Non-linear co-ordinates and their application in Computer Graphics

Vaclav Skala[i]

Center of Computer Graphics and Visualization
Department of Computer Science and Engineering
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 those 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. Some properties of those systems are discussed including geometric transformation in the polar, cylindrical and spherical co-ordinate systems. The geometric transformations can be expressed by matrix operations in considered co-ordinate systems. As the memory capacity and hardware graphics accelerator grow fast it can be reasonable to reconsider fundamental functionality of it, as it is limited generally to the vector dot multiplication. Also use of non-linear co-ordinate systems and the influence on design of algorithms with lower algorithm complexity is presented, too.

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

1. INTRODUCTION

There are some applications, where the polar, cylindrical or spherical co-ordinate systems can be used for finding a solution of the given problem. Especially some technical problems [Lio97a], like sonar and radar applications, where the distance from an object is measured under known angles, flow computation, radiation, medical imaging and ultrasound imaging etc. could benefit from their use.

It is a usual practice that all the data from those applications are transformed to the Cartesian 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, cylindrical and spherical co-ordinate systems offer some possibilities how to handle graphical information in an unambiguous way and also make an effective processing. On the other hand, it is necessary to say that in usual practical graphical applications the direct use of non-linear co-ordinates can be quite complicated can hopefully lead to new understanding of some fundamental algorithms and developing of new more effective methods.

2. CARTESIAN CO-ORDINATE SYSTEM

Cartesian co-ordinates and point representations are 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 to represent all geometric transformations by 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 an explicit functions as

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

or

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

or parametrically as

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

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

The problem arises when we need to manipulate with the lines. How can we handle them easily? Let us consider a point **x** and its co-ordinates in the polar co-ordinate system

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

This representation enables us to represent a point unambiguously and define operations like 'move' and 'rotate'. There is a problem with concatenation of several geometric transformations, indeed. Also if we want to display geometric primitives, e.g. points, in the Cartesian co-ordinate space, it will lead to expensive *cos* and *sin* function computations.

It can be seen that every point *A* 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.

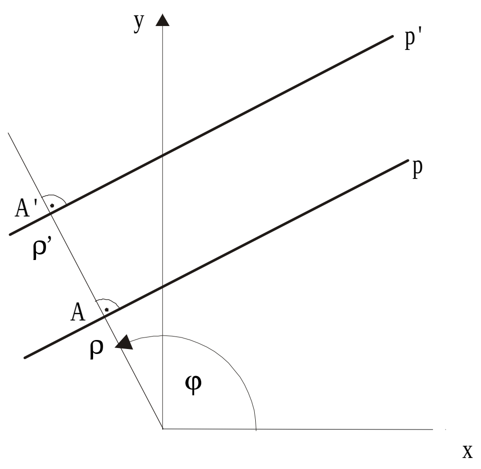

Figure 1

It is well known, that the geometric transformations in the homogeneous co-ordinates 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?

Of course there are unambiguous transformations to the polar and from the polar to the Cartesian co-ordinate systems. Therefore the question whether similar operations for geometric transformations can be defined for the polar co-ordinate system, what would be properties and possible use should be answered.

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

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

where: $\varphi \in \langle 0, 2\pi)$, $\rho \in \langle 0, \infty)$

Inverse transformation is a little bit more complicated and it is defined as:

$$\rho = \sqrt{x^2 + y^2}$$

$$\cos\varphi = \frac{x}{\rho} \qquad\qquad \sin\varphi = \frac{y}{\rho}$$

These transformations are 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 polar co-ordinates as

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

and we will avoid divisions needed in the Cartesian co-ordinates (X, Y) computation from the Cartesian homogeneous co-ordinates, where:

$X = x / w$ and $Y = y / w$

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 polar co-ordinates representation enables us to represent not only the given point but also the given line in the polar co-ordinate system unambiguously without the need of *sin* and *cos* function computations.

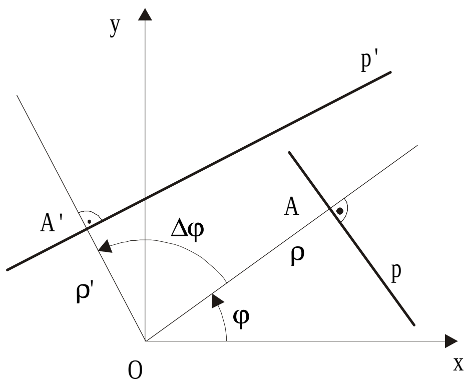

Figure 2

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. To be able to represent geometric transformations by matrix operations we introduce "polar homogeneous co-ordinates" of the point as

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

As the "1" stays for homogeneous co-ordinate.

