Comput. & Graphics Vol. 13, No. 3, pp. 305–309, 1989
Printed in Great Britain.

0097-8493/89 $3.00 + .00

GLOBULAR DYNAMICS: A CONNECTED PARTICLE SYSTEM FOR ANIMATING VISCOUS FLUIDS

GAVIN MILLER
Apple Computer, Inc., The Advanced Technology Group, 20525 Mariani Ave., M5:22Y, Cupertino, CA 95014

and

ANDREW PEARCE
Alias Research, Inc., 500-110 Richmond St., E., Toronto, Ontario, Canada M5C 1P1

Abstract—Connected particle systems can depict many objects difficult to model in any other fashion. We present a method for animating viscous fluids by simulating the forces of such particles interacting with each other. This method allows for collision detection between the particles and obstacles, both stationary and mobile, and it allows solid objects to break and melt. An approximate method for covering the particles with an isosurface for efficient rendering is also presented.

1. INTRODUCTION

We are interested in modeling materials such as lava, mud, slime, oil and salad dressing, as well as breakable, meltable solids. Existing techniques which could be considered include bicubic patches[1] and blobby molecules[2]. However, problems arise with both methods. It is difficult to model amorphous shapes with bicubic patches since they cannot be broken into arbitrary pieces. Blobby molecules do model amorphous shapes but two problems arise:

1. Blobs need a dynamic model for animation which includes collision detection and blob to blob interaction.
2. We wish to render many blobs with limited memory, using existing fast rendering hardware to produce animations quickly and inexpensively.

Particle systems[3] provide a dynamic model for the trajectories and collisions but do not take into account particle interaction. Soft objects[4] apply some motion control to blobs but ignore collisions between blobs. Recently, elastically deformable models[5] were proposed which provide a technique for computing the shape changes of continuous materials when subjected to external forces; however, the bonds between adjacent pieces of material are of a fixed topology.

The surfaces we are interested in form and break bonds over time. We introduce a connected particle system which uses a dynamic model for particle to particle interaction and interaction with other environmental constraints. Each particle has an associated radial force field which leads to a dynamically changing topology of interactions. Our approach is related to the behaviour of *flocks*[6] in that each particle reacts to its neighbouring particles and to the local environment.

2. GLOBULAR DYNAMICS

We will refer to an element of the connected particle system as a *globule*, thus avoiding the established connotations associated with words such as particle or blob. The word globule means "sphere like" but it also has associations with thick viscous liquids such as oil.

Each globule is associated with several characteristics as shown in Table 1.

We are interested in modeling *soft* collisions between globules to avoid the rigid stacking problems exhibited by marbles or billiard balls. Soft collisions involve forces which vary gradually with distance which allows globules to flow over one another.

There are other parameters available, such as colour, but we will limit our discussion to those which are relevant to globular dynamics. Detailed methods for shading multi-coloured particles and solid texturing isosurfaces have been developed elsewhere[7].

2.1. *Globule to globule forces*

The positional change in the globule over the time step (t) is calculated by double integration of the sum of the forces acting on the particle:

$$x_p = \frac{1}{M_p} \iint \sum_{i=1}^{N} f_{ip} \, dt\, dt \tag{1}$$

where

$$f_{ip} = \begin{cases} \bar{P}_\Delta \left[s_r \left(\dfrac{b_1}{D^m} - \dfrac{b_2}{D^n} \right) - s_d \dfrac{(\bar{V}_\Delta \cdot \bar{P}_\Delta)}{D^2} \right] & \text{if } i \neq p \\ 0 & \text{if } i = p \end{cases} \tag{2}$$

and where $\bar{P}_\Delta$ is the vector formed by the positional difference of the two globules x_i and x_p, $\bar{V}_\Delta$ is the difference in their velocities, D is the distance between them and m and n are constants which define the variation of force with distance. For the illustrations in this paper, $m = 5$ and $n = 3$.

The variable b_2 is related to b_1 by $b_2 = b_1 r_0^{n-m}$ where r_0 is the inter-globule spacing for which the attraction and repulsion terms cancel exactly.

The two scaling factors, s_r (repulsion/attraction) and

Table 1. Globule characteristics.

