LOGO TO POSTSCRIPT

Vladimir Batagelj
Department of mathematics, FMF, University of Ljubljana
Jadranska 19, 1000 Ljubljana, Slovenia
e-mail: `vladimir.batagelj@uni-lj.si`

Abstract

A picture produced by logo on the screen can be mirrored into its PostScript description on the file. These files provide a high quality records (snap-shots) of logo screen to be included in other documents.
`Logo2PS` is a collection of logo commands that by redefining turtle commands support the mirroring of turtle movement in the PostScript.

Keywords

`Logo, PostScript, screen snap-shot, redefining logo primitives`

1 Introduction

The usual approach to get snap-shots of logo screen for inclusion in different documents (manuals, tutorials, papers, books) is to save (a part of) the screen as a bitmap. Adjusting such bitmap picture to the dimensions required by a document we can considerably lose on its quality.

In this paper another approach is proposed. A picture produced by logo on the screen can be mirrored into its PostScript description on the file. These files provide a high quality snap-shots of logo screen and can be included in other documents, clipped and scaled to the available space (see Figure **1**).

2 Redefining logo primitive commands

Berkeley logo [7] and its adaptation for Windows, MSWlogo [9] allow the user to redefine also primitive commands. In general this is a dangerous, but sometimes very useful, practice. To

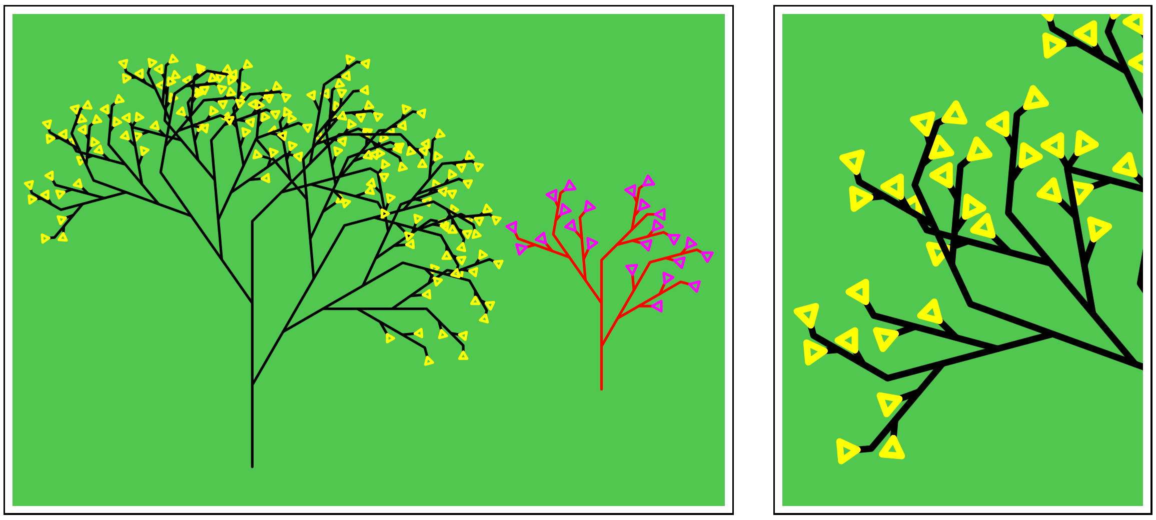

Figure 1: Trees.

enable this possibility we have first to switch-on the special (system) variable `REDEFP`

```
MAKE "REDEFP "true
```

To redefine commands we shall use two logo commands

`COPYDEF` *newname oldname*

that makes *newname* a procedure identical to *oldname*; and

`DEFINE` *procname* `[` *params body* `]`

that defines a procedure with name *procname*, parameters list *params* and *body* – sequence of lists containing instruction lines.

