YU ISSN 0005-1144
ATKAAF 31 (1—2), 25—30 (1990)

Václav Skala

General Conics Clipping – Problem Solution

UDK 007.001.33
IFAC IA 1.2;2.8

Original scientific paper

New algorithms for 2D quadratic arcs clipping against convex and non-convex windows are presented. Algorithms do not use parametric equations for the arcs description. The only a square root function is needed. The algorithms require information how edges of the given window are oriented. The design, implementation and verification is the first step toward to the quadratic arcs usage in computer graphics as new basic primitives. The presented algorithms have not been published in a literature known to the author.

Key words: Clipping, Quadratic Arcs, Algorithms

1. INTRODUCTION

Clipping is a very important part of all graphics packages. There are many efficient algorithms as [1], [6], [7], [10] for clipping lines against convex windows or for clipping lines against a window with holes and with the non-linear boundaries, see [13], [14]. Unfortunately no one known algorithm deals with a problem how to clip circles or ellipses against a window. This problem is a fundamental one if we want to introduce quadratic arcs as a basic primitive for 2D graphics packages. The below described algorithms enable to clip general quadratic curves or arcs against convex or non-convex window.

2. CLIPPING BY A CONVEX AREA

Provided a convex area is given by its vertices in the clockwise order and a oriented circle given by its center $\mathbf{x}_w$ and radius r. We want to find those parts of the given circle which lie inside of the given convex window. For easier understanding the following notation will be used:

$\mathbf{x}_w$ circle center $\qquad$ $\mathbf{x}_i$ polygon vertex

$\mathbf{x}_k$ instead of $\mathbf{x}_{i+1}$; the $+$ is meant as modulo n addition

$\mathbf{s}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$ $\qquad$ $\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_i$

${}^i\mathbf{s}_w = \mathbf{x}_w - \mathbf{x}_i$

$\mathbf{t} = [y_w - y, x - x_w]^T$ tangent vector at the point (x, y)

To solve this problem it is necessary to find intersection points of the circle and line $w(q)$ on which the window border lies, i. e. to solve the following equations:

$$\mathbf{x}(q) = \mathbf{x}_i + (\mathbf{x}_{i+1} - \mathbf{x}_i) \cdot q$$

$$(x - x_w)^2 + (y - y_w)^2 - r^2 = 0 .$$

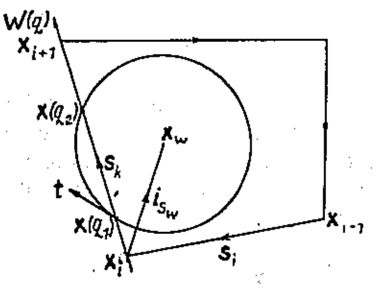


Figure 2.1.

Solving these equations with regard to the variable q the quadratic equation

$$a q^2 + b q + c = 0$$

will be obtained, where:

$$a = |\mathbf{s}_k|^2 \quad b = -2\,{}^i\mathbf{s}_w^T \cdot \mathbf{s}_k \quad c = |{}^i\mathbf{s}_w|^2 - r^2$$

In the case that the line $w(q)$ intersects or touches the given circle two solutions are generally obtained that are not necessarily different:

$$q_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The obtained values q_1, q_2 must be ordered so that $q_1 \leq q_2$. For the further processing only points for $q \in \langle 0, 1)$ will be considered. Of course some possible situations must be distinguished, see fig. 2.2. For a general case, when $q_1 \neq q_2$, the tangent vector $\mathbf{t}_1$ always points out of the given window while the tangent vector $\mathbf{t}_2$ always points into the given window, see fig. 2.2. cases a and d. Therefore the circular arc $\mathbf{x}(q_1)\,\mathbf{x}(q_2)$ cannot be considered for further processing. The cases b and c from fig. 2.2. are a little bit more complicated because the tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$ are equal. It means that it is necessary to introduce some special attri-