Type	Field Name	Function
float	M	Mass
float	r	Radius
float	T	Temperature
point	x	Current position
vector	V	Velocity

s_d (drag), attenuate the inter-globule force based on distance. These factors describe a radius of influence for the globules. Their values are given by:

$$s_r = \begin{cases} 1 - \dfrac{D^2}{c_r^2(r_i + r_p)^2} & \text{if } D^2 < c_r^2(r_i + r_p)^2 \\ 0 & \text{if } D^2 \geq c_r^2(r_i + r_p)^2 \end{cases}$$

$$s_d = \begin{cases} 1 - \dfrac{D^2}{c_d^2(r_i + r_p)^2} & \text{if } D^2 < c_d^2(r_i + r_p)^2 \\ 0 & \text{if } D^2 \geq c_d^2(r_i + r_p)^2. \end{cases} \quad (3)$$

To reduce the amount of computation needed for (2), globules which are sufficiently distant not to interact can be culled by detecting $s_d = 0$. Also, in the case where $s_r = 0$, the repulsion/attraction term in (2) need not be calculated.

The two terms c_r and c_d allow different types of material to be modeled. The repulsive force between globules is modulated using an $(r_{\max}^2 - r^2)$ term to limit the forces to a finite range. At $r = r_{\max}$, the force is zero.

Powders. For powder-like motion the damping and radial forces are equally attenuated so that damping only occurs when the globules are under compression:

$$c_r = r_0$$
$$c_d = r_0. \quad (4)$$

Eq. (4) produces an inelastic collision when globules collide. Fig. 1 shows a graph of the repulsive force as a function of distance and Images 1a–d are a sequence of frames from an animation for which $c_r = c_d = r_0$. The globules bounce off each other and show no tendency to cluster in lumps. The globules have a tendency, however, to stack against the walls of the constraints, an effect which can be remedied by randomising the radii of the globules, or by modeling collisions with the walls as slightly elastic.

Since in 2-D the globules are discs which collide inelastically, their kinetic energy decays over time. They fall under gravity and stack into a hexagonal arrangement. The globules will flow to fill up available space. In this state they are behaving more like grains of sugar or sand than gas molecules. If all the globules are the same radius the resultant perfect crystal has preferential directions of slip. Randomising the radii of the globules should help simulate material more isotropic in its physical properties.

Fluids. To simulate a liquid, set $c_d = 2r_0$ and $c_r = r_0$. Globules tend to cluster because relative motion is damped, but external forces can part the globules easily. Images 2a–d are frames from an animation with $c_d = 2r_0$. The globules drip over the edge of the constraints and cluster into heaps.

Solids. To model a solid which can tear or shatter, the attraction term in (2) is used. By setting $c_r = c_d = 2r_0$, we create a short range repulsion, a medium range attraction and long range indifference (see Fig. 2). Images 3a–d are frames from an animation with these parameters. The globules cluster into a gelatinous solid with little relative movement occurring once they attach to the main mass.

In all cases for $r < r_0$, the repulsion increases as the distance between two globules decreases, an effect which becomes increasingly severe as the distance between them becomes small. As a consequence, there is an approximate measure of volume conservation in the model. The strength of the repulsion is controlled by the size of b_1; the larger its value, the less compressible the material. As the repulsion term is increased, however, the numerical solutions become less stable.

The temperature values for a given pair of globules can be used to change their interaction behaviour, allowing the melting of solids and the freezing of liquids. The effects of temperature transference between globules is an area for further research.

2.2. *Globule-constraint interactions*

The method of integration for the acceleration of a globule depends on how external constraints are implemented. Impulse-based collisions detect whether the motion of a globule intersects a surface, in which case the new position and velocity are computed analytically (see Fig. 3).

For a globule travelling from $\bar{P}_j$ to $\bar{P}_{j+1}$, the intersection with a plane is calculated given the plane normal, $\bar{N}$, and a point on the plane, $\bar{C}$. First the plane tangent is calculated for the point of intersection.

$$\bar{T} = \frac{\bar{N} \times (\bar{N} \times (\bar{P}_j - \bar{P}_{j+1}))}{|\bar{N} \times (\bar{P}_j - \bar{P}_{j+1})|} \quad (5)$$

