A Novel Representation of 3D Shapes Based on Modified Sphere Equation

Piotr Kurek*

Computer Graphics Group
Technical University of Szczecin
Szczecin / Poland

Abstract

Term blob is used with reference to 3D models with smooth shapes which can be described as a liquid in weightlessness. There are several methods of getting blobs. Our main goal has been to present novel method called GSE (Generalized Sphere Equation). GSE method allows getting models which are generalization of classical blobs. This paper includes description of input data, mathematical interpretation and solutions which allow rendering pictures of GSE models using ray tracing.

Keywords: blobby models, soft objects, 3D modeling, ray tracing, image synthesis

1 Introduction

The blob term is used with reference to 3D models with smooth shapes. This paper discusses a new method of obtaining models which are generalization of classical blobs. The method has been called GSE (Generalized Sphere Equation). The GSE models differ significantly, depending on parameters which have influence on their shape, size and surface. The simplest variant of GSE method allows to obtain the classical blobs. In more complicated cases the GSE models can have many kinds of various surface bumps (see section 3.1).

Blob models are described as three-dimensional surfaces. There are two types of such surfaces: parametric and implicit.

Parametric surface is a set of points generated by three functions of given number of variables. For example, bivariate parametric surfaces are generated by functions of two variables: $f_x(i,j), f_y(i,j), f_z(i,j)$. Implicit surface is a set of points which satisfy some equation in the three-dimensional space:

$$F(x,y,z) = 0.$$

In case when above-mentioned equation is of the second degree we deal with quadric surfaces. (This term is also used with curves.) Thus, surface equation is:

$$Ax^2 + Bxy + Cxz + Dx + Ey^2 + Fyz + Gy + Hz^2 + Iz + J = 0. \quad (1)$$

Depending on values of parameters $A,\ B,\ ...,\ J$ equation 1 can describe plane, sphere, cone, cylinder and the like.

Section 2 examines how the surfaces allied to quadrics have been used to making blobs. Section 3 includes the input data and the mathematical interpretation of the GSE method. In section 4 the usage of GSE method in ray tracing is discussed. There is description of implementation and results in section 5. The last section includes conclusions and discussion of future works.

2 Related Works

2.1 J. F. Blinn's method

The first blobs were generated in 1982, when James F. Blinn used a method allied to quadric surfaces to solve the problem of computer aided visualization of molecular model.

In [1], a physical interpretation is given for mathematical solutions used. According to quantum mechanics, the electron in an atom can be represented as a density function of the spatial location. For a hydrogen atom, density function is:

$$D(x,y,z) = e^{-ar},$$

where

$$r = \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2}$$
$$(x_1, y_1, z_1) - \text{the center of an atom.}$$

According to superposition theorem, density function can be used for many atoms. Thus the sum of density contributions of each atom should be taken into account:

$$D(x,y,z) = \sum_i b_i e^{-a_i r_i},$$

where $r_i = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}$ is the distance from (x,y,z) to the center of atom i. A molecular surface can be defined as a set of points where density

*pkurek@wi.ps.pl

function equals some threshold value T. Using implicit surface definition a molecular surface can be described as:

$$F(x,y,z) = \sum_i T e^{-a_i r_i^2 - B_i}.$$

In above equation b_i has been specified in term of a blobbines parameter B_i.

[1] describes how to obtain the pictures of Blinn's blobs using ray tracing. Deriving ray equation begins with the various transformations into a standard viewing space. Reciprocal transformations let calculate viewing ray equation. For each point of the ray, coordinates x and y depend on the z coordinate. Thus, calculating the z coordinate is enough to get the ray and surface intersection. J. F. Blinn has divided z value calculating into two phases:

1. Root isolation phase: finding the range in which is a solution. In Blinn's solution for each viewing ray, a list of n values: $z_1, z_2, ...z_{n-1}$ is made. The z_i value is a z coordinate of this point on the ray in which atom i has the biggest influence D_i on density function D. The list of $z_1, z_2, ...z_{n-1}$ values is sorted in ascending order. The list is searched for the first value z_i for which $D_i(z_i)$ is greater than or equals T. Thus, $\langle z_{i-1}, z_i \rangle$ is a required range for numeric methods of finding zeros.