Since we want to add the generation of PostScript code to the turtle moving and related commands, the general pattern of redefinitions will be

`COPYDEF ".`*cmd* `"`*cmd*
`DEFINE "`*cmd* `[ [` *params* `] [` *ext1* `] [ .`*cmd params* `] [` *ext2* `] ]`

We first save the original definition of the primitive command *cmd* as `.`*cmd*. Afterwards we redefine *cmd* by surrounding the application

`.`*cmd params*

of original command with additions *ext1* and *ext2*.

This approach can be used also for other purposes. For example, in the current (version 5.2c) implementation of MSWlogo the function `SQRT` (and also `LN` and `LOG10`), in the case of negative argument, does not handle the error by an internal `THROW "ERROR`, as one would expect. We can achieve this behavior by redefining it:

```
COPYDEF ".sqrt "SQRT
DEFINE "SQRT [[x]
  [IFELSE :x < 0 [THROW "ERROR][OUTPUT .sqrt :x]]
]
```

3 PostScript

PostScript is a graphics programming language for describing, in a device-independent manner, text and other graphical objects and how they are placed on page(s)/screen. It was developed in 1985 by Adobe Systems in a joint project with Apple Computer on the development of Apple LaserWriter. This version is known as PostScript Level 1. At the first PostScript Conference in 1990, PostScript Level 2 was announced which integrated several improvements into a new version of the PostScript language.

A PostScript program is a text (ASCII) file. Usually it is produced by some other graphics or text formatting program (Word, Word Perfect, Corel Draw, Mathematica, ...), but it can be also prepared and maintained by user by any text editor. The simplest way to display the results of a PostScript program on *file*`.ps` is to send it to a PostScript printer (`copy` *file*`.ps lpt:` or `print` *file*`.ps`).

PostScript programs are interpreted by an interpreter built in a display device (i.e., laser printer) or by a software interpreter in the user's computer. The most widespread software PostScript interpreter is Ghostscript (Aladdin Enterprises and Free Software Foundation). Ghostscript 4.03 (September 1996) implements PostScript Level 2. Ghostscript enables us to preview the PostScript documents on the screen and to print them on several nonPostScript printers.

A PostScript program starts with

```
%!PS
```

followed by the description of page(s). PostScript recognizes, besides a printable subset of the ASCII character set, also characters *space*, *tab* and *newline* (CR or LF or CR LF). The content of the line from `%` till the end of line is a comment.

PostScript is a stack based language and uses a postfix (reverse Polish) notation for commands

$$p_1\ p_2 \ldots p_n\ cmd$$

The interpreter puts the arguments $p_1, p_2, \ldots, p_n$ on the stack and leaves the results of command cmd on it.

PostScript is a powerful programming language which besides general programming elements: data types (integer, real, boolean, string, array, dictionary, file), control statements, arithmetic operations and functions, operations and functions on other data types, conversion operators, stack commands, environment commands; contains also many specific graphics commands: coordinate system changing commands, path drawing commands, attribute setting commands, font commands, and displaying commands.

PostScript's own coordinate system is based on units called points (72 pt = 1 inch). It has the origin (0,0) in the lower left corner (letter = 8.5 $\times$ 11 inch = 612 $\times$ 792 pt; A4 = 21 $\times$ 29.7 cm = 595 $\times$ 842 pt). The content of the page is composed of page elements – parts of pictures or text. Each page element is determined by set of paths (lines, arcs, curves) and their properties which are realized after the application of some displaying command. The page is displayed by command `showpage`. Characters are also treated as pictures, but supported by a special set of very efficient commands.

