NON-ORTHOGONAL CO-ORDINATES IN COMPUTER GRAPHICS[i]

Vaclav Skala [ii]

Department of Informatics and Computer Science
University of West Bohemia, Univerzitní 8, Box 314
306 14 Plzen, Czech Republic
skala@kiv.zcu.cz http://iason.zcu.cz/~skala

ABSTRACT

There are many applications that are not naturally based on the orthogonal co-ordinate systems, like ultrasound imaging, astronomy, mechanical computations etc. In many cases it is necessary to transform the problem formulation to the orthogonal co-ordinate system, where the problem is solved, transform the solution back and visualize the solution. The polar, spherical or cylindrical co-ordinate systems are natural to many problems and sometime also used for computations. It will be shown that the polar systems in E^2 can be used for the visualization pipeline and some properties of such system will be discussed including geometric transformation in the polar system.

Keywords: algorithm complexity, computer graphics, geometric transformation, non-orthogonal co-ordinate system, point-in-polygon, line clipping, convex polygon.

1 INTRODUCTION

There are many applications where the polar co-ordinate system is used for finding a solution of the given problem, especially in technical problems solutions (FEM, flow computation, radiation, radar and sonar applications), medical imaging (ultrasound imaging) etc.

It is a usual practice that all data from those applications are transformed to the standard orthogonal co-ordinate system, where all data are processed. The data are then displayed directly or transform back to the original co-ordinate system. Nevertheless it is well known that representation of a point in E^2 is different from a line representation and therefore the processing pipeline have to respect this fact. The polar system offers some possibilities how to handle graphical information in an unambiguous way and hopefully can lead to new understanding of some fundamental algorithms and developing new more effective methods.

2 ORTHOGONAL CO-ORDINATE SYSTEM

The orthogonal co-ordinate system and point representations are well known and used nearly exclusively. In the majority of applications the homogeneous co-ordinates are used and a point is represented as

$$[\, x\, ,\, y\, ,\, w\,]^T$$

where: w is the homogeneous co-ordinate.

This representation enables us represent all geometric transformations by using matrix multiplication. On the other side a line p can be represented by an implicit function as

$$a\, x + b\, y + c = 0$$

or by explicit functions as

$$y = k\, x + q \qquad \text{if} \quad k \neq \infty$$

or as

$$x = m\, y + p \qquad \text{if} \quad m \neq \infty$$

or parametrically as

$$\boldsymbol{x}(t) = \boldsymbol{x}_A + \boldsymbol{s}\, t \qquad t \in (-\infty\, ,\, +\infty)$$

where: $\mathbf{x}_A$ is a point on the line p and $\boldsymbol{s}$ is directional vector of this line.

The problem is when we want to manipulate with the lines. How we can handle them? Let is consider a point $\mathbf{x}$ and its co-ordinates in the polar co-ordinate system

$$\boldsymbol{x} = [\, \rho\, ,\, \varphi\,]^T$$

It can be seen that there the point $\boldsymbol{x}$ defines also unambiguously a line p that is orthogonal to the line defined by this point and by the origin of the polar co-ordinate system, see fig.1.

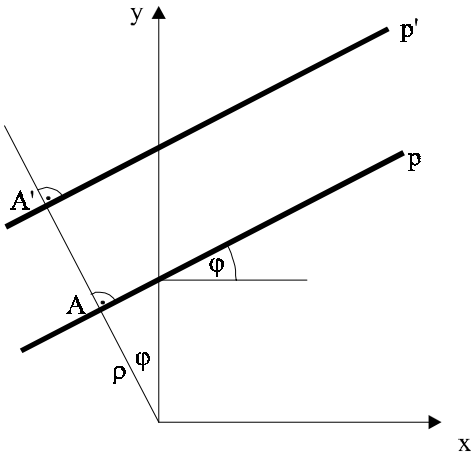

Figure 1

It is well known, that the geometric transformations can be represented by matrices generally, i.e. for the translation we get:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{T}\,(a\,,b\,)\,\boldsymbol{x}$

and for the rotation we get

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin-\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{R}\,(\varphi\,)\,\boldsymbol{x}$

But what does these geometric transformations mean for applications in the field of radar signal processing where the polar representation is native one?

Of course there are an unambiguous transformations to the polar and from the polar to the Cartesian co-ordinate systems, but it is not a natural co-ordinate system to the problem. Therefore the question whether a similar operations for geometric transformations can be defined for polar co-ordinate system can be derived and what would be properties and possible usage.