Note that it is possible to introduce an alternative representation for the polar system

$$[\ \rho^2\ ,\ \rho\ ,\ \rho \cos\ \varphi\ ,\ \rho \sin\ \varphi\ ]^T$$

and therefore we can write

$$[\rho^2\ ,\ \rho\ ,\ \ x\ ,\ y\ ]^T$$

We will use slightly different notation for homogeneous representation of this representation and will see that some additional properties that can be expected from this alternative representation latter on.

In the polar co-ordinate system the radial displacement, analogue to the translation, the 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. $\mathbf{x}' = \mathbf{D}\ (\ r\ )\ \mathbf{x}$

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. It is actually the scaling transformation and it can be described as:

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

i.e. $\mathbf{x}' = \mathbf{S}\ (\ \lambda\ )\ \mathbf{x}$

It should be noted, that the translation operation in polar co-ordinate system is something "strange" as if we move points, let us say the end-points of the line segment $A_1\ A_2$, we will get a line segment $A'_1A'_2$, see fig.3.

We can imagine the radial displacement as a translation operation in some sense because it is the operation when objects move forward to or backward from an observer in the origin of the polar system.

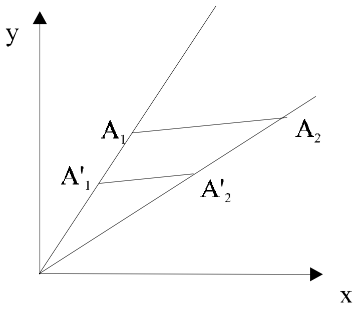

Figure 3

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 [Rek96a]

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

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

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 a matrix for a rotation in the 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. $\quad x' = \mathbf{R}\,(\Delta\varphi)\;x$

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 case is defined as:

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

i.e. $\quad x' = \mathbf{Q}\,(\,r\,,\,\lambda\;,\,\Delta\varphi\,)\,x$

If we compare the structure of the matrix ***Q*** with the structure of a matrix for a transformation in homogeneous co-ordinate system we can see that the matrix ***Q*** has a simple structure. A user can easily recognize single transformation parameters, that is not so easy in the homogeneous co-ordinate system representation. We can see that each transformation parameter has a 2 x 2 block of values in the matrix ***Q***.

In case that we need the translation operation known from the Cartesian homogeneous co-ordinate representation in E^2 we can use a different representation for a point ***x*** as

$$[\,\rho\cos\varphi\,,\,\rho\sin\varphi\,,\,1\,,\;\rho^2\,,\,\rho\,]^T$$

and translation vector $\boldsymbol{x_1}$ is given as

$$[\,\rho_1\cos\varphi_1\,,\,\rho_1\sin\varphi_1\,,\,1\,,\;\rho_1^{\,2}\,,\,\rho_1\,]^T$$

then the final vector can be determined as

$$\rho'^{\,2} = \rho_1^{\,2} + \rho^2 - 2\rho\,\rho_1\cos(\,\varphi - \varphi_1\,)$$

whrere:

$$\cos(\,\varphi - \varphi_1\,) = \cos\varphi\,\cos\varphi_1 + \sin\varphi\,\sin\varphi_1$$

This operation can be expressed by the matrix operation that must be completed by the computation of the ρ' as the $\rho' = \sqrt{\rho'^2}$ *cannot be expressed by matrix multiplication.*

Nevertheless all previous transformations like rotation, mirror, scaling can be expressed in a similar manner as

$$\mathbf{x}' = \mathbf{Q} * \mathbf{x}$$

where: $\quad \mathbf{x} = [\,\rho\cos\varphi\,,\,\rho\sin\varphi\,,\,1\,,\;\rho^2\,,\,\rho\,]^T$

$$\mathbf{x}' = [\,\rho'\cos\varphi'\,,\,\rho'\sin\varphi'\,,\,1\,,\;\rho'^2\,,\,\rho'\,]^T$$

$$\mathbf{Q} = \begin{bmatrix} 0 & 2\rho_1\cos\varphi_1 & 0 & 0 & 0 \\ 0 & 2\rho_1\sin\varphi_1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 2\rho_1\cos\varphi_1 & 2\rho_1\sin\varphi_1 & \rho_1^2 & 1 & 0 \\ 0 & 0 & 0 & \sqrt{*} & 0 \end{bmatrix}$$

***** means generalized matrix multiplication in the sense that the value of $\rho' = \sqrt{\rho}$ for all transformed points.