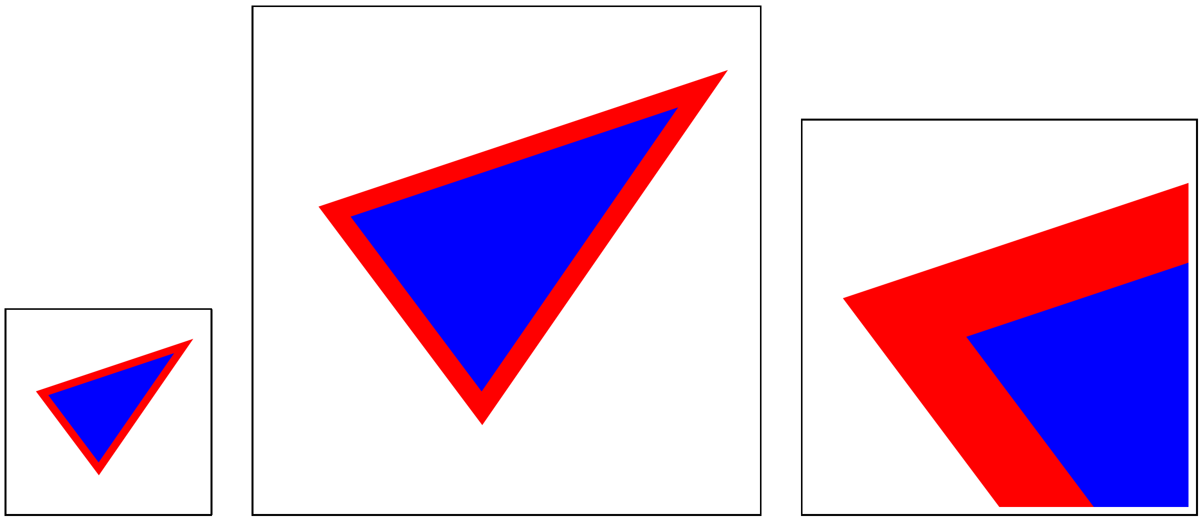

Figure 2: Example.

PostScript program lines beginning with `%%` are special comments intended for the programs (previewers, utilities) processing PostScript programs. The rules determining the structure and meaning of these comments are known as DSC – Document Structuring Conventions. The first line `%!PS-Adobe-3.0` tells us that the program conforms to DSC – version 3.0. An example of a previewer using DSC comments is GsView (by Russell Lang) which by using `%%Page:` comments allows us to see or print the selected page(s).

Encapsulated PostScript format is used to import already prepared parts of the picture into a document. The EPS file should contain only one page and shouldn't use the operators that would perturb the graphics environment of the surrounding PostScript. It usually starts with a line

```
%!PS-Adobe-3.0 EPSF-3.0
```

and one of the following lines should be

`%%BoundingBox:` $ll_x\ ll_y\ ur_x\ ur_y$

which defines the bounding box of the picture on the file.

Let us look to a simple example (used in Figure **2**):

```
%!PS
%%BoundingBox: 0 0 100 100
15 60 moveto  90 85 lineto  45 20 lineto
closepath
gsave
  0 0 255 setrgbcolor  fill
grestore
4 setlinewidth  255 0 0 setrgbcolor  stroke
showpage
```

The first line of the program declares that this is a PostScript program. In the second line we define a 100×100 square to be the bounding box of the picture. In third line we first move to the point $(15, 60)$, then we make a line to the point $(90, 85)$ and again a line to the point $(45, 20)$ and close the path (return to the starting point $(15, 60)$). We constructed a triangle. The command `gsave` saves the current graphic environment. We fill the interior of the triangle with blue and restore the graphical environment. Now we set the line width to 4pt, its color to red and draw the triangle. It has to be emphasized that path constructing and attribute setting commands create only descriptions of paths which are not realized on the page until some displaying command (`stroke` or `fill`) is issued. The command `showpage` at the end of the page requires that the interpreter displays the page.

4 Logo to PostScript

To use Logo2PS we simply load the `Logo2PS.LGO` file. It defines the following commands:

`PSInit`
: Redefines turtle commands to mirror the turtle movement in the PostScript.

`PSExit`
: Erases Logo2PS commands and variables.

