The basic principles and concepts of a software package for visualisation of Mathematics

Vesna Veličković

Abstract

We give a short survey of the basic ideas and concepts of our software. Our software and its extensions run on any PC under Windows. It is the purpose of our software to provide the geometric setup and the techniques to represent geometric configurations on PC screens, plotters and postscript devices such as printers, and to create visualisations and animations. It is available in two variants. The first one is for the users familiar with programming and provides the freedom of extensions. It requires a TURBO-PASCAL compiler 5.5 (or any more advanced version), or Borland DELPHI. No other graphics package is needed. The second one is for the users who want to obtain fast results without programming. For them our Windows applications are available which work independently and allow the user to change number of parameters.

Mathematics Subject Classification: 53A05,68N05; 40H05,46A05.
Key words: computer graphics, visualisation, animation, software development.

1 Introduction

Visualisation and animations are of vital importance in modern mathematical education. They strongly support the understanding of mathematical concepts. It also has various applications in research.

We developed our own software package ([8, 6, 7, 9]) in Borland PASCAL and DELPHI to create our graphics for visualisation and animations of the results from classical geometry, analytic geometry or differential geometry. It has applications to physics, chemistry, cartography, crystallography ([2, 3]), and the engineering sciences.

It is available in two variants, a free software in Pascal and the Windows applications.

The first variant is for users familiar with programming. The source code of the programmes and units are available. Thus offers the freedom for extensions. The users can use an existing programme or write new ones according to their needs. Adding

Thě Fifth Conference of Balkan Society of Geometers, Aug. 29 - Sept. 2, 2005, Mangalia, Romania; BSG Proceedings 13, Geometry Balkan Press pp. 192-203.

new objects and units, they can visualise results in new fields. It requires a TURBO-PASCAL compiler 5.5 (or any more advanced version), or Borland DELPHI. No other graphics package is needed.

The second variant is for users who do not want to programme, but to otain the available graphics fast. For them we provide our Windows applications which work independently of any other software. The users can change number of parameters of the programmes which have the graphics interface. They can work efficiently and comfortably, but can neither add new programmes nor objects.

Our graphics can be exported to several formats such as BMP, PS, PLT, SCR (screen files under DOS), or GCLC, the Geometry Constructions Language Converter developed at Belgrade University ([1, 4], or for further information, `http://www.matfbg.ac.yu/~janicic/gclc`). By means of any graphics converter software, these formats can be converted to a number of other graphical formats such that GIF, JPG, PCX, TGA, TIF, PSD, EPS, PNF or PDF, and can be included in a TEX or LATEX file.

We create animations in animated GIF format by means of the software package Animagic GIF 32. For this we produce a number of GIF files of our graphics. They can be included in an HTML file to obtain our animations.

We emphasize that all the graphics in this paper were created with our software package, and then processed in the way described above; we did not use any other software package.

2 The features of our software

The most important features of our software are

- *openness*, the users have access to its source files, and therefore may change or add their own programmes
- *extendability to other areas*, thus it is applicable to both teaching and research
- *line graphics*, the ability to draw arbitrary curves on surfaces and more surfaces than one at a time and their lines of intersection
- *visibility checks*, the concept for the solution of visibility problems strongly supports the visualization of geometric principles
- *independence*, the concepts of our software are independent of a particular programming language; they can easily be transferred to other OOP languages

No other comparable, comprehensive software is available in this field.

The main principles of our software are

- *strict separation of geometry and technique of drawing*
- *line graphics*
- *central projection*
- *independent visibility checks*

We strictly separate geometry from the technique of drawing. This immensely adds to the overall transparency of the structure of our software. We may concentrate on geometry, once all the tools needed in the actual drawing process have been developed.

The use of line graphics means in particular that we represent surfaces by families of curves on them, normally by their parameter lines. This also helps to solve the problems of drawing arbitrary curves on surfaces and the lines of intersection of different surfaces which constitute major problems in many conventional graphics packages (Figure 1). Furthermore, line graphics are most suitable for many problems in differential geometry (Figure 2). Curves may be given by parametric representations or equations. They are approximated by polygons.

