Proceedings of ALGORITMY 2002
Conference on Scientific Computing, pp. 245–252

POLYGONIZATION BY THE EDGE SPINNING [*]

M. ČERMÁK AND V. SKÁLA[†]

Abstract. This paper presents a new method for polygonization of implicit surfaces. Our method put emphasis on the shape of triangles of the resulting polygonal mesh. The main advantages of the triangulation presented are simplicity and the stable features that can be used for next expanding. The implementation is not complicated and only the standard data structures are used. This algorithm is based on the surface tracking scheme and it is compared with the other algorithms which are based on the similar principle, such as the Marching cubes and the Marching triangles algorithms.

Key words. implicit surfaces, marching triangles, polygonization, triangulation, marching method

AMS subject classifications. 65D18, 68U05

1. Introduction. Implicit surfaces seem to be one of the most appealing concepts for building complex shapes and surfaces. They have become widely used in several applications in computer graphics and visualization.

An implicit surface is mathematically defined as a set of points in space $\boldsymbol{x}$ that satisfy the equation $f(\boldsymbol{x}) = 0$. Thus, visualizing implicit surfaces typically consists in finding the zeroset of f, which may be performed either by polygonizing the surface or by direct raytracing.

There are two different definitions for implicit surfaces. The first one [2], [3] defines an implicit object as $f(\boldsymbol{x}) < 0$ (function $f_1(\boldsymbol{x})$ below) and the second one, F-rep [7], [12], [14] (functional representation, function $f_2(\boldsymbol{x})$) defines it as $f(\boldsymbol{x}) \geq 0$. In our implementation, we use the F-rep definition of implicit objects. The implicit functions described below show the differences between both definitions for the function Sphere.

$$f_1(\mathbf{x}) : x^2 + y^2 + z^2 - r^2 = 0\,,$$

$$f_2(\mathbf{x}) : r^2 - x^2 - y^2 - z^2 = 0\,.$$

2. Data structures. The presented algorithm uses only the standard data structures used in computer graphics. The main data structure is edge that is used as a basic building block for polygonization. We use the standard winding edge and therefore, the resulting polygonal mesh is correct and complete with neighborhood among all triangles generated. The basic data structures used there are:

- edge – winding edge
- active edge – an edge that lies on the triangulated area's border; implemented as an index into winding edge's array
- list of active edges – dynamically allocated list of active edges
- point – if a point lies on an active edge it contains also two pointers to left and right active edge; left and right directions are in active edges orientation

[*]This work was supported by Grant No.: MSM 235200005.

[†] Department of Computer Science, University of West Bohemia, Univerzitni 8, Box 314, 306 14 Plzen (`cermakm@kiv.zcu.cz, skala@kiv.zcu.cz`)

3. Principle of our algorithm. Our algorithm is based on the surface tracking scheme and therefore, there are several limitations. A starting point must be determined and only one separated implicit surface can by polygonized for this first point. Several disjoint surfaces can be polygonized from a starting point for each of them. The whole algorithm consists of following steps:

1. Find a starting point $\boldsymbol{p}_0$.
2. Create a first triangle T_0, see Fig. 1.
3. Include the edges (e_0, e_1, e_2) of the first triangle T_0 into the active edges list.
4. Polygonize the first active edge e from the active edges list.
5. Delete the actual active edge e from the active edges list and include the new generated active edges at the end of the active edges list.
6. Check the distance between the new generated point $\boldsymbol{p}_{new}$ and all the other points which lie on the border of already triangulated area (which lie in all the other active edges).
7. If the active edges list is not empty return to step 4.

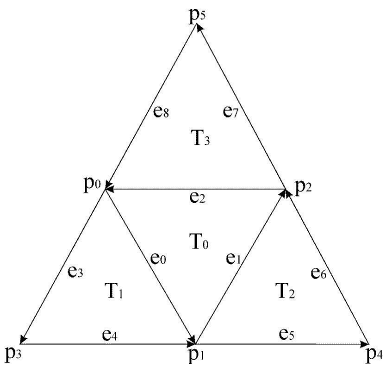

FIG. 1. *The first steps of the algorithm.*

4. Starting point. There are several methods for finding a starting point on an implicit surface. These algorithms can be based on some random search method as in [2] or on more sophisticated approach. In [15], searching in constant direction from an interior of an implicit object is used.