`PicHead :p :xll :yll :xur :yur :d :c :j`
: Initializes the file p`.EPS` for a new picture with a screen bounding box $(xll, yll; xur, yur)$. d sets the decimal places of real numbers written to p`.EPS`, c selects the type of line caps (0 – butt cap, 1 – round cap, 2 – square cap), and j selects the type of line joins (0 – mitered join, 1 – round join, 2 – beveled join).

`BegPic :p`
: Essentially calls `PicHead :p -300 -300 300 300 2 1 1`

`EndPic`
: Ends the current picture on the EPS file.

`BegShape :w :pc :fc`
: Starts a new shape with border width w, border color pc and fill color fc. Between `BegShape` and `EndShape` these attributes should not be changed.

`EndShape :t`
: Ends the drawing of a shape. The list t contains logo commands that determine a point in the interior of the shape, required by the logo `FILL` command. After the fill the position before the t moves is restored.

and some auxiliary commands.

To start tracing the turtle movement to the EPS file(s) we execute the `BegPic` *filename* command. Then we run the logo commands that produce the picture. We finish tracing by command `EndPic`

For example:

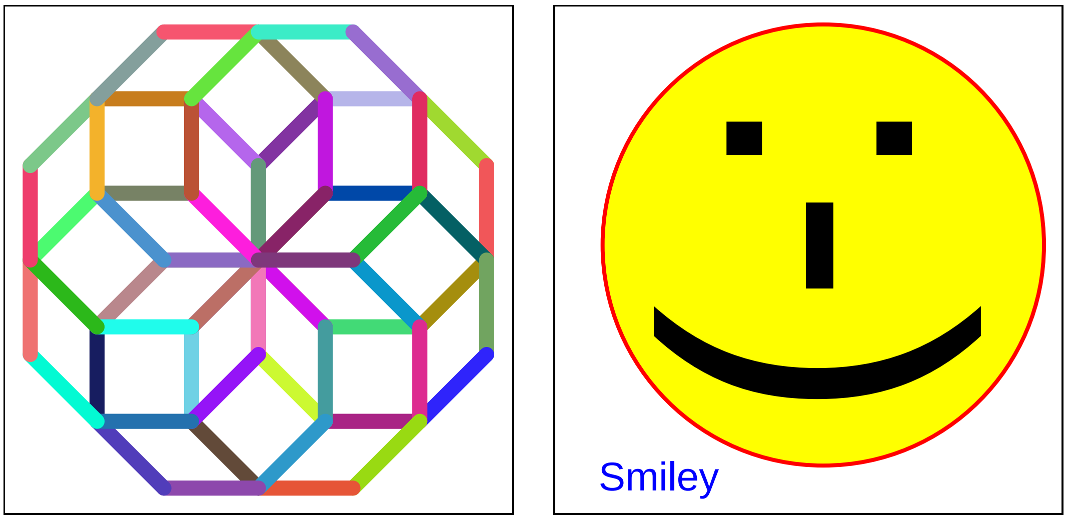

Figure 3: Pictures from examples.

```
    BegPic "test
     CS SetPenSize [8 8]
     REPEAT 8 [ REPEAT 8 [
       SetPenColor (LIST random 256 random 256 random 256)
       FD 50 RT 45 ] RT 45 ]
    EndPic
```

produces the picture presented on the left side of Figure **3**.

Commands `BegShape` and `EndShape` were introduced because the logic of filling in logo is different from that in PostScript. `Logo2PS` partially supports also the use of fonts. The use of both is illustrated with the picture on the right side of Figure **3**, obtained by the following command:

```
TO Smiley
  BegPic "smiley
    BegShape 4 [255 0 0] [255 255 0]
      PU SETPOS [0 0] PD CIRCLE 220 PU
    EndShape [ SETPOS [0 0] ]
    SETPC [0 0 0] SETPOS [208 120] SETHEADING 180
    SETTEXTFONT [[Helvetica] 350 0 0 400 0 0 0 0 3 2 1 34]
    LABEL "|:-)|
    SETPC [ 0 0 255] SETPOS [-220 -210] SETHEADING 90
    SETTEXTFONT [[Times New Roman] 40 0 0 400 0 0 0 0 3 2 1 18]
    LABEL "Smiley
  EndPic
END
```