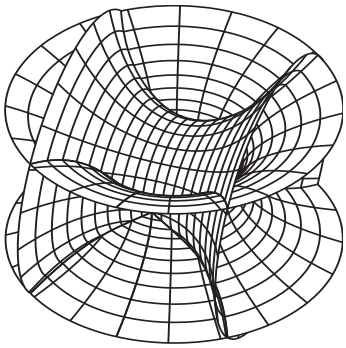

Fig. 1. Catenoids and their lines of interection

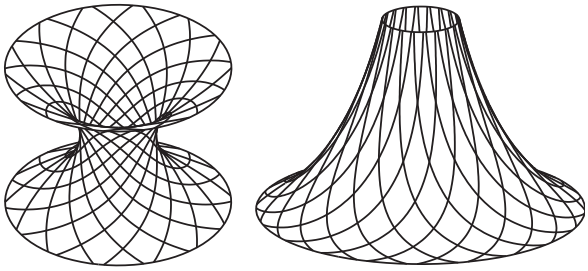

Fig. 2. Asymptotic lines on a catenoid and a pseudo–sphere

We developed an independent visibility check to analytically test the visibility of any point, immediately after the computation of its coordinates. Thus our graphics are generated in a geometrically natural way. The independence of the check procedure enables us to manipulate visibility to be able to show, if necessary, desirable but geometrically unrealistic effects, or not to use any test at all for a fast first sketch (Example 2.1 and Figure 3).

Example 2.1. Dandelin's spheres
The uniquely defined spheres that are both tangent to a cone along a circle line and an intersecting plane are called Dandelin's spheres*. They are tangent to the plane of intersection at the foci of the resulting conic section (Figure 3).*

Two consecutive points of a curve are joined by a straight line segment if and only if both of them are visible. Invisible parts of curves may either be dotted or not drawn at all. To speed up the process we use interpolation to close occasional gaps.

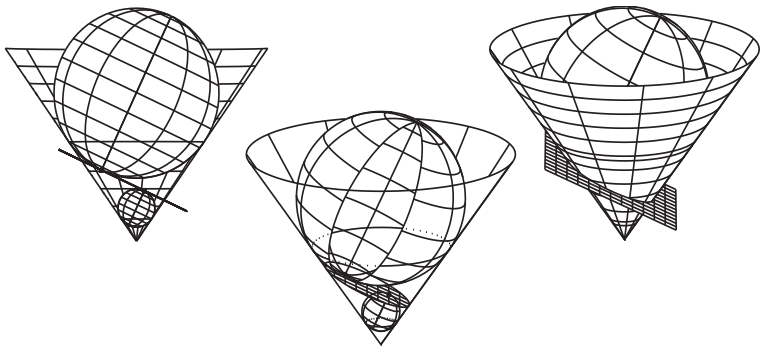

Fig. 3. Dandelin's spheres, concept, reality and desirable representation

We use central projection with a free choice of parameters to create a two–dimensional image of any three–dimensional geometric configuration. This is the most general case. The central projection is uniquely defined by the centre of projection $C = COP$ and the projection plane $Pl = PrPl$. The choices of C and Pl are free with the pathological exceptions that C must not be in Pl, and the plane parallel to Pl through C must not intersect the world interval.

We decided to give the parameters of the projection in spherical coordinates, since it seems to be easier to have an idea of the position of the centre of projection and the projection plane with respect to spherical coordinates. Normally we choose the projection plane orthogonal to the direction of the centre of projection C, and its origin $PrPl.O$ antipodal to C (Figure 4); this prevents distortions (Figure 5). The projection is illustrated in Figures 6 and 7.