In our approach, we use a simple algorithm for finding a starting point. A starting point is sought from any place in a defined area in the direction of a gradient vector ∇f of an implicit function f. The algorithm looks for a point $\boldsymbol{p}_0$ that satisfies the equation $f(\boldsymbol{p}_0) = 0$.

5. First triangle. The first triangle in polygonization is assumed to lie near a tangent plane of the starting point $\boldsymbol{p}_0$ that is on the implicit surface.

1. Determine the normal vector $\boldsymbol{n} = (n_x, n_y, n_z)$ in the starting point $\boldsymbol{p}_0$, see Fig. 2.; $\boldsymbol{n} = \nabla f / \|\nabla f\|$
2. Determine the tangent vector $\boldsymbol{t}$ as in [5].
   If $(n_x > 0.5)$ or $(n_y > 0.5)$ then $\boldsymbol{t} = (n_y, -n_x, 0)$; else $\boldsymbol{t} = (-n_z, 0, n_x)$.
3. Use the tangent vector $\boldsymbol{t}$ as a fictive active edge and use the edge spinning algorithm (described bellow) for computation coordinates of the second point $\boldsymbol{p}_1$. The pair of points $(\boldsymbol{p}_0, \boldsymbol{p}_1)$ forms the first edge e_0.

4. Polygonize the first edge e_0 with the edge spinning algorithm for getting the third point $\boldsymbol{p}_2$. Points $(\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$ and edges (e_0, e_1, e_2) form the first triangle T_0.

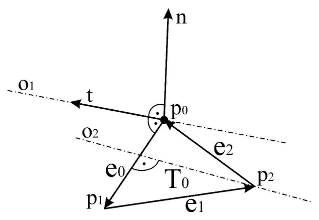

FIG. 2. *First triangle generation.*

6. Edge spinning algorithm. The main goal of this work is a numerical stability of a surface point coordinates' computation for objects which are defined by the implicit function. Differential properties for each implicit function are different in dependence on the modeling techniques [6], [7], [10], [12], [13], [14] and the accurate determination of a position of a surface vertex depends on them. In general, a surface vertex position is searched in direction of a gradient vector of an implicit function f, e.g. in [5]. In many cases, the computing of a gradient of the function f is influenced by a major error. Because of these reasons, in our approach, we have defined these restrictions for finding a new surface point $\boldsymbol{p}_{new}$:

- The new point $\boldsymbol{p}_{new}$ is sought in a constant distance, i.e. on a circle; then each new generated triangle preserves the desired accuracy of polygonization – the average edge's length δ_e. The circle radius is proportional to the δ_e.
- The circle lies in the plane that is defined by the normal vector of triangle T_{old} (see Fig. 3) and axis o of the actual edge e; this guarantees that the new generated triangle is well shaped (isosceles).

Then, the algorithm is:

1. Set the point $\boldsymbol{p}_{new}$ to its initial position; the initial position is on the triangle's T_{old} plane on the other side of the edge e, see Fig. 3. Let the angle the of initial position be $\alpha = 0$.
2. Compute the function values $f(\boldsymbol{p}_{new}) = f(\alpha)$, $f(\boldsymbol{p}'_{new}) = f(\alpha + \Delta\alpha)$ – initial position rotated by the angle $+\Delta\alpha$, $f(p''_{new}) = f(\alpha - \Delta\alpha)$ - initial position rotated by the angle $-\Delta\alpha$; the rotation axis is the edge e.
3. Determine the right direction of rotation; if $|f(\alpha + \Delta\alpha)| < |f(\alpha)|$ then $+\Delta\alpha$ else $-\Delta\alpha$.
4. Let the function values $f_1 = f(\alpha)$ and $f_2 = f(\alpha \pm \Delta\alpha)$; actualize angle $\alpha = \alpha \pm \Delta\alpha$.
5. If $(f_1 \cdot f_2) < 0$ then compute the accurate coordinates of the new point $\boldsymbol{p}_{new}$ by the binary subdivision between the last two points which correspond to function values f_1 and f_2; else return to step 4.
6. Check if both triangles T_{old} and T_{new} do not cross each other; if the angle between these triangles β is greater than β_{lim} (see Fig. 4) then point $\boldsymbol{p}_{new}$ is accepted; else point $\boldsymbol{p}_{new}$ is rejected and return to step 4.