2. Root refinement phase: finding a solution in a given range (using general numeric methods of finding zeros, like Newton method or $regulafalsi$ method). Blinn suggested using combination of two numeric methods of finding zeros: Newton method and the $regulafalsi$ method.

The surface normal at a given point can be found by taking the gradient of the surface defining function, F:

$$N = \left[\frac{\partial F}{\partial x}(x,y,z), \frac{\partial F}{\partial y}(x,y,z), \frac{\partial F}{\partial z}(x,y,z) \right],$$

$$\frac{\partial F}{\partial x} = \sum_i -2a_i(x - x_i)e^{-a_i r_i^2 - B_i},$$

$$\frac{\partial F}{\partial y} = \sum_i -2a_i(y - y_i)e^{-a_i r_i^2 - B_i},$$

$$\frac{\partial F}{\partial z} = \sum_i -2a_i(z - z_i)e^{-a_i r_i^2 - B_i}.$$

Described method works well for small number of atoms (a few dozen). Yet it is much to slow when number of atoms is in the order of a few thousands. That is why the algorithm has been optimized.

The idea is based on the fact that only a part of atoms have influence on surface and ray intersection. Thus, the calculation can be limited to these atoms which are close enough to the viewing ray. Each atom could then be enclosed in a sphere. The enclosing spheres of all atoms are projected into screen space. Thus, it is possible to find the atoms which can have influence on the color of each pixel.

2.2 Other Solutions for Blinn's Method

The solutions of optimizing the algorithm of ray tracing implicit surfaces are described in [3].

While looking for intersection of viewing ray and implicit surface it is usually not possible to use an analytical method. That is because surface functions are very complicated as it was in case of the function H in J. F. Blinn's method.

The solution is to use a numeric analysis. Usually, modifications or combinations of classical numeric methods are used, preceded by necessary transformations. Authors follow J. F. Blinn and divide calculations into root isolation phase and root refinement phase.

One of the methods of optimizing the root isolation phase is to approximate the complicated implicit function with a polynomial. In 1986, Geoff Wyvill suggested the approximation by the polynomial function $C(r^2)$:

$$C(r^2) = \begin{cases} -\frac{1}{144}(r^2)^3 + \frac{17}{144}(r^2)^2 - \frac{11}{18}r^2 + 1 \\ \qquad \text{if } \mathrm{r}^2 < 4 \\ 0 \text{ otherwise.} \end{cases}$$

The received models have been called soft objects. They are described in [2].

[5] describes using method taken from numerical analysis in root isolation phase. The method is called interval analysis and originated in 1966 [6].

To start the interval analysis, a function and a bracket of arguments must be given. The method narrows given bracket down to new bracket in which function is monotonic and has zero.

An interval $[a, b]$ is an ordered pair $a \geq b$ representing the range of numbers $x : a \geq x \geq b$.

Several operations on intervals were defined: addition, subtraction, multiplication, division, squaring, exponentation. They allowed to calculate the value of Blinn's surface function when the argument is a given interval.

The interval analysis method loops over five steps as follows:

1. Calculating the interval $[s_0, s_1] = H[t_0, t_1]$,

2. If $0 \notin [s_0, s_1]$, it means that in the bracket $\langle t_0, t_1 \rangle$ function doesn't have zeros. The algorithm is finished with the adequate communicate,

3. Calculating the interval: $[r_0, r_1] = H'[t_0, t_1]$,

4. If $0 \notin [r_0, r_1]$, it means that the function f is monotonic in bracket $\langle t_0; t_1 \rangle$. The algorithm is finished and the $\langle t_0, t_1 \rangle$ bracket is returned.