COP	centre of projection in spherical coordinates r, ϕ, Θ
$PrPl$	projection plane (orthogonal to the direction of COP)
$PrPl.O$	origin of the projection plane (antipodal to COP)
S^1	unit sphere (to mark the ϕ and Θ)
P_1	point on S^1 with angles ϕ and Θ

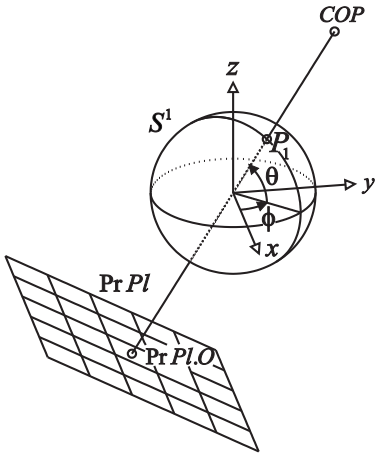

Fig. 4. The definition of the central projection

Distortions may occur, when the direction of the centre of projection is not orthogonal to the projection plane

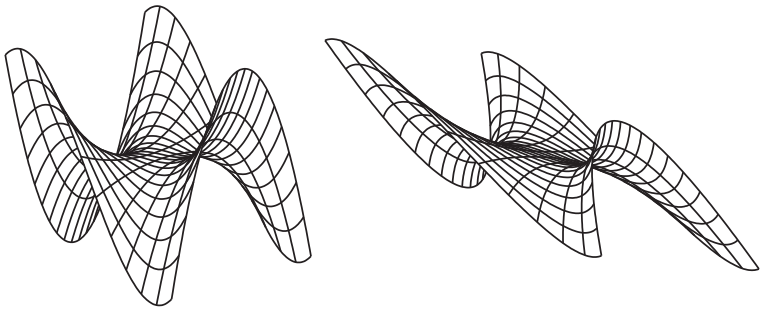

Fig. 5. Distortion by central projection

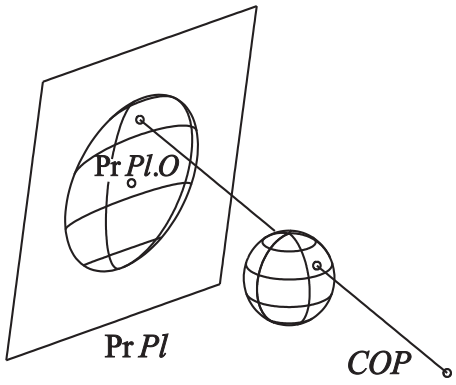

Fig. 6. The principle of the central projection

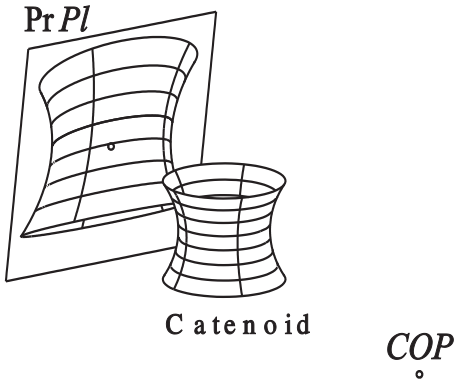

Fig. 7. The central projection of a catenoid

3 The visibility of points

In this part, we deal with the visibility problems that appear in the representation of surfaces. Since these problems depend on the geometry of the objects to be drawn, we implemented the methods for the solution of the visibility problems in the types for the surfaces of various classes.

In general, we may want to draw several surfaces in one configuration. Every point of each surface has to be tested analytically for its visibility. This means it has to be determined

(1) if the point is visible with respect to the surface it is on

(2) if the point is not hidden by any of the other surfaces

Hence there are two procedures for the visibility check of a point, $Visibility$ and $NotHidden$.

Let P be any point in three–dimensional $\mathbb{R}^3$, S be a surface with a parametric representation $\vec{x}(u^i)$ $((u^1, u^2) \in D \subset \mathbb{R}^2)$, where D is a domain, and $C = COP$ be the centre of projection. We denote the position vector of P by $\vec{p}$ and put $\vec{v} = \overrightarrow{PC}$. Then P is hidden by S if and only if there exit a pair $(u^1, u^2) \in D$ and a real $t > 0$ such that the following condition is satisfied

$$\vec{x}(u^1, u^2) = \vec{p} + t\vec{v}. \tag{3.1}$$

Let P be a point on a surface S in a class $\mathcal{S}$ of surfaces. The method

```
PROCEDURE S.Visibility (P:Pt3D; PrRay:Line3D; Dist:EXTENDED;
                        VAR Vis:BOOLEAN);
```

determines if P is hidden by some other point on S, that is if there is a point on the projection ray $PrRay$ between P and the centre of projection COP, in which case P is invisible, hence $Vis = FALSE$, or not, in which case $Vis = TRUE$.

