Original scientific paper
UDK: 655.2

Designing Raster Cells as the Basis for Developing Personal Graphic Language

Jana Z. Vujić[1], Aleksandra Bernašek[2], Tajana Koren[2]

[1]Polytechnic of Zagreb, Croatia
[2]Faculty of Graphic Arts, Zagreb, Croatia

Corresponding author: Jana Z. Vujic,
email: janazv@tvz.hr

Abstract
Continuous work in creating new designer solutions points towards the need to create personal routines as personal communication in the relation comprising design, algorithms, and original computer graphics. This paper shows procedures for developing a control language for creating graphic designs with individual raster elements (screening element obtaint by halftoning). Personal commands should set routines in a language understood by the printer and the designer. The PostScript basis is used because we mix vector and pixel graphics in the same program stream, as well as different colour systems, and our own raster forms. The printing raster is set with the target of special design multi-use, and this includes the field of security graphics and art computer reproduction. Each raster form assumes modifications, creating their raster family. The raster cell content is transformed with PostScript, allowing the setting of basic values, angle and liniature for each pixel separately. Raster cells are mixed in multi-colour graphics to the level of individual designs with variable values of parameters determining them.

Key Words: vector graphics, pixel graphics, personal computer routines, individual screening

Graphic Language Development

Postscript offers us only basic commands and procedures to develop our own program language. The typical command system on basis of Postscript is the EPS printout of images created in Photoshop, Indesign, Freehand. Each of the listed programs developed its own manner of creating the library of computer procedures. Computer graphic designers were free to create all the values for describing image forms. Even the simple elements such as squares must be programmed. Each new routine is joined by new application instructions. The control command selects data from the stack top. Such data should be positioned at certain locations before their activation. A variable may use other variables as sub-actions. The order of taking data from the stack top must be respected. Seldom used variables are named with words that are of domestic expression.
Adobe Photoshop, Acrobat destiler, GSview or Ghostscript programs are used for carrying out graphics that are described with personal routines. The last program listed can only display the graphics, but it has a unique advantage. While it is being carried out, it gives information on the variables condition, their values and possible reasons for unwanted algorithm derivation stopping, i.e. the error diagnosis. Such display is not possible with other programs. We suggest adopting the routines that may be used in other programs, with the development of new screen forms being our prerogative goal.

First received: 16.04.2011.
Accepted: 30.05.2011.

Personal Routines

Let us set the variables: Square shape. Raster form. RGB and CMYK colours. Random number. Printout. Font.

Further down are commands for the announced routines:

(1)

```
%%BoundingBox: 0 0 800 1200
%%EndComments
%%EndProlog
```

(2)

```
/s { mark pstack pop } def
```

(3)

PostScript individual raster (screen) form relations:

(*„Propeller“*)

```
/propeler {dup 2 index 3 exp sub abs sqrt 3 1 roll 3 exp add abs sqrt exch sub abs 1 exch sub} bind def
```

(*„Cut glass“)*

```
/izrezanostaklo {dup 2 index dup dup mul mul mul abs sqrt 3 1 roll dup dup mul mul mul abs sqrt exch sub abs 1 exch sub} bind def
```

(*„Coffee“*)

```
/kava {dup 2 index 3 exp sub abs sqrt 3 1 roll 3 exp sub abs sqrt exch sub abs 1 exch sub} bind def
```

(*„Bat“*)

```
/sismis {dup 2 index 3 exp abs sub abs sqrt 3 1 roll 3 exp abs add abs sqrt exch sub abs 1 exch sub} bind def
```

(„*Circle*“)

```
/krug {dup mul exch dup mul add 1 exch sub} bind def
```

(„*Ellipse*“)

```
/elipsa {0.8 mul dup mul exch 0.5 mul dup mul add } bind def
```

For comparison reasons the round dot form has been added found in the conventional programs for image processing

(4) /RGB {r g b setrgbcolor} def

(5) /CMYK {c m y k setcmykcolor} def

(6) /HSB {h s b sethsbcolor} def

(7) /kvadrat {4 2 roll moveto exch dup 0 rlineto exch 0 exch rlineto neg 0 rlineto closepath fill} def

(8) /font {/Times-Roman findfont 180 scalefont setfont} def

(9) %%EOF

Explanation for the planned routines:

(1)

The demonstrated commands are necessary in order to be able to use PostScript databases in other applications. We set the working area size with the BoundingBox command. In this proposal alignment to the horizontal axis is 800 pixels, and 1200 pixels to the vertical axis.