5. Back to the first step for intervals: $[t_0, t_0 + 1(t_0 + t_1)]$ and $[t_0 + 1(t_0 + t_1), t_1]$.

Realization of above algorithm gives monotonic segment of the function, so then the root refinement phase can start.

3 Mathematical Interpretation of GSE Method

The idea of the GSE method is based on observation that sphere is a set of points which distance from one point (center) is constant whereas ellipsoid is a set of points which sum of distances from two points is constant. The idea is to generalize this principle for three, four or more points. Thus, the generalized sphere equation is:

$$\sum_{i=0}^{n-1} d_i = d, \tag{2}$$

where

$d_i = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}$,
(x, y, z) is the point on the surface,
(x_i, y_i, z_i) is the ith equivalent of the center.

We assume that the models, made using the GSE method must meet the following conditions:

1. The model surface should be determined by a set of points (they have been called gravity points).
2. The model should have soft shape.
3. The gravity points should have only a local influence on the shape of model (to make modeling intuitive).

The models described by equation 2 meet the first and second conditions, but the third condition which concerns local influence of the gravity point on the model shape is not met. So, the GSE model equation has been expanded to:

$$\frac{\sum_{i=0}^{n-1} w(d_i)\, d_i}{\sum_{i=0}^{n-1} w(d_i)} = d, \tag{3}$$

where

$$d_i = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2}$$
$$w(d_i) = \varphi_i \left(f(d_i) + g(d_i)\right).$$

Derivation of the equation 3 can be found in section 3.2. The accurate meaning of $f(d_i)$, $g(d_i)$ and φ_i is expleined in section 3.1. In case when $n = 1$, $\varphi_0 = 1$, $f(d_i) = 1$, $g(d_i) = 0$ we result in a sphere equation (or circle on the plane) were the radius equals d:

$$\frac{d_0}{1} = d \Longrightarrow d_0 = d, \text{ or}$$

$$\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} = d.$$

3.1 Input Values

Every model is described by:

- set of n $(n \geq 1)$ gravity points in the 3D space $\{(x_0, y_0, z_0), ..., (x_{n-1}, y_{n-1}, z_{n-1})\}$,

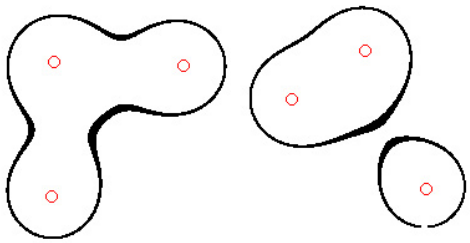

Figure 1: The influence of the gravity points on the model shape

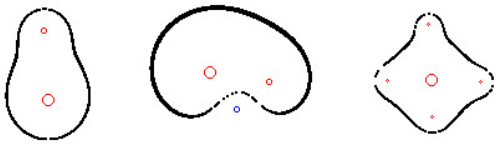

Figure 2: The influence of the weights assigned to the determining points on the model shape

- set of weights $\{\varphi_0, ...\varphi_i, ..., \varphi_{n-1}\}$, $\varphi_i \in [-1, 1]$ assigned to the points,
- d parameter determining size of the model,
- pair of functions f and g describing the shape and the bumps of the model, respectively.

The gravity points describe shape of the model. Every intersection of the model is a closed curve of soft shape or set of such curves (fig. 1). The weights describe influence of every single point on shape of the model. Weights range from –1 to 1, both inclusive. In fig. 2 there are model intersections with the gravity points marked. The d parameter determines size of the model. The larger the model, the smaller the influence of each gravity point (fig. 3).

The function f describes the overall shape of the model. It's arguments are $d_0, d_1, d_2, ..., d_{n-1}$. d_i stands for the distance between a given surface point and the ith gravity point. For the model to have expected shape, the function f should have smooth graph. Moreover, the function f should assign lower values to the further gravity points.