The method

```
PROCEDURE S.NotHidden (P:Pt3D; PrRay:Line3D; Dist:EXTENDED;
                       VAR NotHid:BOOLEAN);
```

determines if an arbitrary point P in space is hidden by some point of the surface S, in which case $NotHidden = FALSE$, or not, in which case $NotHidden = TRUE$.

The procedure

```
PROCEDURE Check (P:Pt3D; PrRay:Line3D; Dist:EXTENDED;
                 VAR Chk:BOOLEAN);
```

uses the methods $Visibility$ and $NotHidden$ to determine whether or not the point P on a surface S in the class $\mathcal{S}$ is visible, in which case $Chk = TRUE$, or not, in which case $Chk = FALSE$. We have

$$Chk = true \iff \begin{cases} Vis = true & \text{in } S.Visibility(P, PrRay, Dist, Vis) \\ & \text{and} \\ NotHid = true & \text{in } S^*.NotHidden(P, PrRay, Dist, \\ & NotHid), \text{for all } S^* \in \mathcal{S} \setminus \{S\}. \end{cases}$$

4 The contour line of a surface

The use of line graphics has the effect that surfaces appear unfinished without their so–called contours (Figure 8).

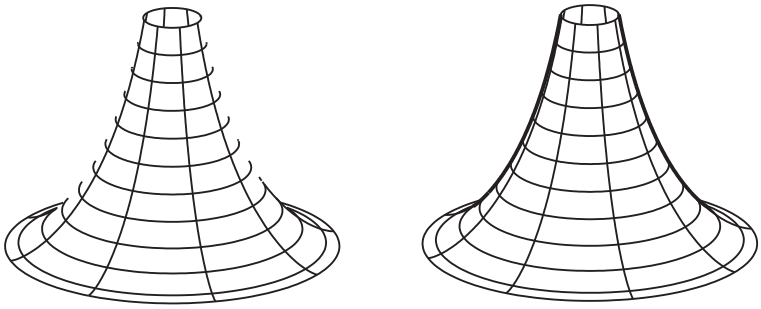

Fig. 8. A pseudo sphere without and with contour

One might have the idea of the contour line of a surface as being a curve that separates the visible points of the surface from invisible ones. But this is not true in every case, as is illustrated in Figure 9.

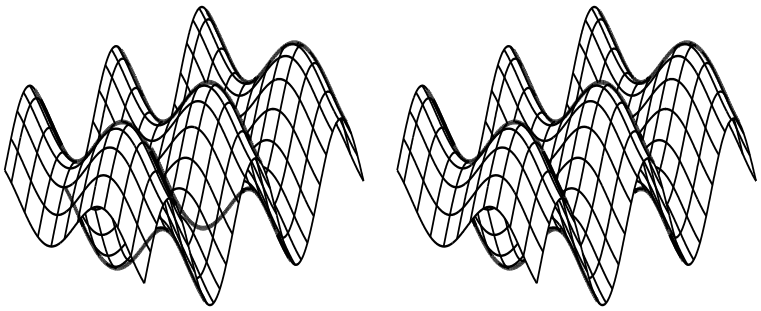

Fig. 9. A contour line without and with visibility check

Let S be a surface with a parametric representation $\vec{x} = \vec{x}(u^i)$ and surface normal vector $\vec{N}$, P be a point on S and C be the centre of projection. Then we say that P is a *contour point* if and only if the following two conditions hold.

(i) The projection ray to the point P is orthogonal to the surface normal vector $\vec{N}$ at P, hence

$$\overrightarrow{CP} \bullet \vec{N} = 0. \tag{4.1}$$

(ii) Let E be the plane through P spanned by the vectors $\overrightarrow{CP}$ and $\vec{N}$, and γ be the curve of intersection of S and E. Then there is a neighbourhood of P in which γ runs on one side of the projection ray and has no points of intersection with it other than P (Figure 10).