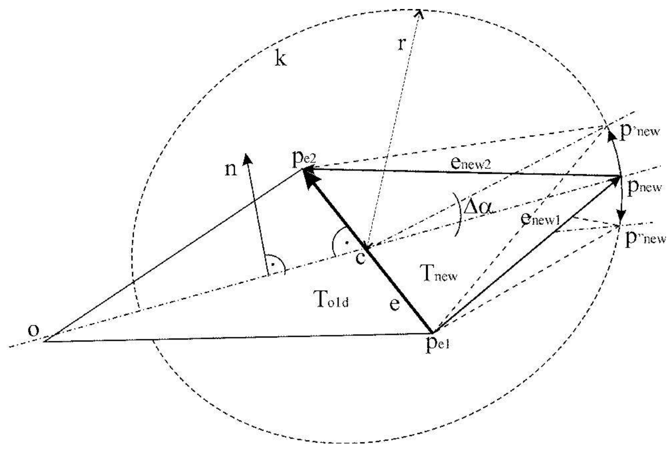

FIG. 3. *The edge spinning algorithm principle.*

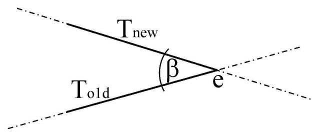

FIG. 4. *The angle between two triangles; the view is in direction of edge's vector e.*

7. Active edge polygonization. Polygonization of an active edge e consists of several steps. At first, the algorithm checks adjacent active edges of the active edge e and determines which case appeared, see Fig. 5.

- If $(\alpha_i < \alpha_{lim\_1})$ then case a); $i = 1, 2$.
- If $(\alpha_2 < \alpha_{lim\_2})$ and $(\|p_{e1} - p_{r\_e2}\| < \delta_{lim\_1})$ then case a); analogically for α_1.
- If $(\alpha_2 > \alpha_{lim\_3})$ and $(\|p_{e1} - p_{r\_e2}\| < \delta_{lim\_2})$ then case b); analogically for α_1.
- else case c)

The relations among limit angles are $\alpha_{lim\_1} < \alpha_{lim\_2} \leq \alpha_{lim\_3}$.

Possible cases which are illustrated in Fig. 5 are:

a) In this case, algorithm creates a new one triangle and includes a new active edge e_{new} to the end of the active edges list.
b) In some situations, the length of certain edges can be shorter then tolerable limit. In this case, algorithm must repair the length of the new edges e_{new1} and e_{new3} to achieve better shapes of next triangles. The axis o_1 (see Fig. 5.) is used as a fictive active edge for the algorithm edge spinning and the new point $\boldsymbol{p}_{new}$ is created as well as two new triangles.
c) In all the other situations, the edge e is polygonized by the standard algorithm edge spinning.

8. Distance test. To preserve the correct topology and the shape of the mesh triangles it is necessary to perform the distance check between the new triangle and a border of already triangulated area. Therefore, each new generated point $\boldsymbol{p}_{new}$ must