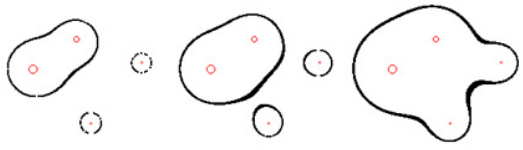

Figure 3: The influence of the d parameter on the model size and shape

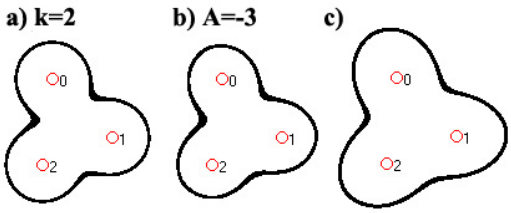

Figure 4: The influence of the function f on the model shape

Example equations that meet both conditions are shown below. (d_{max} stands for the distance between given surface point and the furthest gravity point.)

a) $f(d_i) = \begin{cases} \frac{1}{d_i^k} \; for \; d_i \neq 0 \\ 0 \; for \; d_i = 0 \end{cases}$

b) $f(d_i) = \begin{cases} \frac{1}{2}\left(\cos\left(\frac{d_i}{d_{\max}}\pi\right) + 1\right) for \; d_i \neq 0 \\ 0 \; for \; d_i = 0 \end{cases}$

c) $f(d_i) = \begin{cases} \frac{1}{2}\left(\cos\left(\frac{d_i}{d_{\max}}\pi\right) + 1\right) for \; d_i \neq 0 \\ 0 \; for \; d_i = 0 \end{cases}$

The function g describes the surface bumps. In previous examples, the function g was constant and equal to 0. In fig. 5 an example of a model with non-constant function g is shown.

3.2 Derivation of GSE

Let's exchange the sum on the left side of equation 2 with the weighted average:

$$\frac{\sum_{i=0}^{n-1} w(d_i)d_i}{\sum_{i=0}^{n-1} w(d_i)} = d.$$

Coefficients $w(d_0)...w(d_{n-1})$ should meet the following conditions:

1. The coefficients should be chosen so that the gravity points located closer to a given point of the surface have more influence on it's location than further gravity points.
2. The coefficients should depend on weights φ_i assigned to each gravity point.
3. The coefficients should have influence on surface bumps described by the function g.

The first condition means that coefficients should have the greatest values for the nearest gravity points and the smallest values for the furthest ones. An example solution is the inverted square of the distance:

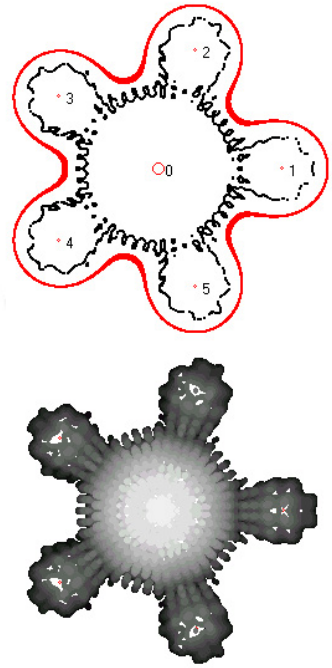

Figure 5: Intersection (on top) and draft 3D image of model with non constant function g

$$w''(d_i) = \frac{1}{d_i^2}.$$

Let's generalize and replace the inverted square of the distance with any function f(see section 3.1) that meets the first condition:

$$w''(d_i) = f(d_i).$$

To meet the second condition, value of each coefficient should be determined by a weight assigned to each gravity point :

$$w'(d_i) = \varphi_i w''(d_i) = \varphi_i f(d_i).$$

To meet the third condition, we need to take into consideration the surface bumps described by the function g (see section 3.1).

$$w(d_i) = \varphi_i(f(d_i) + g(d_i)).$$

4 Application of GSE Method in Ray Tracing

This section discusses two calculating methods: finding the intersection of a ray with a GSE model and finding normal vectors. These two methods enable the GSE models to be used in ray tracing.