(2)

A short procedure named „s“ is shown for printout of the stack condition and the text „mark“. A stack is the storage from which we eliminate data in the opposite order in respect to the order by which we had initially stored it. The interpreter gives parameters and names from the program that we had allotted ourselves, and places them at the stack top, pushing further down the previous values in the stack by one position under. The „s“ procedure is suggested as the source of information for values we doubt in while the program is developing.

(3)

Illustrations of the procedures named propeler, izrezanostaklo, kava and sismis are individual rasters. The names have been given at random, whereas the routines have been set by arithmetic operators. Individualized raster elements have been researched by the Mathematica program and later on they were translated into PostScript.

Every setscreen command activating has two x and y values at that moment when found at the stack top. After the covering capacity estimate has been carried out, those values disappear from the stack top. Variables x and y appear several times in raster forms kava and sismis and this is provided by commands roll and index.

(4)/(5)/(6)

The RGB system is defined by three parameters that are assigned in interval 0.0 to 1.0, and this informs that the coverage is 100%. The same interval applies to process dyes in printing, the four-channel CMYK system. Conversion between these two systems is carried out automatically, at the moment when sent to print. The HSB colour system is different because we have control with the help of the following parameters: colour lightness, colour tone and saturation. It is also set in the 0.0 to 1.0 interval. The transition from the RGB system into the CMYK system is also on basis of typical academic relations:

$R = 1 – C;$

$G = 1 – M;$

$B = 1 – Y;$

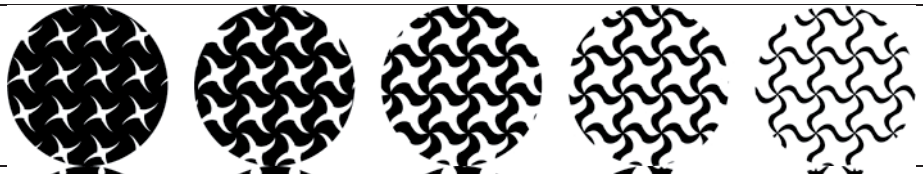
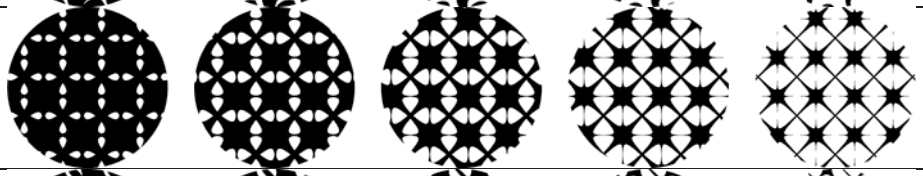
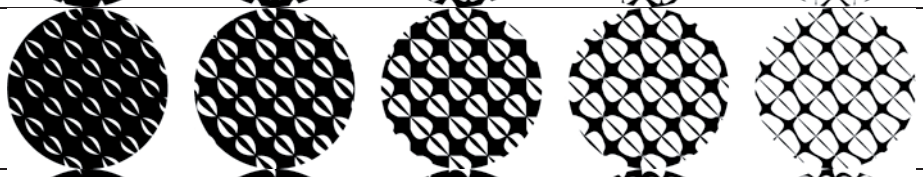
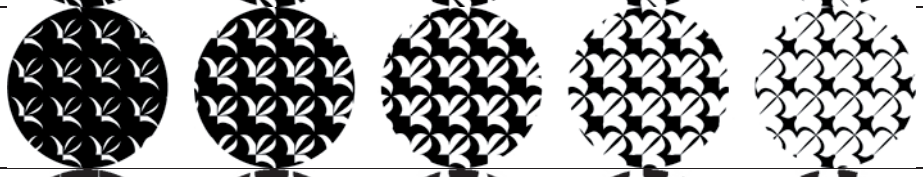
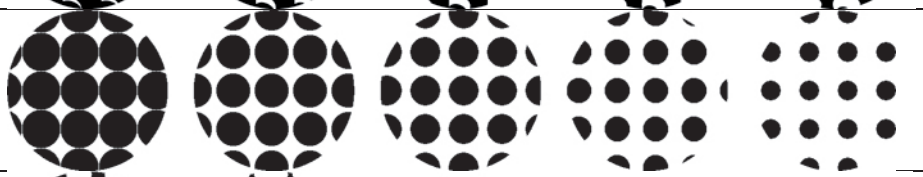
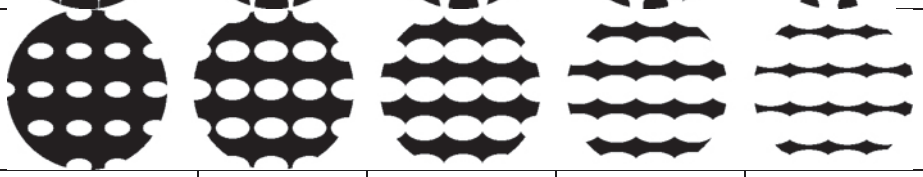
$K = (C + M + Y)$ conventional separation definitions