3 POLAR CO-ORDINATE SYSTEM

It is well known that the transform from the Cartesian to the polar co-ordinate system can be defined as

$$x = \rho\ cos\ \varphi$$
$$y = \rho\ sin\ \varphi$$

where $\varphi \in < 0\ ,\ 2\ \pi\ )$.

This transformation is often used for a circle or an ellipse (with small modification) generation. It is necessary to have an efficiency of all operations in mind in all computer graphics and data visualization system as the volume of processed data is very high.

Therefore we must avoid all *cos* and *sin* function computations as much as possible. Also we will require that all geometric transformation will be possible to represent by matrices and the composition of the geometric transformation should be represented by matrix multiplication.

Let us suppose that the point $\boldsymbol{x} = (\ x\ ,\ y\ )$ can be represented by a vector in "homogeneous polar space" as

$$[\, \rho\ ,\ cos\ \varphi\ ,\ sin\ \varphi\,]^T$$

so that

$$x = \rho\ cos\ \varphi$$
$$y = \rho\ sin\ \varphi$$

and we will avoid divisions needed in the "cartesian homogeneous co-ordinates", where

$$X = x\ /w \qquad \text{and} \qquad Y = y\ /w$$

Of course that there are different representations for the polar co-ordinate system but there is a hope that this is the effective one.

This representation enables us to represent not only the given point but also the given line in the polar co-ordinate system unambiguously.

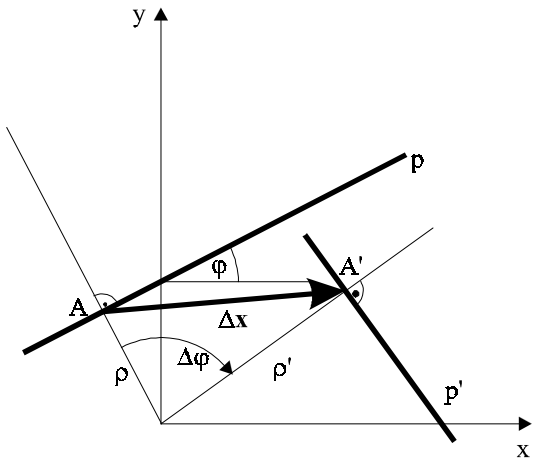

Figure 2

It can be seen that the point A is unambiguously represented by the vector $[\ \rho\ ,\ cos\ \varphi\ ,\ sin\ \varphi\ ]^T$ as well as the line *p*. The line *p* is defined as a line passing the point *A* that is orthogonal to the line *OA*, where *O* stands for the origin of the polar co-ordinate system.

In the polar co-ordinate system the translation matrix is defined as

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi' \\ \sin\varphi' \end{bmatrix} = \begin{bmatrix} 1 & r & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{T}(r)\ \boldsymbol{x}$

where: *r* is the translation parameter. There is also an alternative definition that enables to move points λ times, e.g. relatively to the original distance from the origin.

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi' \\ \sin\varphi' \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{T}_R(\lambda)\ \boldsymbol{x}$

The rotation operation is defined in a similar way as the rotation in the Cartesian co-ordinate system. Let us suppose that we want to rotate the point about the origin with an angle $\Delta\varphi$. It is well known that

$$sin(\varphi+\Delta\varphi) = sin\varphi\ cos\ \Delta\varphi + cos\varphi\ sin\ \Delta\varphi$$

$$cos(\varphi+\Delta\varphi) = cos\varphi\ cos\Delta\varphi - sin\varphi sin\Delta\varphi$$

These formulas enable us to avoid *cos* and *sin* functions computations for each graphics primitive as we need to compute only $cos\Delta\varphi$ and $sin\Delta\varphi$ functions once for the transformation matrix. Considering this entity we can write the matrix for the rotation in form

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\Delta\varphi & -\sin\Delta\varphi \\ 0 & 0 & \sin\Delta\varphi & \cos\Delta\varphi \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{R}(\Delta\varphi)\ \boldsymbol{x}$

It can be seen, that the operations shown above, can handle with points as well as with lines because the line is unambiguously defined as dual primitive to the given point. Then the general transformation matrix in E^2 case is defined as

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix} = \begin{bmatrix} 1 & r & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\Delta\varphi & -\sin\Delta\varphi \\ 0 & 0 & \sin\Delta\varphi & \cos\Delta\varphi \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{Q}(r\ ,\ \Delta\varphi)\ \boldsymbol{x}$