Thus, if P^* is a point of γ and $\vec{N}(P^*)$ is the surface normal vector at P^*, then the function f defined by

$$f(P^*) = \overrightarrow{CP^*} \bullet \vec{N}(P^*)$$

has a zero at $P^* = P$ where it changes sign, and there are no other zeros of f in some neighbourhood of P.

COP **centre of projection,** $PrPl$ **projection plane,** S **surface,** P **contour point,** $\vec{N}$ **surface normal vector,** $Plane$ **spanned by** $\overrightarrow{PC}$ **and** $\vec{N}$**,** γ **intersection** $Plane \cap S$**.**

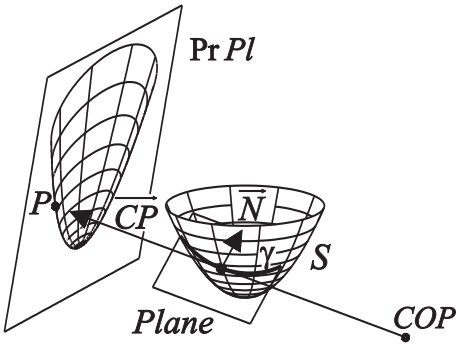

Fig. 10. The definition of a contour point P

We only use condition (4.1) to draw contour lines, since checking condition (ii) is very time consuming and would only exclude rare cases (Figure 11).

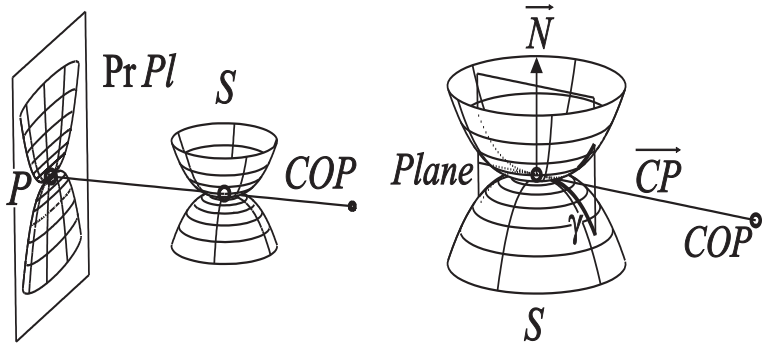

Fig. 11. A phantom point P **that satisfies (4.1), but not (ii)**

The *contour line of a surface* is the set of all its contour points.

Since the contour line of a surface depends on the geometry of the surface, the methods for drawing contour lines have to be developed separately for each type of surface. To find the contour line of a surface, we have to find the zeros of the real–valued function ϕ of two variables defined by

$$\phi(u^1, u^2) = \vec{N}(u^1, u^2) \bullet \overrightarrow{CP}(u^1, u^2). \tag{4.2}$$

An algorithm for the solution of (4.2) and a method for drawing contour lines in the general case, and their implementations can be found in [2, 3, 5].

5 The interactive graphics

The second version of our software works as a Windows application. It allows a user to work independently of any other software package and without any programming.

After the initial *introductory window*, usually the *Selection window* appears. There we can choose one of the geometrical objects which are available in the application. After pressing the button *Select*, the *Object window* appears. There is one object window for each geometrical object which can be selected. They are the most important windows of the application, because all the parameters for the appropriate object are chosen there. After pressing the button *Draw*, the *Image window* appears and the drawing process automatically starts.

The selection window and the object windows are organized in the *page controls* due to the large number of *controls* in them. Each page control has a number of *tab sheets* organized by the subject and role which they have in the application. They also have the *navigation bar* which consists of a few buttons and the *status bar* with the time displayed. The buttons are usually the button *Main* for returning to the initial window, the button *Select* for returning to the select window and the button *Exit* for quitting the application.