4. CYLINDRICAL CO-ORDINATE SYSTEM

Let us consider the cylindrical co-ordinate system, that can be described, see fig.4, as:

$$x = \rho\;\cos\varphi \qquad y = \rho\;\sin\varphi \qquad z = \xi$$

where: $\varphi \in <0\,,\,2\pi),\ \rho \in <0\,,\,\infty),\ \ \xi \in (\,-\infty\,,\,\infty\,)$

It can be seen the point unambiguously represents a plane in E^3 space, too. We can describe a point ***x*** in the cylindrical co-ordinate system by a vector:

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

and the rotation transformation by a matrix $\boldsymbol{R_z}$ as

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

i.e. $\quad x' = \mathbf{R}_z\,(\,\Delta\varphi\;)\,x$

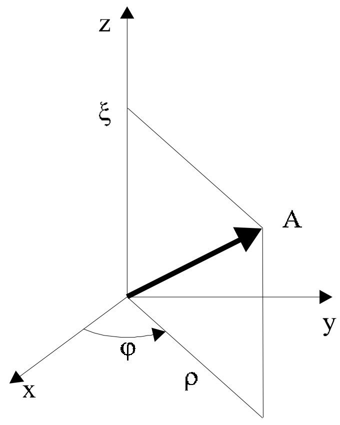

Figure 4

Radial displacement (translation) *r* in the *x-y* plane as

$$\begin{bmatrix} 1 & r & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Translation ξ in the z-axis direction as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \xi \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{T}_z(\xi)\,\boldsymbol{x}$

Scaling by a factor λ in the *x-y* plane as:

$$\begin{bmatrix} \lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{S}_{xy}(\lambda)\,\boldsymbol{x}$

Scaling by a factor η in the direction of z-axis as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \eta & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

i.e. $\boldsymbol{x}' = \boldsymbol{S}_z(\eta)\,\boldsymbol{x}$

All transformations can be easily described by

$$\begin{bmatrix} \lambda & r & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos\Delta\varphi & -\sin\Delta\varphi & 0 & 0 \\ 0 & 0 & \sin\Delta\varphi & \cos\Delta\varphi & 0 & 0 \\ 0 & 0 & 0 & 0 & \eta & \xi \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

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

where: *r* is the translation coefficient in x-y plane, λ is the scaling factor, $\Delta\varphi$ is rotation angle around z axis, ξ is the translation and η is the scaling factor in the direction of z axis. The matrix ***Q*** consists of 2 x 2 blocks of transformation parameters again. It can be seen that others geometric transformations can be defined as follows.

5. SPHERICAL CO-ORDINATE SYSTEM

The second system very often used in E^3 is the spherical co-ordinate system, where

$$x = \rho \cos\varphi \cos\vartheta \qquad y = \rho \sin\varphi \cos\vartheta$$
$$z = \rho \sin\vartheta$$

where: $\varphi \in \langle 0, 2\pi)$, $\vartheta \in \langle -\pi/2, \pi/2 \rangle$, see fig.5.

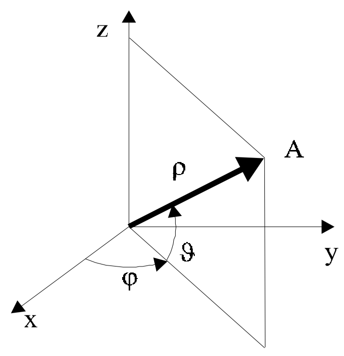

Figure 5

It can be seen the point unambiguously represents a plane in E^3 space, too.

We can describe a point by a vector:

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

and rotation transformation by a matrix ***R*** 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. $\mathbf{x}' = \mathbf{R}\,(\,\Delta\varphi\,,\Delta\vartheta\,)\,\mathbf{x}$

It can be seen that the result of the transformation does not depend on the order of rotation that is not true in the Cartesian homogeneous co-ordinate system representation.

Now we can also define general transformation matrix with others transformations like displacement, scaling etc.

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

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

where: r is the displacement coefficient in direction of OA, λ is the scaling factor, $\Delta\varphi$ is the rotation angle around z axis, $\Delta\vartheta$ is the rotation angle as defined in the spherical co-ordinate system.