There is a straightforward extension to E^3 space if the cylindrical co-ordinate system is to be used. We can describe a point by a vector:

$$[\ \rho\ ,\ 1\ ,\ cos\varphi\ ,\ sin\ \varphi\ ,\ cos\vartheta\ ,\ sin\vartheta\ ]^T$$

and general rotation transformation by a matrix $\boldsymbol{Q}$ as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos\Delta\varphi & -\sin\Delta\varphi & 0 & 0 \\ 0 & 0 & \sin\Delta\varphi & \cos\Delta\varphi & 0 & 0 \\ 0 & 0 & 0 & 0 & \cos\Delta\vartheta & -\sin\Delta\vartheta \\ 0 & 0 & 0 & 0 & \sin\Delta\vartheta & \cos\Delta\vartheta \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{Q}(r\ ,\ \Delta\varphi\ ,\ \Delta\vartheta)\ \boldsymbol{x}$

4 TRANFORMATION CONCATENATION

In general, very complex transformation might be needed and it is desirable to be handling geometric transformations preferably by the corresponding matrix multiplication.

Let's consider a non-trivial case, when the point A should be moved with the given offset $\boldsymbol{\Delta x}$, see fig.3.

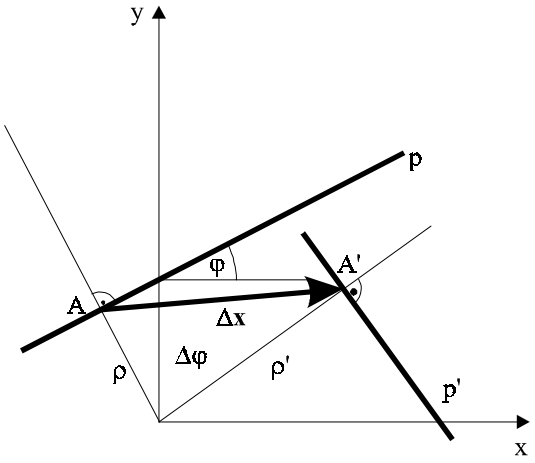

Figure 3

Of course, non-trivial computation is needed if this operation should be performed by using the cartesian co-ordinate system or by deriving the transformation formula directly. Nevertheless this operation can be generally split into the following elementary transformations:

1. Move the point A to the origin distance r_1.
2. Make a rotation by the angle $\Delta\varphi$.
3. Move the point A from the origin to the distance r_2.

So we get the following simple steps:

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi' \\ \sin\varphi' \end{bmatrix} = \begin{bmatrix} 1 & -r_1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\Delta\varphi & -\sin\Delta\varphi \\ 0 & 0 & \sin\Delta\varphi & \cos\Delta\varphi \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi' \\ \sin\varphi' \end{bmatrix} = \begin{bmatrix} 1 & r_2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

where: r is the distance of the point from the origin and $\Delta\varphi$ is the angle that defines the rotation. So we get

$$\boldsymbol{x}' = \boldsymbol{T}(-r_1)\ \boldsymbol{x} \qquad \boldsymbol{x}'' = \boldsymbol{R}(\Delta\varphi)\ \boldsymbol{x}'$$

$$\boldsymbol{x}''' = \boldsymbol{T}(r_2)\ \boldsymbol{x}''$$

It means that the whole transformation can be defined as:

$$\boldsymbol{x}''' = \boldsymbol{Q}(r)\ \boldsymbol{x}$$

where: $\boldsymbol{Q} = \boldsymbol{T}(r)\ \boldsymbol{R}(\Delta\varphi)\ \boldsymbol{T}(-r)$

We have dealt with geometric transformations for points till now. Everybody can see that the representation of the point *A* is unambiguous. We can now define this point as a reference point, which lies on the oriented line *p* and which is perpendicular to the line that passes the line on which the *O* and *A* points lie, see fig.4.

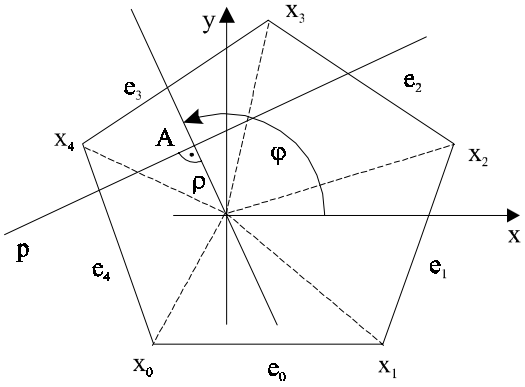

Figure 4

It means that we are able to handle lines in a similar way.

5 APPLICATIONS OF THE PROPOSED REPRESENTATION

There are many possible applications that could use an advantage of the polar or cylindrical representations in some extent. Nevertheless this model is useful especially for non-traditional approach to the algorithm design.