4.1 Finding the Intersection of a Ray with a GSE Model

4.1.1 Problem Description

Finding the intersection of a ray with a GSE model can be realized using every method used for blobs. The method used in the GSE ray tracer is similar to the method described in [1], but seems to be simpler in implementation. In future it will probably be replaced with the interval analysis method.

Let dir be the vector determining the viewing ray direction in ray tracing method. Then, to find the intersection of the ray with the GSE model, we need to solve the following equations:

$$\begin{cases} \frac{\sum_{i=0}^{n-1} \varphi_i(f(d_i)+g(d_i))d_i}{\sum_{i=0}^{n-1} \varphi_i(f(d_i)+g(d_i))} = d \\ x = x_0 + dir_x t \\ y = y_0 + dir_y t \\ z = z_0 + dir_z t. \end{cases}$$

The last three of the above equations are the parametrical description of a line in space. After inserting x, y and z, we result in:

$$\frac{\sum_{i=1}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right) d_i}{\sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right)} = d, \tag{4}$$

$$\begin{aligned} d_i &= \sqrt{d_x^2 + d_y^2 + d_z^2} \\ d_x &= (x_0 + dir_x t - x_i) \\ d_y &= (y_0 + dir_y t - y_i) \\ d_z &= (z_0 + dir_z t - z_i)\,. \end{aligned}$$

The solution is to find the minimal positive value of the t parameter. Due to the complexity of equations 4, instead of searching an analytical solution method, a numerical method was used. As in [1] it has been divided in to two phases: root isolation phase and root refinement phase.

4.1.2 Root Isolation Phase

For a given t, let's denote solution error of equation 4 with $\varepsilon(t)$ (d_i as in equation 4):

$$\varepsilon(t) = d - \frac{\sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right) d_i}{\sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right)}.$$

The precise solution is the smallest of the error function's zero points. Thus, we have to numerically find the smallest t where $\varepsilon(t) \approx 0$.

Let's consider n planes $\pi_0, \pi_1, \pi_2, ...\pi_{n-1}$ crossing the appropriate gravity points and perpendicular to the direction of a given viewing ray. The ray intersects the $\pi_0, \pi_1, \pi_2, ...\pi_{n-1}$ planes in n points determined by the n values of the t parameter: $t_0, t_1, ...t_{n-1}$. We can calculate the intersections of the ray with the planes by combining

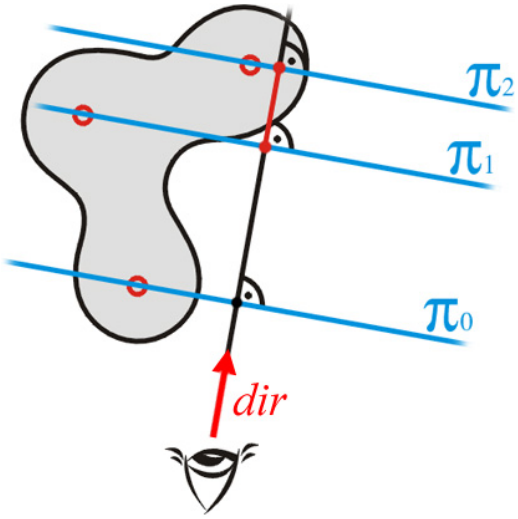

Figure 6: Isolation of the first zero

the ray equation with the appropriate plane equation. The mathematical description can be found in [4].

$$\left.\begin{aligned} dir_x(x_i - x) + dir_y(y_i - y) + \\ dir_z(z_i - z) = 0 \\ x = x_0 + dir_x t \\ y = y_0 + dir_y t \\ z = z_0 + dir_z t \end{aligned}\right\} \Longrightarrow$$
$$t_i = dir_x(x_i - x_0) + dir_y(y_i - y_0) + dir_z(z_i - z_0).$$

From the geometrical interpretation we know that in ranges where the ray runs inside the model, the error function has positive values, while in ranges where the ray runs outside the model the error function has negative values.