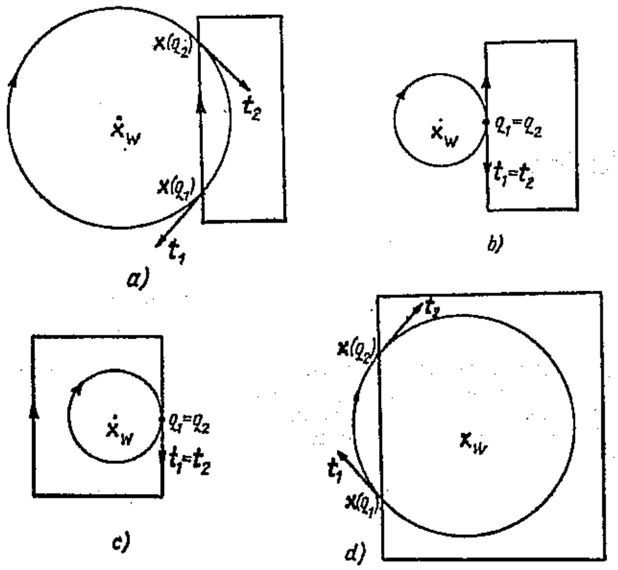


Figure 2.2.

butes for the case b while case c must be handled as no intersection points with the given edge have been found. In the case b only one point $\mathbf{x}(q_1)$ will be drawn. But there are still some special cases to be solved, see fig. 2.3.

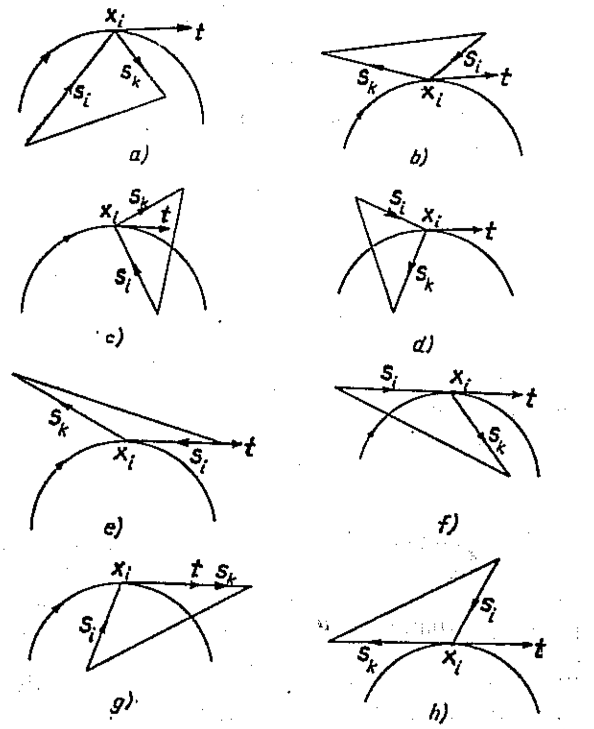


Figure 2.3.

It can be observed that it is necessary to introduce the additional attribute that would determine exactly whether the tangent vector $\mathbf{t}$ points into or out from the clipping window.

First of all it is necessary to distinguish between cases a, b and c, d in fig. 2.3., i. e. to distinguish between »touch« and »pass« types. Now let us examine the result of the cross product of the $\mathbf{s}_i$, $\mathbf{s}_k$ and $\mathbf{t}$ vectors, see fig. 2.3., in order to distinguish some cases, see table 2.1.

Table 2.1.

$[\mathbf{t} \times \mathbf{s}_i]_z$	$[\mathbf{t} \times \mathbf{s}_k]_z$	case	type	sequence
> 0	< 0	a	touch	$\mathbf{x}_i\ \mathbf{x}_i$ $+-$
< 0	> 0	b	touch	$\mathbf{x}_i\ \mathbf{x}_i$ $+-$
> 0	> 0	c	pass	$\mathbf{x}_i$ $+$
< 0	< 0	d	pass	$\mathbf{x}_i$ $-$
$= 0$	> 0	e	touch	$\mathbf{x}_i\ \mathbf{x}_i$ $+-$
$= 0$	< 0	f	pass	$\mathbf{x}_i$ $+$
> 0	$= 0$	g	pass	$\mathbf{x}_i$ $-$
< 0	$= 0$	h	touch	$\mathbf{x}_i\ \mathbf{x}_i$ $+-$
$= 0$	$= 0$	special cases	touch	not allowed

where $+-$ denotes the sign of the cross product z coordinate

It is obvious that when the both cross products are equal to zero it is necessary to distinguish some very special cases. Therefore it is necessary to use the direction of the tangent vector for determining the attribute.

The whole algorithm can be described by a sequence in PASCAL-type style, see algorithm 2.1.