6. TRANFORMATION CONCATENATION

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

Let's consider a non-trivial case, when the point A should be moved with the given offset $\boldsymbol{\Delta x}$, see fig.6.

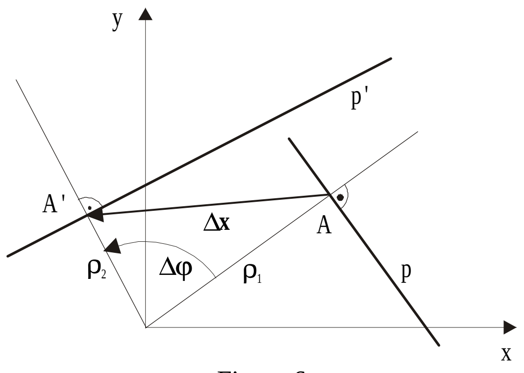

Figure 6

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 ρ_1.
2. Make a rotation by the angle $\Delta\varphi$.
3. Move the point A from the origin to the distance ρ_2.

So we get the following simple steps:

$$\begin{bmatrix} \rho' \\ 1 \\ \cos\varphi' \\ \sin\varphi' \end{bmatrix} = \begin{bmatrix} 1 & -\rho_1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ 1 \\ \cos\varphi \\ \sin\varphi \end{bmatrix}$$

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

$$\begin{bmatrix} \rho''' \\ 1 \\ \cos\varphi''' \\ \sin\varphi''' \end{bmatrix} = \begin{bmatrix} 1 & \rho_2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \rho'' \\ 1 \\ \cos\varphi'' \\ \sin\varphi'' \end{bmatrix}$$

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

$$\mathbf{x}' = \mathbf{T}\,(\,-r_1\,)\ \mathbf{x} \qquad \mathbf{x}'' = \mathbf{R}\,(\Delta\varphi)\ \mathbf{x}'$$
$$\mathbf{x}''' = \mathbf{T}\,(\,r_2\,)\ \mathbf{x}''$$

It means that the whole transformation can be defined as:

$$\mathbf{x}''' = \mathbf{Q}\,(\,r_1, \Delta\varphi, r_2\,)\ \mathbf{x}$$

where:

$$\mathbf{Q}(r_1, \Delta\varphi, r_2) = \mathbf{T}(r_2)\ \mathbf{R}(\Delta\varphi)\ \mathbf{T}(-r_1) = \mathbf{T}(r_2 - r_1)\ \mathbf{R}(\Delta\varphi)$$

Nevertheless this transformation can be realized also as a rotation by $\Delta\varphi$ angle and by translation with $\rho_2 - \rho_1$.

We have dealt with geometric transformations for points till now. It can be seen that the representation of the point *A* is unambiguous. We can now define the point *A* as a reference point, which lies on the oriented line perpendicular to the line *p* and that passes the origin the *O*, see fig.6. It means that we are able to handle lines in a similar manner as points.

7. OTHER USEFUL TRANSFORMATIONS

So far, we have dealt with simple geometric transformations but there are other very useful transformations like scale, mirror, sheering etc. The scaling transformation is a simple one as it was actually defined above as relative translation, i.e. scaling by a parameter λ.

Let us consider others transformations, now. It can be proved that

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

represents mirroring according to y-axis, while

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

represents mirroring according to $x = y$ axis. Others transformations can be derived easily, too.

8. PROJECTIONS

All graphics packages do need an operation for the projection from E^3 to E^2 space. It is well known that only a planar projection is used within the standard graphics packages. Nevertheless there are applications that could benefit from non-planar projections like cylindrical and spherical.

The software that enables that use standard homogeneous co-ordinate system and compute appropriate projection transformations, mostly computationally expensive. The polar representation can be considered actually as a special kind of E^2 to E^1 projection when the observer is in the origin of the polar co-ordinate system with some additional information how far every object is from the observer. So we can represent a wedge $\Delta\varphi$ wide easily. This idea can be directly extended to E^3 case.

So far we have dealt with the representation of geometric transformations using non-linear co-ordinate systems. From the equations it can be seen that if the co-ordinate ρ is not considered, we get the projection of geometric primitives directly, actually without computation. There is a condition that we will use the spherical projection, see fig.7, or cylindrical projection, see fig.8, for geometric primitives represented in spherical or cylindrical co-ordinate systems.

The spherical co-ordinate system enable us to define actually the perspective viewing pyramid in a very simple way as we have to only define $\Delta\varphi$ and $\Delta\vartheta$ appropriately if an observer is in the origin.