Thus, we can deduce that the smallest zero point of the function can be located (see fig. 6):

- in the $\langle t_0 - r;\ t_0 \rangle$ bracket if $\varepsilon(0) < 0$ and $\varepsilon(t_0) \geq 0$,
- in the $\langle t_{i-1};\ t_i \rangle$ bracket if $\varepsilon(t_0), \varepsilon(t_1), ..., \varepsilon(t_{i-1}) < 0$ and $\varepsilon(t_i) \geq 0$.

To recap, the algorithm for root isolation phase can be divided in to the following steps:

1. Finding the t_i values for ray plane intersection (One plane is created for each consecutive gravity point),
2. Sorting the t_i values,
3. Checking if the following condition is met for consecutive t_i:

$$\varepsilon(t_i) \geq 0,\ i = 0, 1, 2, ..., n-1.$$

If yes, we can stop checking condition and return the bracket $\langle T_1; T_2 \rangle$:

$$\begin{aligned} \langle T_1; T_2 \rangle &= \langle t_{i-1}; t_i \rangle \text{ if } i > 0, \text{or} \\ \langle T_1; T_2 \rangle &= \langle t_0 - r; t_0 \rangle \text{ if } i = 0. \end{aligned}$$

4.1.3 Root Refinement Phase

After finishing the above algorithm, we result in a bracket $\langle T_1; T_2 \rangle$ where $\varepsilon(T_1) < 0$ and $\varepsilon(T_1) > 0$.

Now, we can find the function's zero point using one of classical numeric method. The iterative secant method, which is especially well suited, can be found in [7].

4.2 Finding the Normal Vectors

This section discusses two methods of finding the normal vectors for a the GSE models. The first method is quite quick but it works only if $g(d_i) = 0$. The second method is slow than the first one but it works also if $g(d_i) \neq 0$

4.2.1 Method Based on the Dummy Centers

If assume that $g = 0$ the method of finding the normal vectors can be based on the idea of "dummy centers" defined in the following way:

Definition. For a given surface point P(x, y, z) of a solid, a **dummy center** is a C point such so C-P is a normal vector beginning in P.

For the solids described here, a dummy center for any surface point P is a weighted average of the gravity points:

$$\left.\begin{aligned} C_x &\approx \frac{\sum_{i=0}^{n-1} \varphi_i f^2(d_i) x_i}{\sum_{i=0}^{n-1} \varphi_i f^2(d_i)} \\ C_y &\approx \frac{\sum_{i=0}^{n-1} \varphi_i f^2(d_i) y_i}{\sum_{i=0}^{n-1} \varphi_i f^2(d_i)} \\ C_z &\approx \frac{\sum_{i=0}^{n-1} \varphi_i f^2(d_i) z_i}{\sum_{i=0}^{n-1} \varphi_i f^2(d_i)} \end{aligned}\right\}, \quad \left.\begin{aligned} N_x &= x - C_x \\ N_y &= y - C_y \\ N_z &= z - C_z \end{aligned}\right\}.$$

For a sphere, after inserting $n = 1$, $\varphi_0 = 1$, $f(d_i) = 1$, we result in the geometrical center coordinates:

$$\left.\begin{aligned} C_x &= \frac{1 \cdot d_0^2 x_0}{1 \cdot d_0} \\ C_y &= \frac{1 \cdot d_0^2 y_0}{1 \cdot d_0} \\ C_z &= \frac{1 \cdot d_0^2 z_0}{1 \cdot d_0} \end{aligned}\right\} \Longrightarrow \begin{aligned} C_x &= x_0 \\ C_y &= y_0 \\ C_z &= z_0 \end{aligned} \quad , \quad \begin{aligned} N_x &= x - x_0 \\ N_y &= y - y_0 \\ N_z &= z - z_0. \end{aligned}$$

Fig. 7 a) shows results of the described method. Normal vectors are marked with blue lines and the dummy centers with green dots.

