soft objects*. This technique is widely used to represent “soft objects” like liquids [32]. Its popularity comes from the fact that the model is defined by just a few simple constraints providing free-form deformations. In addition, it is used for water modeling because the water droplets merge smoothly together in this model.

In [13] the metaball is defined by its center, its density distribution (the authors applied that of [34]) and a threshold value defining the surface. To overcome the conflict of working with both implicit surfaces (given by the metaball technique) and parametric surfaces (the Bézier surfaces of the problem), a combination of two rendering methods, namely the one in [22] for Bézier surfaces and the one in [23] for metaballs, is proposed. In this new scheme the density distribution along a ray was converted into a Bézier function and the intersection between the ray and the metaball is calculated using a Bézier clipping [22].

2.2 Improving PDE Fluid Dynamics Methods

One of the first attempts to simulate faithfully the behavior of the fluid dynamics was the work of Kass and Miller [14] already analyzed in Sec. 1.2. This model is limited in the sense that it does not address the full range of three-dimensional motion (including rotational and pressure effects) found in a liquid. In addition, since the velocity of the fluid is known only on the surface and internal pressure is not calculated at all, the model cannot easily incorporate dynamic or buoyant objects.

A more realistic model can be obtained by considering the Navier-Stokes (NS) equations, notably the most comprehensive of all fluid models. Basically, these equations describe completely the motion of a fluid at any point within a flow at any instant of time. Because of their ability to capture the turbulent or stable behavior of the fluid with arbitrary viscosity in three dimensions, they are often used to simulate accurately fluid phenomena.

The Navier-Stokes equations have been applied to create models of water motion for computer graphics [3, 5]. In [3] a simplified version of the Navier-Stokes equations in two dimensions was considered. Such a simplification was obtained by removing the vertical dependence and solving the resulting two-dimensional system. In other words, they assumed that the fluid has zero depth, thus treating it as being completely flat during the computation. Although still able to model some kind of interactions between moving objects and the flow, such an assumption considerably restricts the range of phenomena of the model. For example, since the obstacles must be two-dimensional, the model does not deal with submerged objects. In addition, neither convective wave effects nor mass transport can be incorporated into this model.

Another comprehensive methodology based on the NS equations for animating liquid phenomena was introduced in [5] and later extended in [6]. These authors noticed that the real fluid dynamics can only be captured by solving the three-dimensional Navier-Stokes equations. To this end, they considered a two-stage calculation over an environment of static obstacles surrounded by fluid. Firstly, a finite difference approximation to the NS equations was applied to a low-resolution discretized representation of the scene. This scheme was coupled with an iterative step to refine the velocities and determine the pressure field, which was combined with the Lagrange equations of motion to simulate dynamic buoyant objects. At this stage, the fluid position in 2D was tracked by convecting massless marker particles with local fluid velocity. These marker particles are very useful to highlight many internal fluid motion effects, such as rotation and splashing, as well as for animating violent phenomena such as overtunning waves. Additional effects such as vorticity were also taken into account in this step. In the original work, a secondary calculation was performed to accurately determine the liquid surface position through a height field equation. Since this equation describes surfaces as single-valued functions, it can be effectively applied to situations such as puddles, rivers or oceans. For dramatic effects such as crashing waves or splashing, a combination of height field and marker particles was applied. The performance of the method was illustrated through some realistic 2D and 3D examples, such as the animation titled *Moonlight Cove*, a $50 \times 15 \times 40$ mesh used to show the effect of two large ocean waves crashing into a shallow cove. Scene complexity was increasing by including submerged rocks and irregular sea bottom, thus leading to interesting features on the water surface.

The dynamics behavior of splashing fluids was also analyzed in [24]. Essentially, the authors introduced a model that simulates the behavior of a fluid when objects impact or float on its surface. To this purpose, they used a three-part system where each subsystem corresponds to a physical area of the fluid body: the volume, the surface and the spray (see Figure 1).