5.1 The selection window

The selection window is needed for the selection of the graphical object available in the application. Together with the selected object, a short description apperas of the object and its main parameters. This and the button Select are on the first tab sheet of the page control of the selection window. The second tab sheet contains a short explanation of what the software needs with the explanations of the parameters needed in the object windows the user may not be familiar with. On the last tab sheet the user can set the default parameters for the drawing, such as the line and background colors, the thickness of the lines and the resolution of the image.

5.2 The object window

The object windows are organized in a number of tab sheets which may be different depending on the geometrical object to be represented. By now we have Windows applications for several

- 2D Curves given by a parametric representation,
- 2D Curves given by an equation,
- 3D curves given by a parametric representation,
- Surfaces given by a parametric representation.

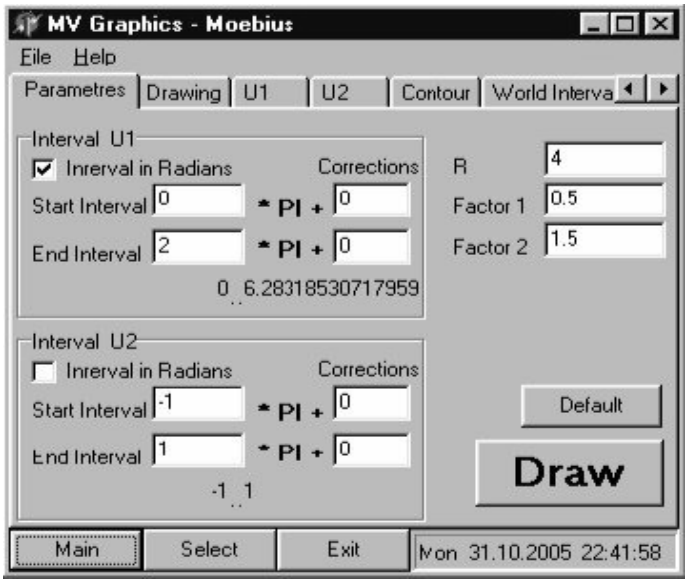

Fig. 12. A tab sheet Parameters

In general they contain

- Tab sheet *Parameters*,
- Tab sheet *Drawing*,
- Tab sheet *World Interval*,
- Tab sheet *Output*,
- Tab sheet *Description*.

The tree–dimensional objects also contain the tab sheet *Projection*. Surfaces have separated tab sheets for the u^1–, u^2– and contour lines.

The tab sheet Parameters is for settings of the mathematical parameters of the object with the intervals of all parameters. These tab sheets can be very different, because they depend on the choice of the geometrical object. An example of a tab sheet Parameters is given in Figure 12.

The tab sheet Drawing is for setting the parameters for the drawing. These are, for instance, the background colour and the resolution of the image. The visibility, colour and thickness of each line to be drawn, number of subintervals and interpolation steps for each line are also parameters for the drawing, but, for the surfaces, they can be on a separate tab sheet. An example of a tab sheet Drawing is given in Figure 13.

The tab sheet World interval is for setting the parameters of the interval of the object space we want to represent. The parameters can be given in absolute values, but we can also let the computer find the smallest world interval which contains the desired object. Also, there is the option to put margins around the object.

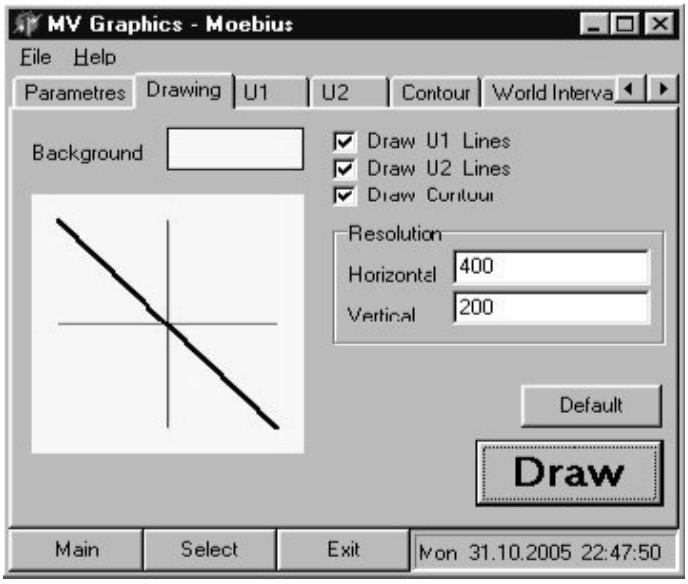

Fig. 13. Tab sheet Drawing

The tab sheet Output allows the user to choose a file name for the output files of the extensions PS, PLT, SCR and GCLC, and which of these files he wants to generate. The files are generated at the same time the image is produced in the Image window.