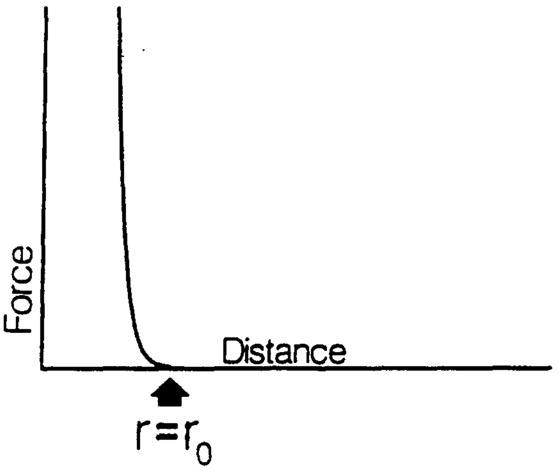
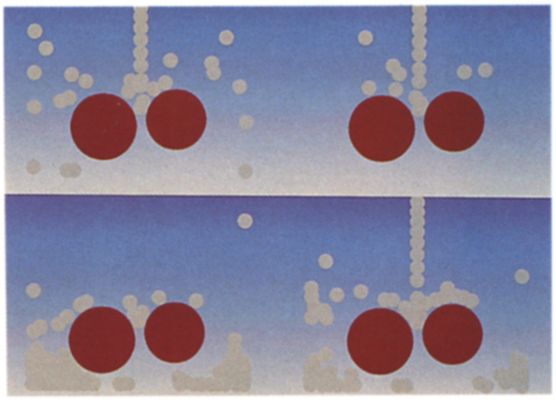
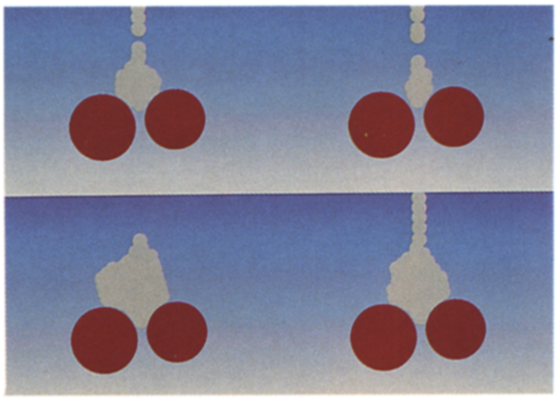


Fig. 1. Graph of finite range forces for powders and fluids.


Images 1a–d. Powder.


Images 3a–d. Solid.

where

$$\bar{I} = \bar{P}_j + (\bar{P}_{j+1} - \bar{P}_j)D \qquad \text{(Ray Eq.)}$$

$$(\bar{I} - \bar{C})\cdot\bar{N} = 0. \qquad \text{(Plane Eq.)}$$

Hence

$$D_{\text{intersect}} = \frac{(\bar{C} - \bar{P}_j)\cdot\bar{N}}{(\bar{P}_{j+1} - \bar{P}_j)\cdot\bar{N}}$$

$$\bar{I} = \bar{P}_j + (\bar{P}_{j+1} - \bar{P}_j)D_{\text{intersect}}. \qquad (6)$$

Define the residual vector $\bar{R}$ to be

$$\bar{R} = \bar{P}_{j+1} - \bar{I}. \qquad (7)$$

Define the reflection vector $\bar{R}'$ to be

$$\bar{R}' = \bar{P}'_{j+1} - \bar{I}. \qquad (8)$$

Therefore

$$\bar{R}' = -\bar{N}(\bar{N}\cdot\bar{R})r_n + \bar{T}(\bar{T}\cdot\bar{R})r_t$$

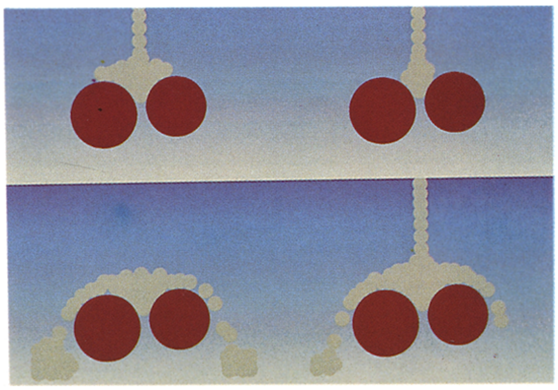
where r_n is the coefficient of normal reflection, and r_t is the coefficient of tangent reflection.