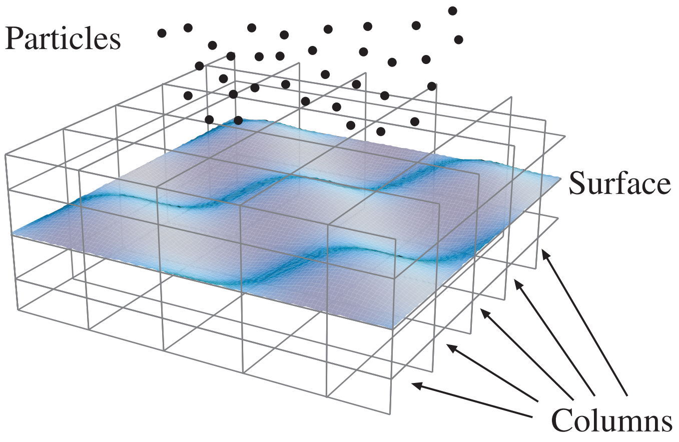

Fig. 1. Scheme of the fluid model in reference [24]. The model is a three-part system containing subsystems for volume (columns), surface and spray (particles)

The water volume that makes up the main body of the fluid was simulated by a collection of vertical tubes or columns connected by virtual pipes, thus allowing here to be a flow between these columns according to the hydrostatics laws for pressure. In addition, flow conditions must be specified to model boundary conditions, such as barriers (flow set to zero) or fluid sources or sinks (flow set to a positive or negative constant). The surface subsystem allows external objects to interact with the fluid system and consists of mesh of control points whose vertical positions are found by averaging the heights of the surrounding columns. Finally, the spray subsystem is a particle system to model water droplets disconnected from the main body of the fluid. These droplets fall under the influence of the gravity and their dynamics is quite simple since the particles are noninteracting. To preserve the total volume of the system, the volume of each particle was substracted from the column from which it was created. Although the model cannot deal with submerged objects, the impact problem can be treated in a natural way. The impact of the object on the surface is transferred to the pressure of the affected columns. When thresholding the vertical velocity of the column, spray is generated yielding a particle system until it is reabsorbed by the fluid, encounters a ground plane or strikes some other object. The model is able to describe a wide range of phenomena such as waves, impacts, splashes or floating objects. On the contrary, those phenomena related to vertical effects such as turbulence or underwater effects, are not adequately described in this model.

A remarkable extension of this proposal was described in [18], including an explicit underwater terrain and the removal of the vertical isotropy, thus making the interaction between particles and fluid possible. The model consists of a volume system (containing the information about the main body of fluid) coupled with a particle system. Although some effects such as friction do not follow the actual physical laws and additional parameters with no physical meaning have been introduced to simulate spreading, the model is overall based on the physics of fluids. The model simulates accurately many different situations: the propagation of the surface waves, the variation of the propagation of the speed with terrain, the variation of wave patterns with the depth of water, the dynamics of the water droplets, the floating and submerged objects, splashing, etc. Some shortcomings of the model are that a height field for the water surface was considered (clearly an unrealistic situation in the real world) and that the force from an object striking the surface is strictly applied in the vertical direction, even although the direction of the object has some angle with the vertical. Extensions of these models to overcome these limitations were subsequently described. They will be analyzed in a future work and reported elsewhere.

References

1. Arvo, J.: Backward ray tracing. Developments in ray tracing. SIGGRAPH'86 course notes, Vol. 12 (1986)
2. Blinn, J.F.: Generalization of algebraic surfaces drawing. ACM Transactions on Graphics, Vol. **2**(3) (1980) 235-256
3. Chen, J.X., Lobo, N.V.: Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. Graphical Models and Image Processing **57**(2) (1995) 107-116
4. Chiba, N., Sanakanishi, S., Yokoyama, K., Ootawara, I., Maruoka, K., Saito, N.: Visual simulation of water currents using a particle-based behavioural model. Journal of Visualization and Computer Animation, Vol. **6**(3) (1995) 155-171
5. Foster, N., Metaxas, D.: Realistic animation of liquids. Proc. of Graphics Interface'96, Calgary, Canada (1996) 204-212; also in: Graphical Models and Image Processing **58**(5) (1996) 471-483
6. Foster, N., Metaxas, D.: Controlling fluid animation. Proc. of Computer Graphics International CGI'97, IEEE Computer Society Press, Menlo Park, CA (1997) 178-188
7. Goss, M.E.: A real-time particle system for display of ship wakes. IEEE Computer Graphics and Applications **10**(3) (1990) 30-35
8. Greene, N.: Environment mapping and other applications of world projections. IEEE Computer Graphics and Applications **6**(11) (1986) 21-29
9. Hall, R.: Illumination and Color in Computer Generated Imagery. Springer-Verlag (Series: Monographs in Visual Communication) New York (1989)
10. Hégron, G.: Rolling on a smooth biparametric surface. The Journal of Visualization and Computer Animation **4** (1993) 25-32
11. Iglesias, A.: Computer graphics techniques for realistic modeling, rendering and animation of Water. Part I: 1980-88. (this volume)