The tab sheet Description contains the same information as the short description of the object in the selection window. The purpose is not to have to switch to the other window while in the object window.

Default values are available for all parameters, so the users can have their first image without making their own choice. This is very helpful in the beginning before the users are familiar with all the parameters.

Also, there is a control that the values entered by the users are of the right type, in the appropriate range, and consistent with the other values.

The parameters can be saved in a file with extension MVG and loaded from it any time. This is useful when the user is satisfied with the obtained image and wants to save the parameters by which it is produced.

5.3 The drawing process

The drawing process works in the same way as in the Pascal environment (Section 2). The image is drawn in the Image window. Since it may take very long to draw certain complex figures, the time when the process started appears in the window. At the end of drawing process a message of the time needed appears. After that the control is returned to the object window.

Every image can be saved as a BMP file in the Image window. This is because bitmap images are saved in the different way from vector images.

Acknowledgement. Work supported by the research projects #1232 and #1646 of the Serbian Ministry of Science, Technology and Environment, and *Multimedia Techonolgies in Mathematics and Computer Science Education* of the German Academic Exchange Service (DAAD).

References

[1] M. Djorić, P. Janičić, *Constructions, instructions, teaching mathematics and its applications*, Oxford University Press, Vol. 23, 2 (2004), 69–88 also available at `http://www.matfbg.ac.yu/~janicic`

[2] M. Failing, *Entwicklung numerischer Algorithmen zur computergrafischen Darstellung spezieller Probleme der Differentialgeometrie und Kristallographie*, Ph.D. thesis, Giessen, Shaker Verlag Aachen, 1996.

[3] M. Failing, E. Malkowsky, *Ein effizientes Nullstellenverfahren zur computergraphischen Darstellung spezieller Kurven und Flächen*, Mitt. Math. Sem. Giessen 229 (1996), 11–28.

[4] P. Janičić, I. Trajković, *WinGCLC–a workbench for formally describing figures*, Proceedings of the Spring Conference on Computer Graphics (SCCG 2003), April 24–26, 2003, ACM Press, New York, USA; also available at `http://www.matfbg.ac.yu/~janicic`

[5] E. Malkowsky, *An open software in OOP for computer graphics and some applications in differential geometry*, Proceedings of the 20th South African Symposium on Numerical Mathematics (1994), 51–80.

[6] E. Malkowsky, *A software for the visualisation of differential geometry*, Visual Mathematics 4, 1 (2002), Electronic publication, `http://www.mi.sanu.ac.yu/vismath/malkovsky/index.htm`

[7] E.Malkowsky, *Visualisation and animation in mathematics and physics*, Proceedings of the Institute of Mathematics of NAS of Ukraine (50)(3) (2004), 1415–1422, `http://www.imath.kiev.ua-snmp2003/Proceedings/Proceedings2003.html`

[8] E.Malkowsky, W. Nickel, *Computergrafik und Differentialgeometrie*, Vieweg–Verlag, Braunschweig, 1993.

[9] E.Malkowsky, V.Veličković, *Analytic transformations between surfaces with animations*, Proceedings of the Institute of Mathematics of NAS of Ukraine (50)(3) (2004), 1496–1501, `http://www.imath.kiev.ua-snmp2003/Proceedings/Proceedings2003.html`

Author's address:

Vesna Veličković
Department of Mathematics, Faculty of Sciences and Mathematics,
University of Niš, Višegradska 33, 18000 Niš, Serbia and Montenegro.
email: vvesna@BankerInter.net