The advantage of such a scheme is that the collision detection calculation is simple, it is equivalent to intersecting a ray with a surface. Collisions are calculated in the static coordinate frame of the constraint, so that P_j and P_{j+1} take account of the motion of the constraint relative to the world coordinate system between successive time steps. Another advantage is that the points are guaranteed not to penetrate the constraint.

When computing the ray-surface intersection, care must be taken to select the correct surface. If the ray which represents the path of the centre of the globule intersects the original surface constraint, then half of the globule will penetrate the surface at the point of intersection. This effect may be acceptable, for instance with drops of a liquid on a floor. It may be required, however, that no part of the globule penetrates the constraint. A thinly walled bottle or a container with comparatively large globules is not rendered correctly if parts of the globules protrude through the sides.

The problem may be overcome by two approaches; either the walls of the container are modelled thicker or the constraint may be placed offset from the actual surface. Offsetting a surface along its normal by the radius of the globules will guarantee that no globules intersect the surface.

Surface offsets may be computed using higher order patches, direct polygonal manipulation or, for arbitrary


Images 2a–d. Fluid.

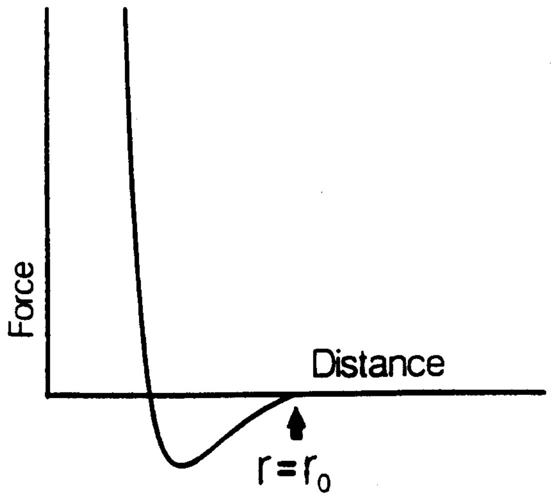


Fig. 2. Graph of finite range forces for solids.

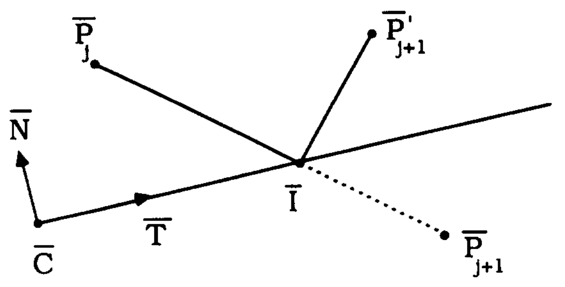


Fig. 3. Globule-constraint collision.

height fields, a z-buffer. Surface offsetting has received considerable attention because of its application to NC milling path generation techniques[8]. It is usually more efficient to offset the geometry once for a given radius and then calculate the intersection rather than compute a new offset for each intersection. This is unfortunate since, as explained earlier, it is sometimes desirable to have globules with randomly varied radii. In the case of randomly sized particles, a reasonable compromise would be to use the maximum radius of the globules when creating the offset constraint.

The disadvantage of impulse-based collisions is that only Euler integration produces a continuous solution:

$$v_{j+1} = v_j + a_j\Delta t$$
$$x_{j+1} = x_j + v_j\Delta t + \tfrac{1}{2}a_j\Delta t^2. \qquad (9)$$

Euler integration requires smaller time steps than higher order methods such as Runge-Kutta[9]. For such methods, however, the constraint must be expressed in terms of a repulsion which increases as the globule approaches the surface, which may be expensive (see [5, 9]).

For the globular dynamics to be stable it is necessary that a globule only moves a small fraction of its radius in each time step, relative to other globules and relative to the constraints. Such small time steps mean that impulse-based collisions are acceptably accurate without special treatment of the exact collision times within a step. This is in contrast to the simulation of impulse-based collisions of rigid bodies such as the technique described in [10].

3. RENDERING GLOBULES

In order to render the globules we wish to cover them with an isosurface. Most hidden surface algorithms cannot directly render an isosurface. Past solutions have been to approximate the isosurface by a polygonal mesh[11, 12] or to ray trace them directly[2]. Our approach is to approximate the appearance of an isosurface by smooth-shading spheres based solely on the gradient of a potential function.