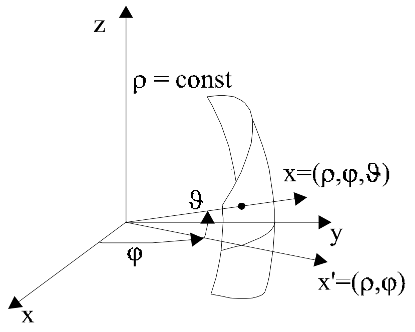

Figure 7

The similar properties are valid for cylindrical co-ordinate system, see fig. 8.

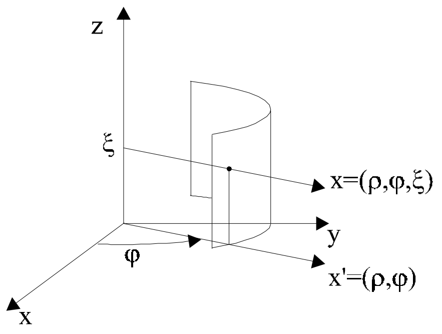

Figure 8

9. SPACE SUBDIVISION

The space subdivision can speed-up the processing. This approach is well known from the ray-tracing applications where the space is split to orthogonal subspaces and all objects interfering with this sub-space are registered. When a ray is shot, all objects in the sub-spaces intersected by this ray are tested for the intersection. Usually the 3DDA algorithm is used to find those sub-spaces in the order of growing distance from the point the ray was shot from. It can be seen that the considered non-linear co-ordinate system have naturally this property as if we subdivide the space by $\Delta\varphi$ and $\Delta\vartheta$ we get rectangular mesh in the φ, ϑ space, that is actually a **viewing pyramid** if the observer is the origin of the spherical co-ordinate system. Naturally the value of ρ gives us the distance of the object from the origin.

10. APPLICATIONS OF THE NON-LINEAR CO-ORDINATE SYSTEMS

There are some possible applications that could use an advantage of the polar, cylindrical or spherical representations in some extent. Nevertheless these models seem to be useful especially for non-traditional approach to the algorithm design.

Let us imagine two simple problems like:

- the point-in-convex polygon test, usually solved in *O(N*) [Ska93a], resp. *O(lg N)* steps,
- the line clipping by convex polygon often solved by the Cyrus-Beck's algorithm that has *O(N)* complexity, see [Ska94b], [Bui97a], or *O(lg N)* algorithm can be considered [Ska94a], [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, see [Kol94a]. The principle of duality is well known and used often used [Joh96a].

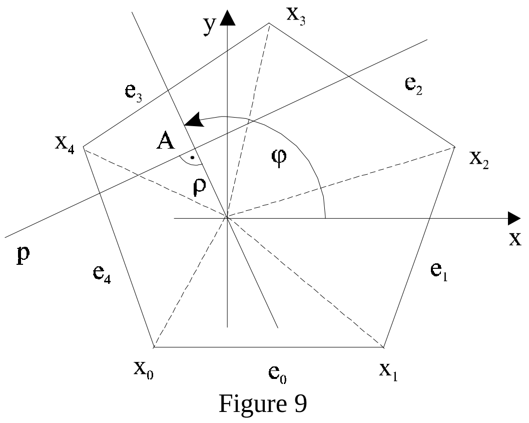

Figure 9

The test point-in-convex polygon is naturally of *O(lg N)* complexity. 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.10. We have to only check on which side of the corresponding edges the point *A* lies.

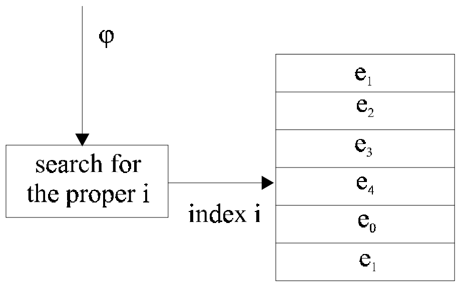

Figure 10

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).*

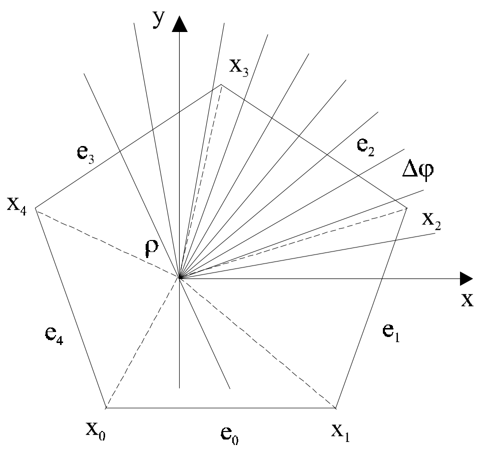

Figure 11

Let us imagine that we split the whole interval for the φ values to *M* very small subintervals equidistantly, see fig.11.