PostScript does not use any coloursetting and it is the only program tool that enables separation independent of the printing conditions. It is necessary to program this transformation for each adaptation to the real colourant and paper characteristics.

(7)

The routine by which a square is drawn is found under the name kvadrat . The square, ordinarily a simple geometrical form is defined in PostScript with a set of commands. In order not to repeat several times the same commands in programming, short procedures are created. Data from the stack top are duplicated and the

Table 1. Examples of individualized rasters used, liniature decreases from left to right

	Raster (screen) forms				
/propeler					
/izrezanostaklo					
/kava					
/sismis					
/krug					
/elipsa					
covering	75%	60%	45%	30%	15%

routine named roll is used for this. The advantage of such programming is that a complex program can be created with only a few lines.

(8)

The digitalized Times New Roman font is found in the procedure named font. We use this font because it is a standard one and the most often used. When setting fonts, we use the name of the chosen font and its size, i.e. the height of the digital four-component group. A letter character is found inside the four-member group. Most of the letter characters are fully set inside the four-member group; only rare signs fall outside the borders, and some fall out completely outside the four-member group. Other commands such as findfont, scalefont and setfont are an obligatory part of the code for setting the font. The command findfont defines searching for the set font with the command, and by the command scalefont the height of the digital four-member group is set in dots, and setfont activates the font display.

(9)

EOF command is abbreviated for the English sentence „*end of file*" ,and it signals that the program is ending.

Raster (Screening element obtaint by halftoning)

Program procedures for making raster forms according to Table 1:

```
linijatura kut {propeler} bind setscreen
/gray 0 def /pomak 70 def
50 50 translate
5 {/pomak pomak 70 add def
pomak 0 30 0 360 arc
/gray gray 0.15 add def
gray setgray  fill } repeat
```

The raster in elliptical form is an example of the conventional round form mutation with the addition of data deformation at the stack top. This modification is given here as fixed flattening values of the circle into ellipse transition amounting to 0.8 and 0.2 . The result can be described as coverage Z mathematically defined as:

$Z = x^2 + y^2$ for the circle, and $z= 0.8*x^2 + 0.5*y^2$ for the ellipse.

PostScript expressions are given in (3) program base relations. We shall add the information that the Post-Screen command setscreen stops operation if value Z computes a value higher than one. Therefore it is recommended to carry out activities at the stack top with values lower than one if coverage is interpreted in a whole row of values from zero to full coverage. It is also simple and acceptable with functions that add up the square values that are positive by themselves and are less than one.
But, in other raster forms such as in propeler where exponentiation and square root extraction are applied, as well as positive and negative relations, long term experimental work in mutational operations is needed. The domain of all variables is within the positive Z value, and in the range of minus one to plus one for coordinates x and y. The raster named elipsa is given in the negative. The remainder is realized in the raster cell that is calculated as coverage.

Vector graphics with individualized rasters

By combining vector and raster graphics we demonstrate the manner in which such symbiosis can be used to create a unique whole with the goal to achieve the utmost uniqueness and quality of a certain graphic.
In the example to follow we are showing how it is possible to apply different raster elements in typographical forms. In doing so we are not using conventional raster forms. Letter characters NS create the already mentioned masks, and they are defined with the help of the PostScript font interpretation of Times New Roman. The design within the mask is created from four individual rasters with randomly set liniature and angle. Mathematical relations for the raster are installed in the codes for each CMYK channel separately. Rasters in typography are an excellent solution if we wish to individualize our design in full. We can implement a unique raster into each letter character, whereby we create a unique product and excellent protection.