The normal for a point on the surface of a sphere (x, y, z) is defined by:

$$i_p^2 = (x - x_p)^2 + (y - y_p)^2 + (z - z_p)^2$$
$$n_x = \sum_{p=1}^{N} (x - x_p)\left[\frac{(r^2 - i_p^2)}{i_p^2}\right] \qquad (10)$$

and similarly for n_y and n_z, thus avoiding square root calculations except while normalising the final sum. In the images shown, the value of r chosen for the rendering was twice r_0 for the dynamics and twice the radius of the sphere models.

Isosurface gradients are calculated on a per pixel (or sub-pixel) basis so that only visible surfaces need to be computed, limiting the number of calculations as the number of globules grows. In addition, it allows for the calculation of the surface normal as a post process if the rendered spheres are stored in a z-buffer[13]. This allows hardware which is very fast at rendering spheres, or polygons if the spheres are tessellated, to perform the hidden surface elimination.

While smooth-shading the sphere normals aids the appearance of the centre of the spheres, the silhouette edges of the many spheres are still distinct. Increasing the sphere size relative to the globule to globule repulsion range diminishes this problem but leads to a lack of detail in the visualisation.

4. SPATIAL SUBDIVISION

Calculating the forces which the globules exert upon one another is of the order $O(N^2)$. Fortunately, globules have a limited range of influence, a property which allows a simple and fast method of voxelising the globules to reduce the number of calculations of (2). By subdividing space into $c_d r_{\max}$ sized voxels, a globule is only assigned to the one voxel which surrounds its centre point. When calculating the force acting upon a globule from other globules, only those globules in the same voxel, or any of the neighbouring 26 voxels can possibly exert any influence, reducing the calculation of forces to an order $O(N)$ problem. The same spatial subdivision scheme can be used for the rendering stage and frame coherence can be exploited by avoiding recalculation of voxels which do not change between frames[14]. Polygonal constraints must be 3-D scan-converted into the voxels much as polygonal surfaces are scan-converted for ray tracing using voxelisation[15].

Table 2. Animation statistics.

Animation	Number of globules	Dynamics Calculation (seconds/frame)	Subintervals per frame
Powder (1a-1d)	75	34	30
Fluid (2a-2d)	75	34	30
Solid (3a-3d)	75	34	30
Fountain (4a-4d)	300	55	20
Column (5a-5d)	108	45	5

4.1. *Parallelisation of globule force calculations*

Globules by definition have a limited and constant radius of influence allowing for a multi-processor approach where each processor needs to communicate only with its nearest neighbours. Also, because there is an upper limit on the number of globules which can occupy any given volume of space, the data can be distributed evenly between many processors with limited local memory.

Other speed issues. Decreasing the size of r and r_0 produces improved images due to a more detailed isosurface, but it also leads to shorter range interactions which decreases the stability of the solutions. To regain stability in the calculation, the length of the time step per iteration must be reduced, resulting in more iterations per frame.

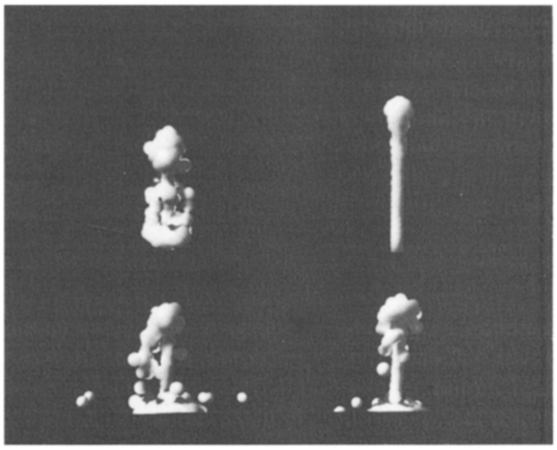
When reducing the radius of the globules, the number of spheres increases as $1/r^3$ for the volume. The number of iterations increases as $1/r$ resulting in a $1/r^4$ increase in processing time as the size of the globules decreases.

The current implementation is written in Alias's interpreted Scene Description Language (SDL). An implementation in a compiled language paired with better integration schemes will improve the speed of the approach.