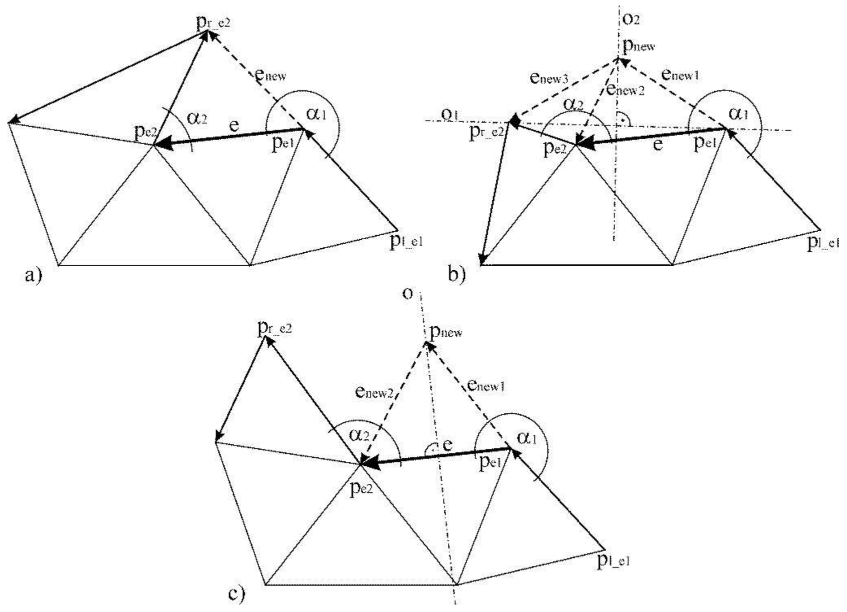

FIG. 5. *The possible cases for polygonization of an active edge e.*

be checked for distance with all the other points which lie in active edges. Let the point $\boldsymbol{p}_{min}$ be the nearest point to this new point $\boldsymbol{p}_{new}$ and distance between both points is $\delta = \|\boldsymbol{p}_{new} - \boldsymbol{p}_{min}\|$. Further, let $\boldsymbol{p}_{min}$ not lie in the active edges which are in the neighborhood of both active edges which contain the point $\boldsymbol{p}_{new}$. Then, there are two cases described in Fig. 6.

a) If $\delta < \delta_{lim\_3}$ then the new point $\boldsymbol{p}_{new}$ is replaced with the point $\boldsymbol{p}_{min}$.
b) If $\delta < \delta_{lim\_4}$ then a new triangle must be created between the new point $\boldsymbol{p}_{new}$ and one of two active edges which contain the point $\boldsymbol{p}_{min}$, i.e. either the triangle $(\boldsymbol{p}_{min}, \boldsymbol{p}_{new}, \boldsymbol{p}_{r\_min})$ or the triangle $(\boldsymbol{p}_{l\_min}, \boldsymbol{p}_{new}, \boldsymbol{p}_{min})$, see Fig. 6b. The decision, which active edge will be used, depends on angles α_1, α_2. The angles $\alpha_i, i = 1, 2$ are in interval $< 0, \pi >$ and therefore, the triangle with the angle α_i that is better approximation of angle 90° is chosen.

The relation between distance limits is $\delta_{lim\_3} < \delta_{lim\_4}$.

Now, the situation described in Fig. 6 a) and b) is similar for both cases. Point $\boldsymbol{p}_{new}$ is contained in four active edges e_1, e_2, e_3, e_4 and a border of already triangulated area intersects itself on it. Solution of the problem will be introduced on case b) and solution for case a) is analogical. Let the four active edges be divided into pairs; the left pair is (e_3, e_2) and the right pair is (e_1, e_4). One of these pairs will be polygonized and the second one will be cached in memory for later use. The solution depends on angles β_1, β_2, see Fig. 6b. If $(\beta_1 < \beta_2)$ then the left pair (e_3, e_2) is polygonized; else the right pair (e_1, e_4) of active edges is polygonized. In both cases, the second pair that is not polygonized is deleted from the list of active edges and the point $\boldsymbol{p}_{new}$ is contained only in one pair of active edges.

In Fig. 6b, the first case is valid $(\beta_1 < \beta_2)$, i.e. the active edges (e_3, e_2) are polygonized in order that depends on angles γ_3, γ_2. If $(\gamma_3 < \gamma_2)$ then the active edge e_3 is polygonized as the first; else the active edge e_2 is polygonized first.

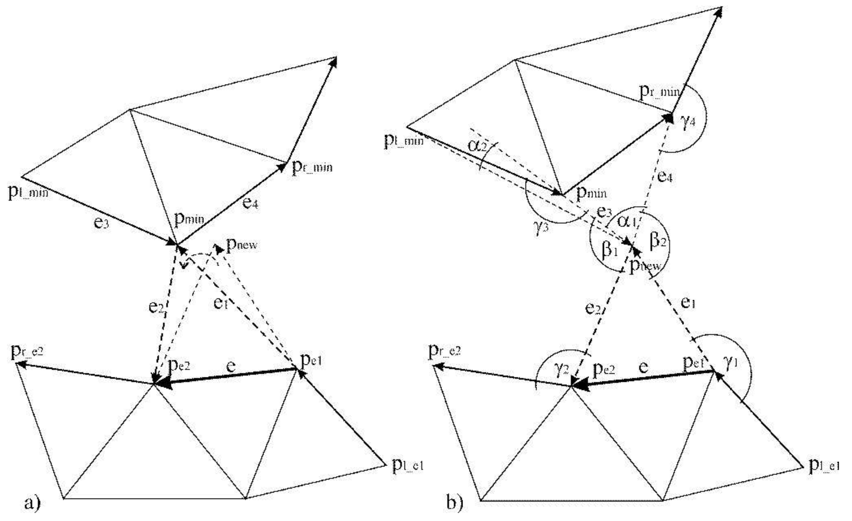

Fig. 6. *The possible cases for distance test.*

Now, the border of the triangulated area does not cross itself in the point $\boldsymbol{p}_{new}$ and the recently polygonized pair of edges is removed from the active edges list. The previously cached pair of edges must be returned into the list of active edges.