4.2.2 Method Based on the Gradient

In case when $g \neq 0$ the normal vector can not be found using dummy centers method. However every implicit surface normal can be found by taking the gradient of the surface defining function. Thus, according to equation 3 we have:

$$F(x, y, z) = \frac{\sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right) d_i}{\sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right)} - d,$$

$$\nabla F = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right].$$

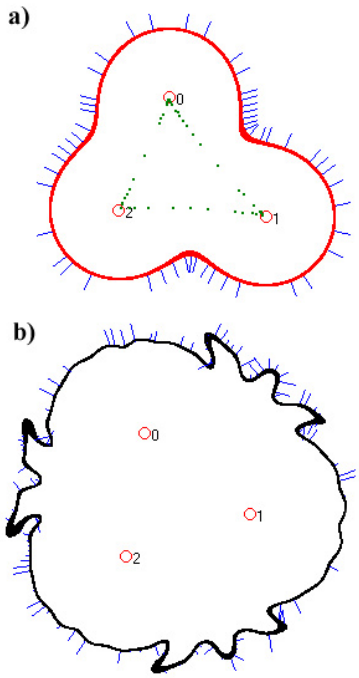

Figure 7: GSE normal vectors calculated using the dummy centers (on top) and the gradient (on bottom)

For example the x componenet will be

$$\frac{\partial F}{\partial x} = \frac{\frac{\partial N}{\partial x} D + N \frac{\partial D}{\partial x}}{D^2},$$

where
$N = \sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right) d_i,$
$D = \sum_{i=0}^{n-1} \varphi_i \left(f(d_i) + g(d_i)\right),$
$\frac{\partial N}{\partial x} =$
$\sum_{i=0}^{n-1} \varphi_i (x - x_i) \left[f'(d_i) + g'(d_i) + \frac{f(d_i)+g(d_i)}{d_i}\right],$
$\frac{\partial D}{\partial x} =$
$\sum_{i=0}^{n-1} \varphi_i \frac{x - x_i}{d_i} (f'(d_i) + g'(d_i)).$

The result for the GSE model intersection is shown in fig. 7 b).

5 Implementation and Results

Two calculation methods described in section 4 has been implemented in ray tracer supporting GSE models. The GSE ray tracer has been written in the C++ Builder and exist as a standalone application. User interface allows changing the following input data:

- position and direction of the camera,
- light position,
- activation/deactivation of antialiasing,

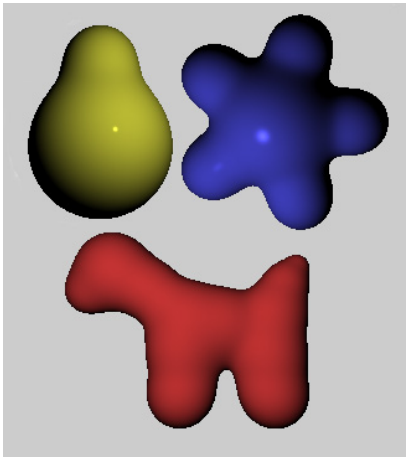

Figure 8: Pictures of GSE models generated by raytracer

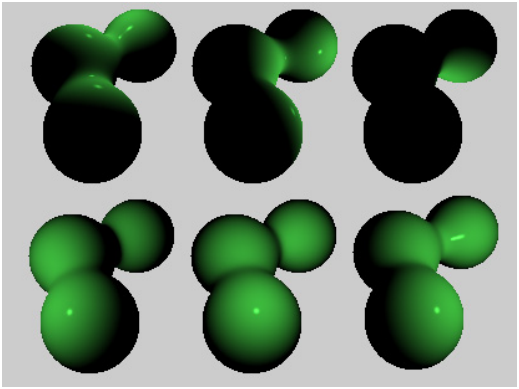

Figure 9: Pictures of the same testing model for six different light position

- activation/deactivation of tracing shadows,
- selection of example scene,
- position of solids (GSE models and spheres) can be seen in 2D projections.