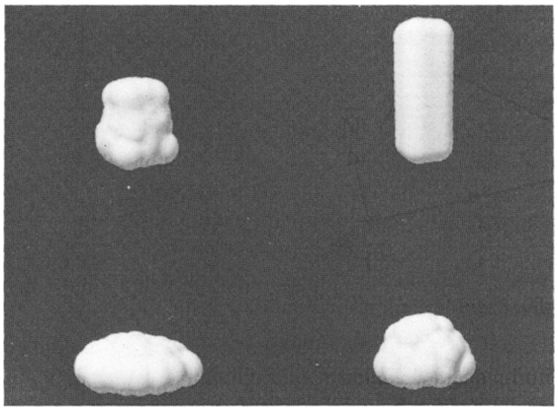
Table 2 presents timings and other relevant statistics about the animations presented in this paper. The timings are from a Silicon Graphics 4D workstation. Two additional three-dimensional example animations are included in the table. The first is from a sequence showing a fountain of fluid (Images 4a–d) and the second is a column of fluid collapsing under gravity (Images 5a–d).

5. CONCLUSIONS

A coherent dynamic model has been presented for modeling fracturable solids, solid/liquid transitions and jets or sprays of liquids. The method uses a coarse approximation of the molecular movement within the fluid. The globules used in the approximation have properties which allow them to be rendered without the expense of actually computing an isosurface. The method seems promising for rendering complex animations of fountains and other flows of viscous liquids.



Images 4a–d. Fountain.



Images 5a–d. Column.

Acknowledgements—We would like to thank the reviewers for their many helpful suggestions. The research described was carried out at Alias Research, Inc., as part of the Natural Phenomena Project, with additional work at Apple Computer, Inc. We thank Alias and Apple for encouraging this work. Scene Description Language and SDL are trademarks of Alias Research, Inc.; Silicon Graphics is a trademark of Silicon Graphics, Inc.

REFERENCES

1. J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Company, Reading, MA, pp. 523–537 (1982).
2. J. Blinn, A generalization of algebraic surface drawing. *ACM Trans. on Graphics* **1**, 235–256 (1982).
3. W. T. Reeves, Particle systems—A technique for modeling a class of fuzzy objects. *SIGGRAPH* 83 *Comp. Graphics* **17**(3), 359–376 (1983).
4. B. Wyvill, C. McPheeters, and G. Wyvill, Animating soft objects. *The Visual Comp.* **2**, 235–242 (1986).
5. D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, Elastically deformable models. *SIGGRAPH* 87 *Comp. Graphics* **21**(4), 205–214 (1987).
6. C. W. Reynolds, Flocks, herds, and schools: A distributed behavioral model. *SIGGRAPH* 87 *Comp. Graphics* **21**(4), 25–34 (1987).
7. G. Wyvill, B. Wyvill, and C. McPheeters, Solid texturing for soft objects. *IEEE Comp. Graphics and Appl.* **7**(12), 20–26 (1987).
8. G. S. P. Miller, Computer Display and manufacture of 3-D models. Ph.D. Thesis, Cambridge University Engineering Department, Cambridge, England (June 1987).
9. J. Wilhelms and M. Moore, Dynamics for everyone. Appendix 1, SIGGRAPH 87 Tutorial Notes, Course 10: Computer Animation: 3-D Motion Specification and Control, pp. 145–146 (July 1987).
10. M. Moore and J. Wilhelms, Collision detection and response for computer animation. *SIGGRAPH* 88 *Comp. Graphics* **22**(4), 289–298 (1988).
11. G. Wyvill, C. McPheeters, and B. Wyvill, Data structure for soft objects. *The Visual Comp.* **2**, 227–234 (1986).
12. J. Bloomenthal, Polygonization of implicit surfaces. SIGGRAPH 87 Tutorial Notes, Course 16: The Modeling of Natural Phenomena, Section 3 (July 1987).
13. K. Perlin, An image synthesizer. *SIGGRAPH* 85 *Comp. Graphics* **19**(3), 287–296 (1985).
14. D. Jevans, B. Wyvill, and G. Wyvill, Speeding up 3D animation for simulation. *Proc. SCS Conference* 1988, pp. 94–100, *Proc. MAPCON IV* (Multi and Array Processors) (1988).
15. A. P. Pearce, An implementation of ray tracing using multiprocessing and spatial subdivision. Masters Thesis, University of Calgary Computer Science Department, Calgary, Canada (October 1987).