9. Experimental results. The Edge spinning algorithm is based on the surface tracking scheme (also known as the continuation scheme). Therefore, we have compared it with other methods based on the same principle – the Marching triangles algorithm (introduced in [5]) and the Marching cubes method (introduced in [2]).

Values from our experiment are shown in Table 1. Variable N represents an average scale of triangle edge's length, i.e. the scene detail grows with N. From measured values follows that the Edge spinning algorithm generates about 15% triangles less then the Marching triangles algorithm and about 24% less then the Marching cubes algorithm.

	N	160	240	400	630	1000
Edge spinning	Triangles	13 368	29 892	81 708	207 290	521 320
	Vertices	6 680	14 942	40 850	103 641	260 656
Marching triangles	Triangles	15 689	35 267	97 943	244 107	613 641
	Vertices	7 840	17 629	48 967	122 049	306 816
Marching cubes	Triangles	17 568	39 520	109 608	271 344	684 016
	Vertices	8 772	19 756	54 800	135 668	342 004

Table 1
A number of triangles and vertices generated for each type of polygonization algorithm.

This experiment was made on the Genus object with the implicit function:

$$r_z^4 z^2 - (1 - (x/r_x) - (y/r_y))\left((x - x_1)^2 + y^2 - r_1^2\right)\left((x + x_1)^2 + y^2 - r_1^2\right) = 0$$

where $r_x = 6; r_y = 3.5; r_z = 4; r_1 = 1.2; x_1 = 3.9$.

The triangular meshes generated by the discussed algorithms are shown in Fig. 7. More detailed comparison of the quality of the triangle mesh generated is shown in histogram, see Fig. 8. The histogram shows the percentage ratio of the angles incidence. This experiment demonstrates that the Edge spinning algorithm has the highest number of angles in triangulation in interval $< 50^o, 70^o >$. The Marching triangles method also generates well-shaped triangles and the Marching cubes algorithm generates poor polygonal mesh.

The presented results were verified on many nontrivial implicit surfaces.

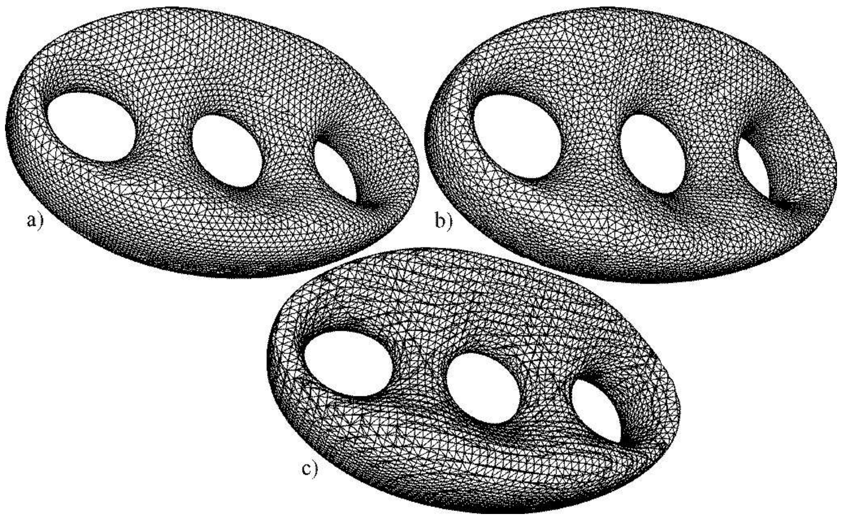

FIG. 7. *Object Genus generated for N=160 (see Table 1) by a) the Edge spinning, b) the Marching triangles, c) the Marching cubes algorithm.*