The implementation details on `Logo2PS` can be seen from its code:

```
; Logo2PS 1.1 - Logo to PostScript
;   15. mar 1997 basic version
;   25. mar 1997 + shapes
;   30. mar 1997 + ellipsearc
; (c)1997 Vladimir Batagelj

TO PSInit
  MAKE "REDEFP "true
; Turtle moves and ellipsearc
  COPYDEF ".forward     "FORWARD
  COPYDEF ".fd          "FD
  COPYDEF ".back        "BACK
  COPYDEF ".bk          "BK
  COPYDEF ".setpos      "SETPOS
  COPYDEF ".setxy       "SETXY
  COPYDEF ".setx        "SETX
  COPYDEF ".sety        "SETY
  COPYDEF ".home        "HOME
  COPYDEF ".clearscreen "CLEARSCREEN
  COPYDEF ".cs          "CS
  DEFINE "FD      [[d][.fd :d .move]]
  DEFINE "FORWARD [[d][.fd :d .move]]
  DEFINE "BK      [[d][.bk :d .move]]
  DEFINE "BACK    [[d][.bk :d .move]]
  DEFINE "SETPOS  [[d][.setpos :d .move]]
  DEFINE "SETXY   [[x y][.setxy :x :y .move]]
  DEFINE "SETY    [[y][.sety :y .move]]
  DEFINE "SETX    [[x][.setx :x .move]]
  DEFINE "HOME    [[][.home .move]]
  DEFINE "CS [[][.cs]
    [IF :.draw [
      IF :.act [PicTail Make ".v :.v + 1
        PicHead (WORD :.p :.v ) -300 -300 300 300 2 1 1 ]]]
  ]
  DEFINE "CLEARSCREEN [[][CS]]
  COPYDEF ".ellipsearc     "ELLIPSEARC
  DEFINE "ELLIPSEARC [[t b a s]
    [ .ellipsearc :t :b :a :s ]
    [IF :.draw [
      IF NOT :.act [PRINT "newpath]
      (PRINT "|matrix currentmatrix|
       FORM XCOR+:.x0 :.m :.d FORM YCOR+:.y0 :.m :.d "translate
       90-HEADING "rotate :a :b "scale)
      (PRINT 0 0 1 180-:t-:s 180-:s "|arc setmatrix|)
      IF NOT (:t < 360) [PRINT "closepath]
      MAKE ".act "true ]]
  ]
```

```
; Pen color and size
  COPYDEF ".setpencolor    "SETPENCOLOR
  COPYDEF ".setpc          "SETPC
  COPYDEF ".setpensize     "SETPENSIZE
  COPYDEF ".setscreencolor "SETSCREENCOLOR
  DEFINE "SETPENCOLOR [[c]
    [.setpencolor :c]
    [IF :.draw [
      IF :.act [PRINT "stroke .porg MAKE ".act "false]
      (PRINT .conv :c "setrgbcolor) ]]
  ]
  DEFINE "SETPC [[c][SETPENCOLOR :c]]
  DEFINE "SETPENSIZE [[s]
    [.setpensize :s]
    [IF :.draw [
      IF :.act [PRINT "stroke .porg MAKE ".act "false]
      (PRINT ITEM 2 :s "setlinewidth) ]]
  ]
  DEFINE "SETSCREENCOLOR [[c]
    [.setscreencolor :c]
    [IF :.draw [PRINT "gsave
      (PRINT .conv :c "setrgbcolor)
      (PRINT "|0 0 moveto|  FORM :.xur :.m :.d 0 "lineto
             FORM :.xur :.m :.d FORM :.yur :.m :.d "lineto
             0 FORM :.yur :.m :.d "|lineto closepath fill| )
      PRINT "grestore]]
  ]