Let us suppose that each wedge is represented by an array element. The i-th element contains an information about the actual edge of the given polygon 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.12.

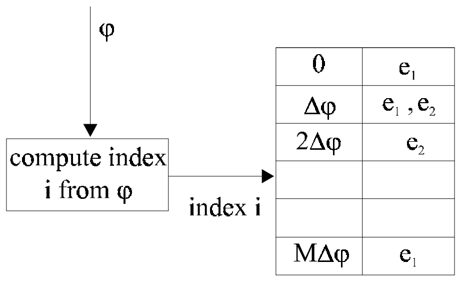

Figure 12

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)*.

Generally, this algorithm cannot be made to run in *O(1)* time, as it would require an infinite memory, due to the fact that we presumed that $\Delta\varphi \to 0$, i.e. $M \to \infty$. It is possible to determine *max.Δφ*, i.e. the maximal angle of the wedge, for the given polygon from **geometrical properties** of the given polygon. In every case the algorithm must expect that two edges will be tested as some wedge can contain two edges, see fig.11.

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 [Ska94d].

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
- otherwise the relevant edge must be tested.

The whole algorithm can be briefly described by following steps:

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

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

```
if $\rho_{min} \ge \rho$ then begin INSIDE; EXIT; end
else if $\rho \ge \rho_{max}$ then begin OUTSIDE; EXIT; end
        else test the point $(\rho,\varphi)$ 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.13.

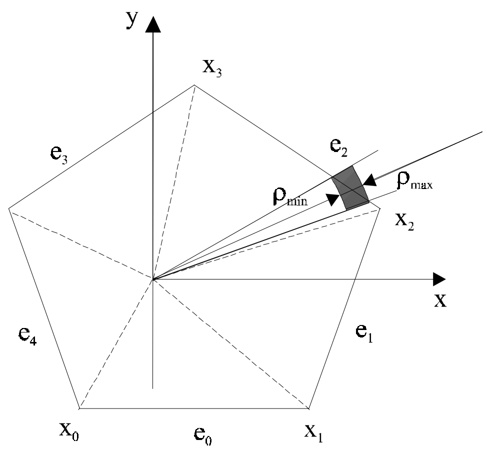

Figure 13

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 [Ska96b]. It is necessary to note that if *Δφ* 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.

The line clipping by convex polygon in E^2, see fig.14, is a problem when we want to find a part of the line *p* that is inside of the given polygon if any. If we transform the convex polygon to the polar co-ordinate system representation we get representation in *(ρ,φ)* co-ordinates, see fig.15, where each region represents the a set of *(ρ,φ)* values of the clipped line that intersect the same edges, in our case edges e_1 and e_4.

It can be seen that if the space subdivision technique is used we can directly determine edges that are intersected by the given line *p* regardless to the number of edges of the given polygon for infinite subdivision in *(ρ,φ)* space. It means that if the subdivision is coarser we will have to test more than two edges, see fig.15. Similar approach has been recently developed in the cartesian co-ordinate space, verified and tested, see [Ska94c], [Ska96b], [Ska96e], and extended to E^3 case [Ska96d], too.

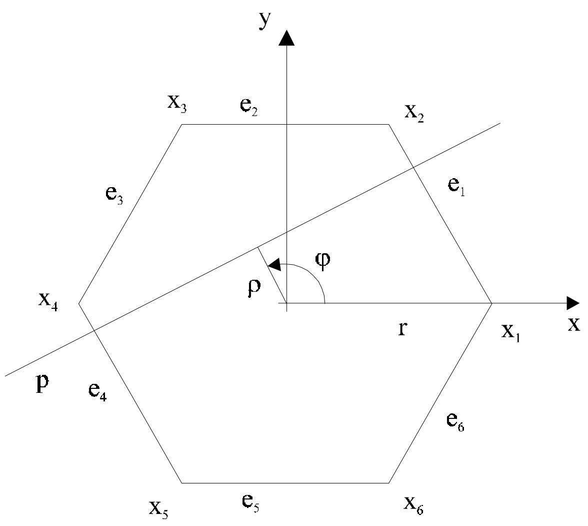

Figure 14

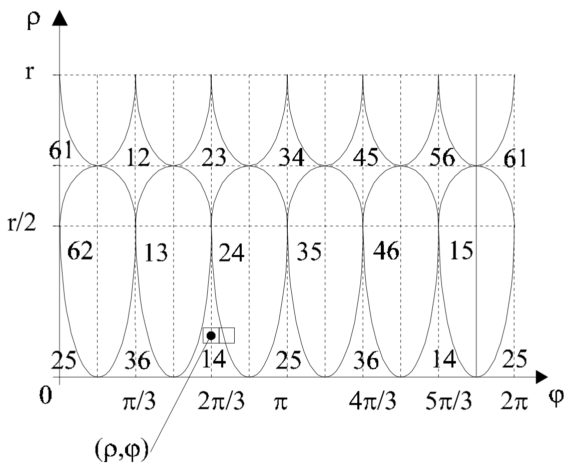

Figure 15

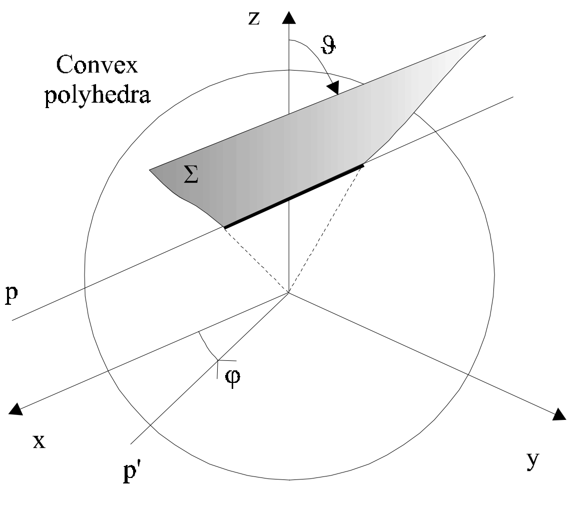

Figure 16

The line clipping by convex polyhedron in E^3, see fig.16, is a problem when we want to find again a part of the given line that is inside of the given convex polyhedron. It is known that this problem can be easily solved by the Cyrus-Beck algorithm or others with *O(N)* complexity, see [Ska97a] or if some additional information is given, like neighbours of each facets, we can get a faster algorithm with *O(√N)* expected complexity, see [Ska97a]. There have been several attempts to find an algorithm similar to the *O(1)* algorithm for E^3 case [Ska96c], but only expected *O(1)* complex have been reached.