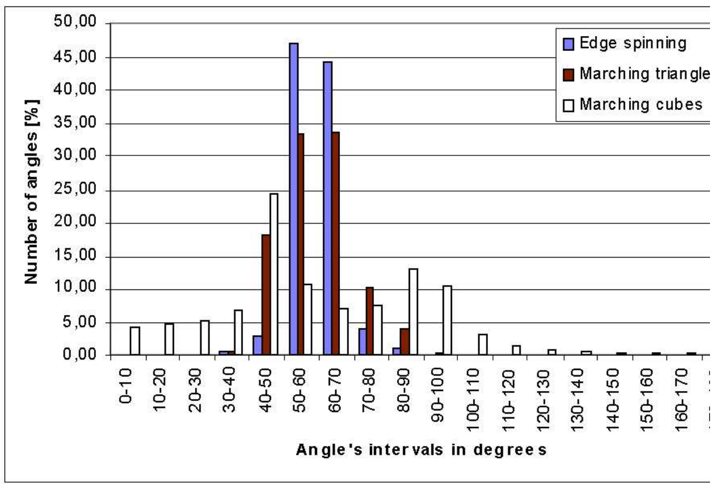

FIG. 8. *Histogram of triangles shape quality for the Edge spinning, the Marching triangles and the Marching cubes algorithms. Generated for N=1000, see Table 1.*

10. Conclusion. In this paper, we have presented the new principle for polygonization of implicit surfaces. The algorithm marches over the object's surface and computes the accurate coordinates of new points by spinning the edges of already generated triangles. Presented algorithm can polygonize implicit surfaces which comply C^1 continuity. In future work, we want to modify the current algorithm for implicit functions with only C^0 continuity. We suppose that our defined restrictions, for polygonization of an active edge, are the right way. In next research, we will work on adapting of the Edge spinning algorithm to local curvature of an implicit surface, [1], [9].

Acknowledgements.. The authors of this paper would like to thank all those who contributed to development of this new approach, especially to colleagues MSc. and PhD. students at the University of West Bohemia in Plzen. Presented project was implemented as a part of the MVE (Modular Visualization Environment), [8], [11].

REFERENCES

[1] Akkouche, S., Galin, E.: *Adaptive Implicit Surface Polygonization using Marching Triangles*, Computer Graphic Forum, **20** (2) (2001), 67–80.

[2] Bloomenthal, J.: Graphics Gems IV, Academic Press, 1994.

[3] Bloomenthal, J.: Skeletal Design of Natural Forms, Ph.D. Thesis, 1995.

[4] Bloomenthal, J., Bajaj, Ch., Blinn, J., Cani-Gascuel, MP., Rockwood, A., Wyvill, B., Wyvill, G.: Introduction to implicit surfaces, Morgan Kaufmann, 1997.

[5] Hartmann, E.: *A marching method for the triangulation of surfaces*, The Visual Computer **14** (1998), 95–108.

[6] Hilton, A., Stoddart, A.J., Illingworth, J., Windeatt, T.: *Marching Triangles: Range Image Fusion for Complex Object Modelling*, Int. Conf. on Image Processing, 1996.

[7] "Hyperfun: Language for F-Rep Geometric Modeling", http://cis.k.hosei.ac.jp/F̃-rep/

[8] MVE – Modular Visualization Environment project, http://herakles.zcu.cz/research.php, University of West Bohemia in Plzen, Czech Republic, 2001.

[9] Ohraje, Y., Belyaev, A., Pasko, A.: *Dynamic meshes for accurate polygonization of implicit surfaces with sharp features*, Shape Modeling International 2001, IEEE, 74–81.

[10] Pasko, A., Adzhiev, V., Karakov, M., Savchenko,V.: *Hybrid system architecture for volume modeling*, Computer & Graphics **24** (2000), 67–68.

[11] Rousal, M., Skala, V.: *Modular Visualization Environment - MVE*, Int. Conf. ECI 2000, Herlany, Slovakia, 245–250 (ISBN 80-88922-25-9).

[12] Rvachov, A.M.: *Definition of Rfunctions*, http://www.mit.edu/∼maratr/rvachev/p1.htm

[13] Shapiro, V., Tsukanov, I.: Implicit Functions with Guaranteed Differential Properties, Solid Modeling, Ann Arbor, Michigan, 1999.

[14] Uhlir, K., Skala, V.: *Interactive system for generating and modeling implicit functions*, submitted for publication, 2002.

[15] Triquet, F., Meseure, F., Chaillou, Ch.: *Fast Polygonization of Implicit Surfaces*, WSCG'2001 Int.Conf., p. 162, University of West Bohemia in Pilsen, 2001.