Let us imagine two simple problems like point-in-convex polygon, usually solved in *O(N)* [Ska93a], resp. *O(lg N)* steps, or line clipping by convex polygon often solved by the Cyrus-Beck's algorithm that has *O(N)* complexity), see [Ska94b], [Bui99a], where *N* is a number of edges of the given polygon.

It can be seen that the both problems are dual in some sense. If the point-in-convex polygon problem is considered than it is naturally of *O(lg N)* complexity as we can seek in the array with φ values and because of the known order we can make it in *O(lg N)* steps, see fig.5. We only have to check on which side of the corresponding edges the point A lies.

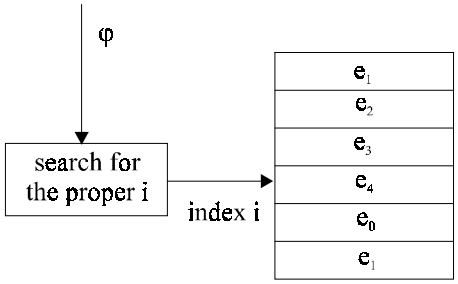

Figure 5

This algorithm can be even speed up in order to avoid the search with the *O(lg N)* complexity. This leads to the algorithm with the run-time complexity bellow *O(lg N)*.

Let us imagine, that we split the whole interval for the φ values to M very small subintervals equidistantly, see fig.6.

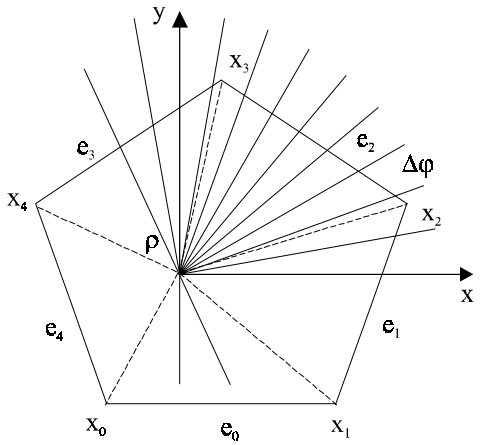

Figure 6

Let us suppose that each element of the array contains an information about the actual edge of the given polygon that is intersected by the i-th wedge that is $\Delta\varphi$ wide and has a vertex in the origin of the co-ordinate system. The value of the index i can be determined as

$$i := \frac{\varphi}{2\pi} M$$

where: M is number of intervals.

It can be seen, that if we have a point with (ρ , φ) co-ordinates, we can determine directly the index of the wedge in which the point lies, see fig.7.

Now it is necessary to determine only on which side of the polygon edge the point lies. Therefore the above mentioned algorithm is of *O(1)* run-time complexity while the pre-processing complexity is of *O(N M)*.

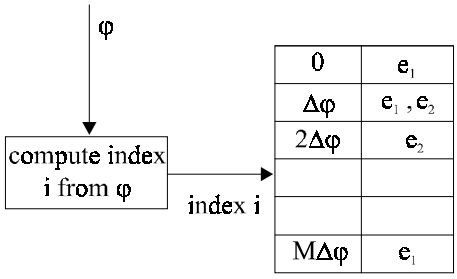

Figure 7

This algorithm cannot be implemented, as it would require an infinite memory, due to the fact that we presumed that $\Delta\varphi \to 0$, i.e. $M \to \infty$. Nevertheless it is possible to determine $max.\Delta\varphi$, i.e. the maximal angle of the wedge, for the given polygon from **its geometrical properties**. In every case the algorithm must expect that two edges will be tested as some wedge can contain two vertices, see fig.7.

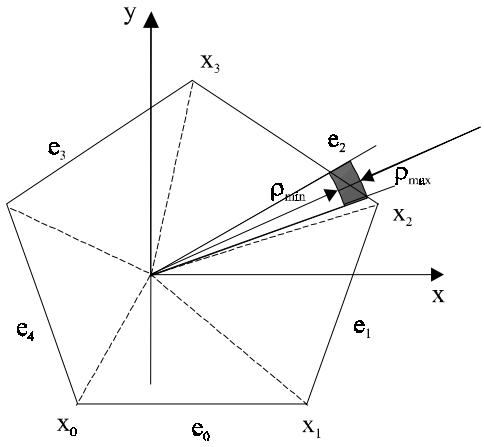

Figure 7

A similar algorithm to this was recently developed for the cartesian co-ordinate system, verified and tested. It proved the *O(1)* expected run-time complexity and detailed description can be found in [Ska96b].