$k := 0;\ i := n - 1;$
flag: = **true**; {circle is not intersected}
while $k < n$ **do**
begin
  $\mathbf{s}_k := \mathbf{x}_k - \mathbf{x}_i$; {needed for intersection solution}
  if COMPUTE VALUES (q_1, q_2)
  then {intersection points exist and $q_1 \leq q_2$}
  begin flag: = **false**;
    if $q_1 = 0$ **then** {pass / touch type}
    begin
      $\mathbf{s}_i := \mathbf{x}_i - \mathbf{x}_{i-1}$; $\mathbf{t} := [y_w - y_i, x_i - x_w]^T$;
      $a := [\mathbf{t} \times \mathbf{s}_i]_z$; $b := [\mathbf{t} \times \mathbf{s}_k]_z$;
      if $(a > 0)$ **or** $(b \geq 0)$ **then** GENERATE $(\mathbf{x}_i, '+')$;
      if $(a \leq 0)$ **or** $(b < 0)$ **then** GENERATE $(\mathbf{x}_i, '-')$;
      if $q_2 \in (0, 1)$ **then**
        begin $\mathbf{t} := [y_w - y(q_2), x(q_2) - x_w]^T$;
          GENERATE $(\mathbf{x}(q_2), \text{sign}([\mathbf{t} \times \mathbf{s}_k]_z))$
          {for circle, ellipse the attribute is always '+'}
        end
    end

```
else
  if q1 ≠ q2 then
  begin
    for j: = 1 to 2 do
      if qj ∈ (0, 1) then
      begin t: = [yw − y (qj), x (qj) − xw]^T;
            GENERATE (x (qj), sign ([t × sk]z))
      end
  end
  else
   {if q1 = q2 then}
   begin t: = [yw − y (q1), x (q1) − xw]^T;
     if t · sk < 0 then
     begin
      GENERATE (x (q1), '+');
        GENERATE (x (q1), '−')
     end
   end;
 i: = k; k: = k + 1
end {while};
```

Algorithm 2.1.

As a result of the algorithm 2.1. sequences shown in table 2.1. are obtained. Now it is necessary to draw appropriate arcs that are inside of the given window. This process can be described by the algorithm 2.2.

```
m: = No of intersections;
if m ≠ 0 then
begin i: = 2;
  while i < m − 1 do
  begin if xi = xi+1 then PLOT (xi)
       else if attr (xi) = '+' then DRAW ARC (xi,
                                   xi+1, r)
                                   else DRAW ARC (xi−1
                                   xi, r);
       i: = i + 2
  end
end {while}
else {it is necessary to distinguish cases when the}
     {circle is totally inside or outside of the given
     window}
  if flag then
  begin flag: = true;
       i: = n − 1; k: = 0;
       while (k < n) and flag do
       begin sk: = xk − xi;  isw: = xw − xi;
           if [sk × isw]z > 0 then flag: = false;
           i: = k; k: = k + 1
       end;
       if flag then DRAW CIRCLE (xw, r)
  end
```

Algorithm 2.2.

The shown algorithm deals with a principal solution and does not particularly care of the arithmetic precision, see [8]. In some cases special criteria must be used in order to respect a limited precision.

The presented algorithm is capable to handle all kinds of quadratic arcs. In the general case the quadratic arc must be given as:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} = 0$$

so that

$$f(\mathbf{x}_w) = \mathbf{x}_w^T \mathbf{A} \mathbf{x}_w < 0$$

where $\mathbf{x}_w$ is the center. The tangent vector $\mathbf{t}$ must be computed as:

$$\mathbf{t} = [f_y, -f_x]^T$$

where f_x, f_y are partial derivations of the function f and the matrix $\mathbf{A}$ represents a general quadratic curve.

3. NON-CONVEX WINDOW CLIPPING

So far presented algorithms have solved the quadratic arc clipping by the convex polygon. But some types of applications do require clipping by non-convex window. Of course, it is possible to split the given non-convex polygon into a set of convex polygons, e. g. [11].