It can be seen that any line *p* unambiguously defines a plane Σ that passes the origin and on which the line *p* lies. So it is possible to pre-compute sets of all triangles of the given convex polyhedron intersected by the plane Σ. They are unambiguously defined by parameters (φ, ϑ). The parameter φ is defined as an angle between the axis x and a line *p*' that is the intersection of the plane Σ and x-y plane. The ϑ parameter defines the angle between the plane Σ and the axis *z*. Let us assume that we will split the space of (φ, ϑ) to $(\Delta\varphi,\Delta\vartheta)$ intervals, i.e. rectangle in the parameter space, assuming that $\Delta\varphi \to 0$ and $\Delta\vartheta \to 0$. Now, we have received lists of triangles for each $(\Delta\varphi,\Delta\vartheta)$ that are intersected by the set of related planes Σ. Nevertheless there are many lines *p* that lie on the plane Σ, but each such a line is unambiguously defined by the polar co-ordinates (ρ,φ') on the plane Σ. We can use a similar approach the *O(1)* line clipping algorithm in E^2 and again pre-compute a list of triangles intersected by the line *p* with co-ordinates (ρ,φ') . It can be seen that if $\Delta\rho \to 0$ and $\Delta\varphi' \to 0$ the list will contain only two triangles that are intersected by the line *p*.

It can be seen that this algorithm is of *O(1)* run-time complexity, but the pre-processing is probably of $O(N^2M^4)$ complexity. Nevertheless this is a limit case as this complexity requires an infinite memory because $\Delta\varphi \to 0$, $\Delta\vartheta \to 0$, $\Delta\rho \to 0$ and $\Delta\varphi' \to 0$.

11. CONCLUSION AND FUTURE WORK

In the previous sections we discussed properties of the polar co-ordinate system in E^2, of the cylindrical and spherical co-ordinate systems in E^3, representation of points, geometric transformations, representation of concatenated geometric transformations and some possible applications of the proposed representations.

The advantage of the proposed approach is that geometric transformations are handled in a unique way using matrix multiplication. 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. 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 are two direct results for *N*-sided convex polygon:

- The test point-in-convex polygon is of *O(lg N)* complexity. It was shown that we can reach ***O(1)* run-time complexity**, if we use a pre-processing of *O(N M)* complexity, where: *M* defines number of wedges to be pre-computed and **depends on geometric properties of the given polygon** $M \to \infty$. This is in correlation with previously obtained results.

- The line clipping against convex polygon is of *O(lg N)* complexity, see [Ska94a], [Ska94b], [Bui99a]. It was shown that it is possible to reach ***O(1)* run-time complexity**, if we use a pre-processing of $O(N^2 M R)$ complexity where: *M*, resp. *R* define number of subdivision for $\Delta\varphi$, resp. $\Delta\rho$ to be pre-computed and **depend on geometric properties of the given polygon** $M \to \infty$, $R \to \infty$. This is in correlation with previously obtained results, see [Ska96b], [Ska96c], [Ska96e] and with extensions to E^3 case, too.