Figure 1. Graphic NS. Separation per CMYK channels with different set rasters.

Separation per CMYK channels shown in Figure 1. is the first step in creating the graphic in Figure 2. With the help of the programs that enable reading of the set formats, the graphic transforms, and the channels connect in the following order: yellow, magenta, cyan and black.

Figure 2. Applying of rasters in typography, a different individualized raster is applied in each CMYK channel

The graphic in Figure 1. are Latin characters NS for which we formed the clip mask. The randomly chosen individualized rasters are well observed within the mask. The rasters used are defined under the names propeler, izrezanostaklo, kava and
Sismis. The liniature and angle alterations are set with parameters, so that a different raster with different mathematical values is set for each CMYK channel.

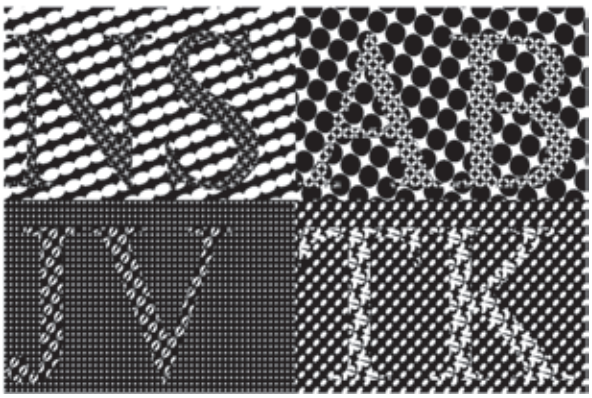


Figure 3. Display of rastered vector graphics, 1bitna slika 0, 1(crno - bijelo)1 bit image 0.1 (black and white)

Figure 4. Display of rvector graphics rastered in colour, 8 bit image (256 gray tones + white and black)

Program procedures for rastering vector formed graphics:

```
 /linijatura 15 def /kut 90 def
linijatura kut {krug} bind setscreen
0.2 0.1 0.0 setrgbcolor 50 40 250 150 kvadrat
/linijatura 8 def /kut 45 def
linijatura kut {elipsa} bind setscreen
0.6 0.1 0.1 setrgbcolor 300 40 250 150 kvadrat
/linijatura 4 def /kut 60 def
linijatura kut {krug} bind setscreen
0.5 0.1 0.3 setrgbcolor 300 190 250 150 kvadrat
/linijatura 5 def /kut 20 def
linijatura kut {elipsa} bind setscreen
0.0 0.8 0.5 setrgbcolor 50 190 250 150 kvadrat
50 200 moveto /linijatura 10 def /kut 40 def
linijatura kut {propeler} bind setscreen
/h 0.6 def /s 0.9 def /bb 0.45 def
font (NS)  hsb show
300 50   moveto /linijatura 5 def /kut 30 def
/r 0 def /g 0.8 def /b 0.6 def rgb
linijatura kut {sismis} bind setscreen
font (TK) show
300 200   moveto /linijatura 10 def /kut 40 def
linijatura kut {izrezanostaklo} bind setscreen
/h 0.7 def /s 0.6 def /bb 0.85 def
font (AB)  hsb show
50 50 moveto /linijatura 5 def /kut 10 def
0.1 0.5 0 0.4 setcmykcolor
linijatura kut {kava} bind setscreen
font (JV) show
```

The graphics from Figures 3 and 4 were derived from the same source. The program code is identical in both the cases. The difference is in the display of program code GSview with interpreter. In the first example (Figure 3.) GSview program is set for displaying a 1-bit image, it recognizes only black and white (0,1), and the set rasters are very well observed. In the second example (Figure 4) GSview is set for displaying a 8-bit image (256 tones), and it recognizes and displays different colour shades. We used different colour systems to carry out this graphic. The substrate is made of two rasters: krug i elipsa. Each square is an object by itself and has a different liniature, angle and colour. The colour is set by the RGB system.

The NS graphic on the green substrate (above left) was created by using the CMYK colour system. By changing variables it was achieved that the edges of the raster element propeler are outlined and seen in a somewhat darker tone than the rest of the display. The HSB colour system was used in creating the graphic on the dark red substrate (above right) and an interesting 2D structure was created. The graphic rastered by kava is situated on the pink substrate (down left) created by using gray tones (Grayscale). The NS graphic is situated on the red substrate (down right) in green tones achieved by the RGB colour system.