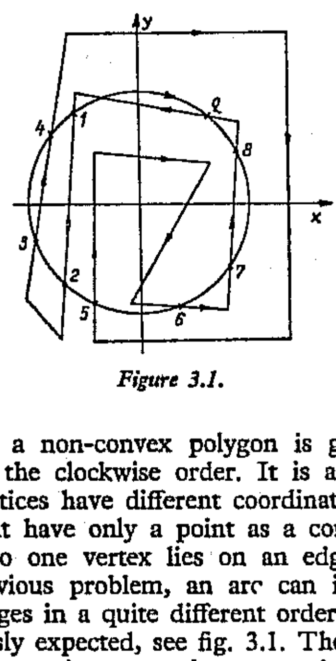


Figure 3.1.

Provided a non-convex polygon is given by its vertices in the clockwise order. It is also assumed that all vertices have different coordinates, that two edges might have only a point as a common point and that no one vertex lies on an edge. Contrary to the previous problem, an arc can intersect the polygon edges in a quite different order than would be previously expected, see fig. 3.1. Therefore some additional operations must be expected. A sequence of points together with their attributes

$$\underset{-}{\mathbf{x}_1}, \ldots, \underset{+}{\mathbf{x}_{10}}$$

is not the sequence that we need for the further processing, because only the following arcs should be drawn:

$$\begin{matrix} \mathbf{x}_4\mathbf{x}_1 & \mathbf{x}_{10}\mathbf{x}_9 & \mathbf{x}_8\mathbf{x}_7 & \mathbf{x}_6\mathbf{x}_5 & \mathbf{x}_2\mathbf{x}_3 \\ +\,- & +\,- & +\,- & +\,- & +\,- \end{matrix}$$

It is obvious that some kind of sort process must be employed in order to get the shown sequence. It is necessary to find a convenient criterion for sorting. The obtained points must be split into two sets according to y value of the given points, i. e.:

$$\Omega_1 = \{\mathbf{x}_j\} \quad y_j \geq y_w \quad \text{for all } j$$

$$\Omega_2 = \{\mathbf{x}_j\} \quad y_j < y_w \quad \text{for all } j$$

In the case shown in fig. 3.1. two sets Ω_1 and Ω_2 are obtained so that:

$$\Omega_1 = \{\underset{-}{\mathbf{x}_1}, \underset{+}{\mathbf{x}_4}, \underset{+}{\mathbf{x}_8}, \underset{-}{\mathbf{x}_9}, \underset{+}{\mathbf{x}_{10}}\}$$

$$\Omega_2 = \{\underset{+}{\mathbf{x}_2}, \underset{-}{\mathbf{x}_3}, \underset{-}{\mathbf{x}_5}, \underset{+}{\mathbf{x}_6}, \underset{-}{\mathbf{x}_7}\}$$

Both sets Ω_1 and Ω_2 must be sorted according to x value of the given points. The set Ω_2 must be sorted according to the descending x value of the given points. Then the ordered sets are:

$$\Omega_1 = \{\underset{+}{\mathbf{x}_4}, \underset{-}{\mathbf{x}_1}, \underset{+}{\mathbf{x}_{10}}, \underset{-}{\mathbf{x}_9}, \underset{+}{\mathbf{x}_8}\}$$

$$\Omega_2 = \{\underset{-}{\mathbf{x}_7}, \underset{+}{\mathbf{x}_6}, \underset{-}{\mathbf{x}_5}, \underset{+}{\mathbf{x}_2}, \underset{-}{\mathbf{x}_3}\}$$

If a new set Ω is created as:

$$\Omega = \Omega_1 \ \mathbf{cont} \ \Omega_2$$

where **cont** is the concatenation operator, i. e.:

$$\Omega = \{\underset{+}{\mathbf{x}_4}, \underset{-}{\mathbf{x}_1}, \underset{+}{\mathbf{x}_{10}}, \underset{-}{\mathbf{x}_9}, \underset{+}{\mathbf{x}_8}, \underset{-}{\mathbf{x}_7}, \underset{+}{\mathbf{x}_6}, \underset{-}{\mathbf{x}_5}, \underset{+}{\mathbf{x}_2}, \underset{-}{\mathbf{x}_3}\}$$

then the required parts of the given quadratic curves are obtained.

If ellipses are considered the situation is a little bit more complicated. The above shown criterion for splitting the intersection points into two sets Ω_1, Ω_2 is not the right one, see fig. 3.2., because the arc for $y \geq y_w$ is not generally a function of x coordinate It means that two points $\mathbf{x}_a$, $\mathbf{x}_b$ must be found so that the tangent vectors of the given ellipse are collinear with y axis in these points. The points can be found as a solution of the quadratic equations:

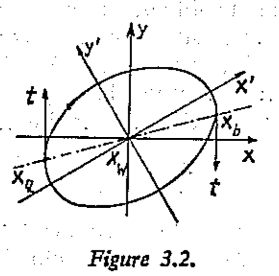


Figure 3.2.