On the output the program generates picture of scene composed of spheres and the GSE models. Sample results are showed in fig. 8.

To test the ray tracer implementation, an example GSE shape (see fig. 9) was rendered in different light condition and the results were evaluated. We did not notice any deformation or other faults. Moreover the correctness of GSE shapes were confirmed by comparison of rendered pictures and GSE intersections (see fig. 10).

Tests showed that finding the intersection of a ray with a GSE model and finding normal vectors give expected results.

Execution time has been measured for pictures of resolution 400x400. The results are presented in tab. 1.

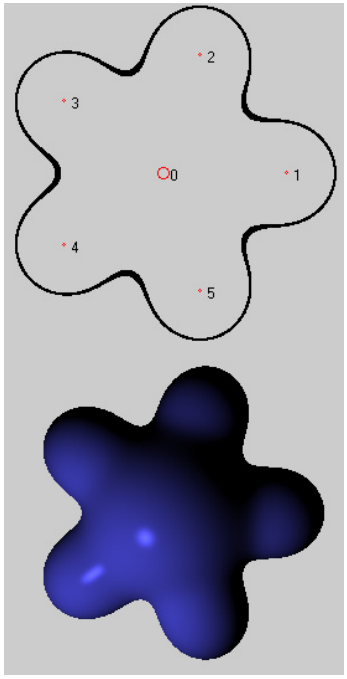

Figure 10: GSE shape intersection and picture generated by the ray tracer

Achieved results let treat calculating methods as quite efficient.

Number of gravity points	Rendering time in [s] AMD K6 300 MHz	Rendering time in [s] AMD Duron 600 MHz
2	9	3
3	11	4
6	24	9
12	53	22

Table 1: Rendering time of GSE models

6 Conclusions and Future Works

Presented results show that GSE method allows getting models which resemble Blinn's blobs. According to expectations, the method lets control the shape of model and modify surface bumps.

The main future work is to ray trace the GSE models with surface bumps ($g \neq 0$). In the fig. 11 draft pictures of such models are presented.

Another future goal is to evaluate the possibilities of efficient calculation of texture coordinates. It would allow to map textures on the GSE shapes. Performance optimization are also taken into consideration.

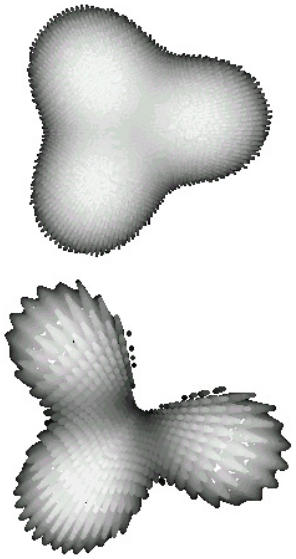

Figure 11: Draft pictures of models with surface bumps

References

[1] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics, Vol.1*, pages 235–256, 1982.

[2] Brian Wyvill Geoff Wyvill, Craig McPheeters. Data structure for soft objects. *Computer Graphics World, May 1993*, pages 227–234, 1993.

[3] John C. Hart. Ray tracing implicit surfaces. 1993.

[4] Teresa Jurlewicz and Zbiegniew Skoczylas. *Algebra liniowa (in Polish)*. Wroclaw, 1999.

[5] D. P. Mitchell. Robust ray intersection with interval arithmetic. pages 68–74, 1990.

[6] R. E. Moore. *Interval Analysis*. Prentice Hall, 1966.

[7] Wanda Ronka-Chmielowiec and Antoni Smoluk. *Metody numeryczne (in Polish)*. 1978.