It is important that the bmpmono format should be set with the first example (Figure 3) when converting the image in the GSview interpreter, whereas format bmp256 is set for the second example (Figure 4) for the same program.

Program procedures:

Pixel Graphic with Individualized Screnning

The key element for achieving the maximum in creativity and expression with individualized rasters is producing a photograph with different rasters for each C, M, Y, and K channel. We prepare the chosen photograph in the programs that are provided for adjusting format parameters necessary to carry out the designs. In order to obtain the best possible visual effect, a different mathematical raster relation is implemented into each of the CMYK channels. The PostScript interpretation of a raster element is derived from chosen trigonometrical functions. Low liniature is applied and the designs have raster elements visible by the bare human eye. There is an important reason for applying low liniature, and that is in order to show the micro structure of certain graphics. We can thus use our computer to complete our designs and fully individualize a graphic design.

Figure 5. Applying rasters in pixel graphics. A different raster is used for each CMYK channel

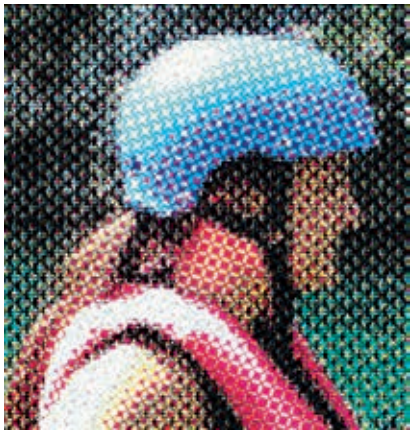


Figure 6. A blown up detail from Figure 5.

Program procedures:
30 600 translate
3 3 scale
/linijatura 10 def /kut 75 def %cijan pod kutom od 75°
linijatura kut {propeler} bind setscreen
%sirina 80 piksla, visina 84 piksla
80 84 8[1 0 0 -1 0 0]
{<8D8E94999A9B99949393919……..28EA7D3D7A13053867B6552575450>}image

A digital photograph is divided into image elements. The image element is set with only one numerical coverage value for each channel separately. With CMYK channel separation we create program codes into which we implement a PostScript interpretation of the mathematical formula for certain functions. Lin and kut are variables with which we set liniature and angle parameters and we call them by their names bind setscreen.

Figure 7. Separation per channels. From left to right (above); channel cyan, raster propeler, channel magenta, raster izrezanostaklo. From left to right (below); channel yellow, raster kava; channel black, raster šišmiš.

Procedure:

The chosen photograph is prepared in the program for photo processing. The pixel dimensions and the resolution for printing display are set. The important thing is for the photograph to have a four-channel separation, i.e. – that it is stored in the CMYK colour system. The photograph is stored in the DCS 2.0 (*.EPS) format by which the program itself creates streams along process channels. The streams are stored as simple *.txt format, where we have a PostScript code set for each channel.

We add are own commands to the obtained separations (pages 1 and 2). We will set a different raster, liniature and angle for each separation, at random. The preliminarily set pixel dimensions must be the same for each of the CMYK channels.
The obtained separations are opened in the GSview program used for displaying PostScript documents. For instance, separation of cyan before converting must have a set depth of 1 bit/pixel, so that the raster elements are visible. At conversion the format is set to bmpmono, while the resolution is random. The obtained *.bmp formats enable reading of the image in various raster and vector programs. In order to obtain a result from Figure 3., we create a new document in the CMYK system. Bitmapped images are opened in the program for photograph processing, and then they are positioned in their channels one by one, with certain preliminary actions. Before copying, the separations must be translated into gray tone, and the unneeded white parts should be cut off. The order of implementing bitmapping separations is starting from the yellow channel, then magenta, cyan and black.
The graphic design in this example is carried out through separation along CMYK channels. Mathematical relations of raster elements translated into PostScript language are implemented into a code for each separation separately. The possibilities of combining raster parameters in pixel graphics are more numerous: deformation of the pixel square structure, pixel deviation, shift from the basic position, rotation and choice of raster structure for each pixel separately. Such transformations are not possible in other image processing programs.
The graphic goes through separation along CMYK channels. The raster element PostScript algorithm is implemented inside each channel. The rastered portrait separations are divided into four identical parts with the help of the photograph processing program, in the following order: C, M, Y, K. In the new document the separations are united into one whole. The work comprises all technological process elements; from photograph processing to raster system implementation. In case we implement into our design random numbers for angle, liniature and pixel deformation, and we initiate a pseudo-random sequence, then we give a new dimension to each graphic design.