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = 0 \text{ and } \frac{\partial}{\partial y} \mathbf{x}^T \mathbf{A} \mathbf{x} = 0$$

Because the $\mathbf{x}_a$, $\mathbf{x}_b$ points are obtained it is possible to split the intersection points of the ellipse and the given window into two sets Ω_1 and Ω_2 so that:

$$\Omega_1 = \{\mathbf{x}_j\} \quad [(\mathbf{x}_b - \mathbf{x}_a) \times (\mathbf{x}_j - \mathbf{x}_a)]_z \geq 0 \quad \text{for all } j$$

$$\Omega_2 = \{\mathbf{x}_j\} \quad [(\mathbf{x}_b - \mathbf{x}_a) \times (\mathbf{x}_j - \mathbf{x}_a)]_z < 0 \quad \text{for all } j$$

The sets Ω_1 and Ω_2 must be sorted again and a new set Ω must be created in the same way.

If a parabolic arc is considered the rules can be derived in a similar way. But because only $\mathbf{x}_a$ point can be obtained it is necessary to find a different criterion how to split the obtained points. The points $\mathbf{x}_v$, $\mathbf{x}_f$, $\mathbf{x}_a$ can be determined easily for the parabolic arc. Therefore the Ω_1 and Ω_2 sets can be defined as follows:

$$\Omega_1 = \{\mathbf{x}_j\} \quad [(\mathbf{x}_f - \mathbf{x}_v) \times (\mathbf{x}_j - \mathbf{x}_a)]_z \geq 0 \quad \text{for all } j$$

$$\Omega_2 = \{\mathbf{x}_j\} \quad [(\mathbf{x}_f - \mathbf{x}_v) \times (\mathbf{x}_j - \mathbf{x}_a)]_z < 0 \quad \text{for all } j$$

If a hyperbolic arc is considered the main problem is to find a convenient criterion for splitting the intersection points into two sets Ω_1 and Ω_2 and the proper criterion for the ordering these sets. In this case it is necessary to find a vector $\mathbf{t}_0$ as:

$$\mathbf{t}_0 = \mathbf{x}_{f2} - \mathbf{x}_{f1}$$

and a vector $\mathbf{t}$ orthogonal to the vector $\mathbf{t}_0$ as:

$$\mathbf{t} = [t_y, -t_x]^T$$

Now the sets Ω_1 and Ω_2 can be defined as:

$$\Omega_1 = \{\mathbf{x}_j\} \quad [\mathbf{t} \times (\mathbf{x}_j - \mathbf{x}_w)]_z \geq 0 \quad \text{for all } j$$

$$\Omega_2 = \{\mathbf{x}_j\} \quad [\mathbf{t} \times (\mathbf{x}_j - \mathbf{x}_w)]_z < 0 \quad \text{for all } j$$

The sets Ω_1 and Ω_2 must be ordered and the ordering according to x coordinate of the given points is not the right one.

The sets Ω_1 and Ω_2 must be reordered with regard to x or y coordinate according to the rotation angle α defined as:

$$2\alpha = \text{arccotg}\,(\xi)$$

where $\xi = \frac{a_{11} - a_{22}}{2\, a_{12}}$ a_{ij} are elements of the matrix **A**. It means that if

$\xi \geq 0$ then the y coordinate must be used for sorting

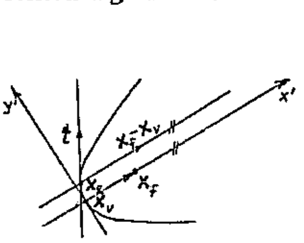
$\xi < 0$ then the x coordinate must be used for sorting

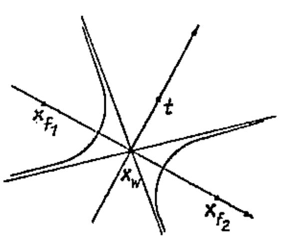
To display the obtained arc segments a similar algorithm to the algorithm 2.2. can be used.