There are some very important questions from the algorithm complexity field.

- What is the relation between the complexity of optimal algorithm and pre-processing and run-time complexities?
- What is the lowest bound for the run-time complexity for a given problem and what will be the lowest pre-processing complexity for this?
- Can be the above-mentioned approach extended to E^3 case?

There is a hope that the above shown principles can be used especially for problems in E^3 [Ska96a], [Ska97a] and problems connected to line clipping, intersection computation, ray tracing methods and others.

As the memory capacity and speed of hardware graphics accelerator grow fast it can be reasonable to reconsider fundamental functionality of geometric engines, too. Formulation of new functionality for the graphics and visualization pipeline might be useful, as it is limited generally to the dot vector multiplication nowadays.

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. Special thanks belong to Dr.A.Kolcun, to anonymous reviewers of this paper and recent reports as they shared some valuable insights on this problem solution. Theirs invaluable critical comments and suggestions improved the manuscript significantly.

REFERENCES

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

[Joh96a] Johnson,M.: Proof by Duality: or the Discovery of "New" Theorems, Mathematics Today, November/December 1996.

[Kol94a] Kolingerova, I.: Dual Representation and its Use in Computer Graphics, PhD Thesis, University of West Bohemia, Plzen, 1994.

[Lio97a] Lionello,R.: Three Dimensional Imaging for Magnetohydrodynamic Computations, WSCG'97 Int.Conf.Proceedings, Univ.of West Bohemia, Plzen pp.282-289, 1997

[Rek69a] Rektorys,K. & others: Survey of Applicable Mathematics, Illife Books Ltd., Dorset House, Stamford Street, London SE1, ISBN 592 03927 7, U.K.

[Ska93a] Skala,V.: An Efficient Algorithm for Line Clipping by Convex polygon, Computers & Graphics, Vol.17, No.4, pp.417-421, 1993.

[Ska94a] Skala,V.: O(lg N) Line Clipping Algorithm in E^2, WSCG´94 Int.Conf. Proceedings, Univ.of West Bohemia, Plzen, pp.174-191, 1994.

[Ska94b] Skala,V.: O(lg N) Line Clipping Algorithm in E^2, Computers & Graphics, Pergamon Press, Vol.18, No.4., 1994.

[Ska94c] Skala,V., Kolingerova,I., Blaha,P.: A Comparison of 2D Line Clipping Algorithms, Machine Graphics & Vision, Vol.3, No.4, pp.625-633, 1994.

[Ska94d] Skala,V.: Point-in-Polygon with O(1) Complexity, TR 68/94, University of West Bohemia, Plzen, 1996.

[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 Compar ison 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.

[Ska96e] Skala,V., Lederbuch,P.: A Comparison of a New O(1) and the Cyrus-Beck Line Clipping Algorithms in E^2, in COMPUGRAPHICS'96 Int.Conf., Paris, 1996.

[Ska97a] Skala,V.: A Fast Algorithm for Line Clipping by Convex polyhedron in E^3, Computers & Graphics, No.2, Vol.21, pp.209-214, 1997.

[i] This work was supported by the Ministry of Education of the Czech Republic - projects A2030801and VS 97155