A black and white photograph divided into parts:
The graphic in Figure 8 is a program design created by combining 6 rasters described in the paper. After every 12 rows the raster form is changed. If listing from top to bottom, the following rasters are found: propeler, izrezano staklo, kava, šišmiš, krug and elipsa. The parameters are the same for all rasters; a 10 lin/cm liniature and an angle of 45° are set. The rastering procedure with white and black photographs is somewhat equalled with the “image“ routine.

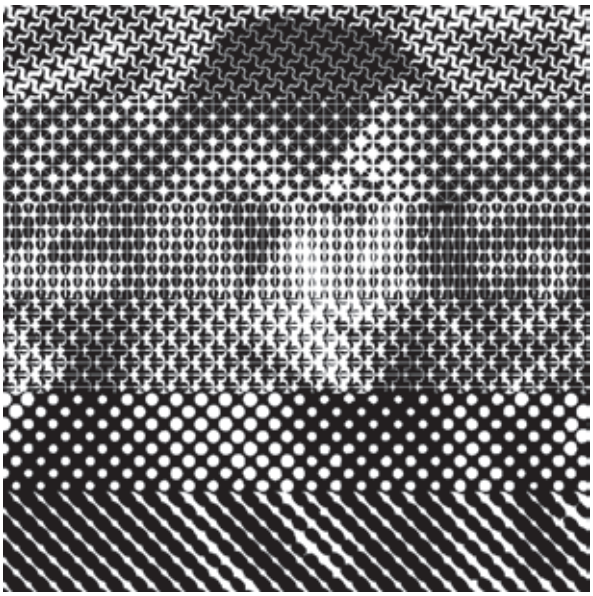


Figure 8. A black and white pixel graphic divided into 6 equal parts

Program procedures for the image in a 8 bit record, 80 pixels horizontally and 12 rows vertically with change of the raster form for the following rows:

```
/linijatura 10 def /kut 45 def
linijatura kut {propeler} bind setscreen
%sirina 80 piksla, visina 80 piksla
80 12 8[1 0 0 -1 0 0]
{<605F61676F809197948B83837F808C93.....
59495BFB6919194A5ADABBB>} image
linijatura kut {izrezanostaklo} bind setscreen
80 12 8[1 0 0 -1 0 0].................
```

Conclusion

Computer graphics are based on algorithms developed as a set of routines in transforming mathematical relations in a program stream for carrying out complex printing preparations. This is a wide area and provides the designer with adequate means to create his personal graphic language. A set of raster forms is given that have been tested in borderline coverage conditions. PostScript is the basis for carrying out printed graphic, from makeup, making the printing forms, all the way to printing, especially the one represented as digital printing. Almost all programs have strict preferences. They are either only pixel or vector oriented, and simultaneously support a limited system of colour and raster form use. PostScript enables fellowship of pixel and vector graphics within the same design. Also, mixing of different colour system definitions is organised : (RGB, CMYK and HSB) within the same program. Uniqueness is achieved with PostScript commands as a constant library for carrying out complex raster designs. Individual rasters implemented into the graphics are a new manner of presenting security and art reproductions. Raster elements were created with the development of mathematical relations of targeted functions that are subjected to strict domains of unit raster cells. The goal in presenting individual rasters is to encourage designers in creating new authorized raster forms by altering parameters that act as deformation and mutation between two or more raster designs.

Literature

1. Agić D., Strgar Kurečić M., Mandić L., Pap K. (2009) Black separation strategies in colour reproduction. In: Katalinić B. (Ed.) DAAAM International Scientific Book 2009,Vienna, Austria, DAAAM International, pp. 001-008.
2. Pap K., Žiljak V., Vujić J. Ž. (2008) Design of Digital Screening, Zagreb, FS d.o.o.
3. Sabati Z. Pogarčić I., Vujić Ž. Jana, Pap K., Žiljak V. (2007) Protection of information in documents by implementing individual rastering:Proceedings of the 18th International Conference on Information and Intelligent Systems, Aurere B., Bača M. (Ed.), Varaždin, Faculty of Organization and Informatics, University of Zagreb. pp 299-302
4. Žiljak V., Pap K. (1998) Postscript programiranje grafike. Zagreb. FS d.o.o.