For the non-convex window clipping a special test »point in polygon« must be employed if no intersection point is found in order to distinguish between cases when ellipse or circle lie totally inside or outside of the given window (in the other cases the whole curve must intersect window boundary if they are inside of the window).

4. CONCLUSION

The new algorithms for clipping quadratic curves and their parts against convex and non-convex window have been presented. The algoritmhs can be modified in order to handle different polygon or curves orientations and more general cases, too. The algorithms use only the implicit function definition for quadratic curves and only a square root function is needed. The algorithms do not solve the problem of the limited arithmetic precision because this problem was solved by [8] and the shown results can be applied straightforwardly with the presented algorithms.





5. ACKNOWLEDGMENT

The author would like to express his thanks to Mr. K. Karlovec for the help, interest and comments, to the students of Computer Graphics course that stimulated this work, to the members of Computer Graphics and CAD System Group for their invaluable comments, help and understanding, to Teplotechna Comp. (Czechoslovakia) for the support that enabled to finish this work successfully.

6. REFERENCES

[1] Cyrus, M., Beck, J.: **Generalized Two- and Three Dimensional Clipping.** Computers & Graphics, Vol. 3, No. 1, pp. 23—28, 1979

[2] Earnshaw, R. A., Wyvill, B. (Ed.): **New Advances in Computer Graphics.** Proceedings Computer Graphics International '89, Leeds, Springer-Verlag, 1989

[3] Foley, J. D., van Dam A.: **Fundamentals of Interactive Computer Graphics.** Addison Wesley, Reading, Mass., 1982

[4] Hansmann, W., Hopgood, F. R. A., Strasser, W. (Ed.): **Conference Proceedings EUROGRAPHICS '80.** Hamburg 1989, North Holland, 1989

[5] Kilgour, A. C.: **Unifying Vector and Polygon Algorithms for Scan Conversion and Clipping.** Report CSC 87/R7, Computer Sci. Dept., Univ. of Glasgow, 1987

[6] Liang, Y. D., Barsky, B. A.: **An Analysis and Algorithms for Polygon Clipping.** CACM 26, No. 11, pp. 868—876, 1984

[7] Liang, Y. D., Barsky, B. A.: **A New Concept and Method for Line Clipping.** ACM Trans. on Graphics, Vol. 3, No. 1, pp. 1—22, 1984

[8] Middleditch, A. E., Stacay, T. W., Tor, S. B.: **Intersection Algorithms for Lines and Circles.** ACM Trans. on Graphics, Vol. 8, No. 1, pp. 25—40, 1989

[9] Newman, W. M., Sproull, R. F.: **Principles of Interactive Computer Graphics.** 2nd ed., McGraw Hill, New York, 1979

[10] Nicholl, T. M., Lee, D. T., Nicoll, R. A.: **An Efficient New Algorthm for 2D Line Clipping: Its Development and analysis.** ACM Computer Graphics, Vol. 21, No. 4, pp. 253—262, 1987

[11] Rogers, D. F.: **Procedural Elements for Computer Graphics.** pp. 151—152, McGraw Hill, 1985

[12] Skala, V. (1989) **Algorithms for 2D Line Clipping.** in [2], pp. 121—128

[13] Skala, V.: **Algorithms for 2D Line Clipping.** in [4], pp. 355—366, 1989

[14] Van Vyk C. J.: **Clipping to the Boundary of a Circular Arcs Polygon.** Computer Vision, Graphics and Image Processing, Vol. 25, No. 3, 1984

Odrezivanje stožnica — rješenje problema. Prezentiran je novi postupak odrezivanja lukova krivulja drugog reda za konveksne i nekonveksne prozore. Postupak ne koristi parametarske jednadžbe za opis lukova. Potrebna je jedino funkcija drugog korjena. Postupak zahtjeva informaciju o orjentaciji bridova datog prozora. Oblikovanje, primjena i provjera su prvi korak u korištenju lukova krivulja drugog reda kao novih osnovnih primitiva u računarskoj grafici. Pokazani postupak dosad nije bio publiciran u literaturi koliko je autoru poznato

Ključne riječi: odrezivanje, lukovi krivulja drugog reda, postupci.

AUTHOR'S ADDRESS:

Doc. Ing. Václav SKALA, CSc.,
Dept. of Informatics and Computer Science,
Institute of Technology,
Nejedlého sady 14, Box 314
306 14 Plzeň
Czechoslovakia

Received: 1990-3-20.