Nevertheless the polar co-ordinate system gives us even faster solution for point-in-polygon test as we can also pre-compute $\rho\, min$ and $\rho\, max$ values and store those values with each wedge and possible situations are as follows:

- $\rho \le \rho_{min}$ point is inside
- $\rho_{max} \le \rho$ point is outside
- in this case the relevant edge must be tested.

So the whole algorithm can be briefly described by following steps:

$$i := \frac{\varphi - \varphi_{\min}}{\varphi_{\max} - \varphi_{\min}} M$$

where: *M* is number of wedges used to split the given polygon. $\varphi max = 2\pi$ and $\varphi min = 0$.

if $\rho min \geq \rho$ **then begin** INSIDE; **EXIT; end**
else
if $\rho \geq \rho max$ **then begin** OUTSIDE; **EXIT; end**
else test the point (ρ,φ) with the i-th edge

It means that we will need a detailed computation only for those cases when the point lies inside of the hatch area, see fig.7.

It can be shown that due to the duality principle that a line is dual to a point and vice versa, see [Kol94a] for details, and the point-in-convex polygon test is dual to the test whether a line intersects the given convex polygon. Therefore a similar approach could be taken for a line clipping problem and develop an algorithm with *O(1)* complexity in polar co-ordinate system. A similar algorithm to this was recently developed for the cartesian co-ordinate system, verified and tested. It proved the *O(1)* expected run-time complexity and detailed description can be found in [Ska96d].

It is necessary to note that if $\Delta\varphi$ is actual angle of a wedge and

$$\Delta\varphi max < \Delta\varphi$$

we get the *O(N)* complexity in the worst case as we have broken the presumption of the algorithm.

6 CONCLUSION AND FUTURE WORK

The presented approach and suggested graphical data representation is not primarily used for direct usage in the fields of computer graphics and data visualization. This representation gives also a very simple way how to handle widgets or cones that might be very useful in some cases.

The most important result is that this approach gives a quite different view on some problems to be solved and there is a hope that this approach can be used for the cylindrical and spherical co-ordinate systems, too. It is expected that non-linear co-ordinate systems with application of dual representation principles can lead to new, more simple, more robust and faster algorithms.There is a believe that the above shown principles can be used especially for problems in E^3 and problems connected to line clipping, intersection computation, ray tracing methods and others.

ACKNOWLEDEMENTS

The author would like to thanks to all who contributed to this work, especially to MSc. and PhD. students at the University of West Bohemia in Plzen that have stimulated this thoughts and development of new algorithms based on it. This paper benefited from several discussions with them a lot.

REFERENCES

[Bui97a] Bui,D.H., Skala,V.: Fast Algorithms for Line Segment and Line Clipping in E^2, The Visual Computer, 1997.

[Bui99a] Bui,D.H., Skala,V.: New Fast Line Clipping Algorithm in E^2 with O(lg N) Complexity, Int.Conf. SCCG'99, Budmerice, Slovak Republic, pp.221-228, 1999.

[Kol94a] Kolingerova, I.: 3D - Line Clipping Algorithms - A Comparative Study, Visual Computer, Vol.11, No.2, pp.96-104, 1994.

[Ska94a] Skala,V.: O(lg N) Line Clipping Algorithm in E^2, WSCG´94 International Conference, pp.174-191, 1994.

[Ska94b] Skala,V.: O(lg N) Line Clipping Algorithm in E^2, Computers & Graphics, Pergamon Press, Vol.18, No.4.

[Ska96a] Skala,V.: An Efficient Algorithm for Line Clipping by Convex and Non-Convex Polyhedra in E^3, Computer Graphics Forum, Vol.15, No.1, pp.61-68, 1996.

[Ska96b] Skala,V.: Line Clipping in E^2 with O(1) Processing Complexity, Computers & Graphics, Vol.20, No.4, pp.523-530, 1996.

[Ska96c] Skala,V., Lederbuch,P. ,Sup,B.: A Compari son of O(1) and Cyrus-Beck Line Clipping Algorithm in E^2 and E^3, SCCG96 Conference proceedings, Comenius Univ. Bratislava, Slovak Republic, pp.27-44, 1996.

[Ska96d] Skala,V.: Line Clipping in E^3 with Expected Complexity O(1), Machine Graphics and Vision, Poland Academy of Sciences, Vol.5, No.4, pp.551-562, 1996.

---

[i] This work was supported by the Ministry of Education of the Czech Republic - projects ME 259, A2030801 and VS 97155

[ii] Affiliated with Multimedia Technology Research Centre, University of Bath, BATH BA2 7AY, U.K.