12. Kaneda, K., Kagawa, T., Yamashita, H.: Animation of water droplets on a glass plate. Proc. of Computer Animation'93 (1993) 177-189
13. Kaneda, K., Zuyama, Y., Yamashita, H., Nishita, T.: Animation of water droplets on curved surfaces. Proc. of Pacific Graphics'96, IEEE Computer Society Press, Los Alamitos, Calif. (1996) 50-65
14. Kass, M., Miller, G.: Rapid, stable fluid dynamics for computer graphics. Proc. of SIGGRAPH'90. Computer Graphics **24**(4) (1990) 49-57
15. Kharitonsky, D., Gonczarowski, J.: A physical based model for Icicle growth. The Visual Computer, Vol. **10**(2) (1993) 88-100
16. Mallinder, H.: The modeling of large waterfalls using string texture. Journal of Visualization and Computer Animation, Vol. **6**(1) (1995) 3-10
17. Miller, G., Pearce, A.: Globular dynamics: a connected particle system for animating viscous fluids. Computers and Graphics **13**(3) (1989) 305-309
18. Mould, D., Yang, Y.-H.: Modeling water for computer graphics. Computers and Graphics **21**(6) (1997) 801-814
19. Murakami, S., Ichihara, H.: On a 3d display method by metaball technique. Journal of the Electronic Communications **70**(8) (1987) 1607-1615
20. Nishikawa, N., Abe, T.: Artificial nature in splash of droplets. Compugraphics'91 **1** (1991) 457-466
21. Nishita, T., Miyawaki, Y., Nakamae, E.: A shading model for atmosferic scattering considering luminous intensity distribution of light sources. Proc. of SIGGRAPH'87. Computer Graphics **21**(4) (1987) 303-310
22. Nishita, T., Sederberg, T.W., Kakimoto, M.: Ray tracing rational trimmed surface patches. Proc. of SIGGRAPH'90. Computer Graphics **24**(4) (1990) 337-345
23. Nishita, T., Nakamae, E.: A method for displaying metaballs by using Bézier clipping. Proc. of EUROGRAPHICS'94. Computer Graphics Forum **13**(3) (1994) 271-280
24. O'Brien, J.F., Hodgins, J.K.: Dynamic simulation of splashing fluids. Proc. of Computer Animation'95 (1995) 198-205
25. Sillion, F., Puech, C.: A general two-pass method integrating specular and diffuse reflection. Proc. of SIGGRAPH'87. Computer Graphics **23**(3) (1989) 335-344
26. Sims, K.: Particle animation and rendering using data parallel computation. Proc. of SIGGRAPH'90. Computer Graphics **24**(4) (1990) 405-413
27. Tonnesen, D.: Modeling liquids and solids using thermal particles. Proc. of Graphics Interface'91 (1991) 255-262
28. Wallace, J.R., Cohen, M.F., Greenberg, D.P.: A two-pass solution to the rendering equation: a synthesis of ray tracing and radiosity methods . Proc. of SIGGRAPH'87. Computer Graphics **21**(4) (1987) 311-320
29. Watt, M.: Light-water interaction using backward beam tracing. Proc. of SIGGRAPH'90. Computer Graphics **24**(4) (1990) 377-385
30. Wyvill, B., McPheeters, C., Wyvill, G.: Data structure for soft objects. The Visual Computer **2**(4) (1986) 227-234
31. Wyvill, B., McPheeters, C., Wyvill, G.: Animating soft objects. The Visual Computer **2**(4) (1986) 235-242
32. Wyvill, B.: Soft. Proc. of SIGGRAPH '86, Electronic Theater and Video Review **24** (1986)
33. Wyvill, G., Wyvill, B., McPheeters, C.: Solid texturing of soft objects. IEEE Computer Graphics and Applications, December (1987) 20-26
34. Wyvill, G., Trotman, A.: Ray-tracing soft objects. Proc. of Computer Graphics International CGI'90, Springer Verlag (1990) 469-476