; fonts and labels
  COPYDEF ".label          "LABEL
  COPYDEF ".settextfont    "SETTEXTFONT
  DEFINE "LABEL [[t]
    [IF :.draw [.porg  (PRINT "gsave 90-heading "rotate
        -0.1*:.fs -:.fs*0.87 "rmoveto )
      (TYPE "\( .delim :t "\) ) PRINT "| show grestore| ]]
    [.label :t .porg]
  ]
  DEFINE "SETTEXTFONT [[s]
    [.settextfont :s]
    [IF :.draw [MAKE ".fs ABS ITEM 2 :s
      (PRINT "|/Helvetica| :.fs "selectfont ) ]]
  ]
; header
  PRINT "|Logo to PostScript 1.1 loaded|
  PRINT "|  BegPic "filename  - start picture|
  PRINT "|  EndPic            - end picture|
  PRINT "|  PSExit            - restore standard Logo|
  MAKE "REDEFP "false MAKE ".draw "false MAKE ".act "false
END
```

```
TO PicHead :p :xll :yll :xur :yur :d :c :j
  MAKE ".x0  -:xll MAKE ".xur :xur-:xll
  MAKE ".y0  -:yll MAKE ".yur :yur-:yll
  MAKE ".d :d  MAKE ".m 4+:d
  MAKE ".eps (WORD :p ".eps ) MAKE ".draw "true
  PRINT :.eps OPENWRITE :.eps SETWRITE  :.eps
  PRINT  [%!PS-Adobe-3.0 EPSF-3.0]
  (PRINT [%%Title:] :p )
  PRINT  [%%Creator: Logo2PS 1.1\; Copyright 1997, Vladimir Batagelj]
  (PRINT [%%CreationDate:] time )
  (PRINT [%%BoundingBox:] 0 0 ROUND :.xur ROUND :.yur )
  PRINT  [%%EndComments]
  (PRINT  [%%Page:] (WORD "picture :.v))
  (PRINT :c "setlinecap :j "setlinejoin )
  MAKE ".act "false  MAKE ".fs ABS ITEM 2 TEXTFONT
  (PRINT "/Helvetica :.fs "selectfont )
  .CS .porg
END

TO BegPic :pic
  MAKE ".p :pic  MAKE ".v 1
  PICHEAD :pic -300 -300 300 300 2 1 1
END

TO PicTail
  PRINT "stroke  PRINT "showpage  PRINT [%%EOF]
  CLOSE :.eps  SETWRITE []
END

TO EndPic
  PicTail  MAKE ".draw "false
END

TO BegShape :w :pc :fc
  .setpensize (LIST :w :w) .setpencolor :pc
  MAKE ".w :w MAKE ".pc :pc MAKE ".fc :fc
  IF :.draw [
    IF :.act [ PRINT "stroke PRINT "newpath
      .porg  MAKE ".act "false ]
  ]
END

TO EndShape :t
  (LOCAL "tp "ps "h "dr)
  MAKE "tp POS MAKE "ps PENDOWNP MAKE "h HEADING MAKE "dr :.draw
  SETFLOODCOLOR :.fc PU MAKE ".draw "false RUN :t PD FILL PU
  .SetPos :tp SETHEADING :h MAKE ".draw :dr
  IF :ps [ PD ]
  IF AND :.draw :.act [
    (PRINT "closepath
           "gsave .conv :.fc "setrgbcolor "fill "grestore)
    IFELSE :.w > 0 [
      (PRINT :.w "setlinewidth .conv :.pc "setrgbcolor "stroke)
    ][PRINT "newpath ]
    .porg  MAKE ".act "false
  ]
END

TO .conv :c
  OP (LIST FORM (ITEM 1 :c)/255 6 4
      FORM (ITEM 2 :c)/255 6 4  FORM (ITEM 3 :c)/255 6 4)
END

TO .move
  IF :.draw [
    (TYPE FORM XCOR+:.x0 :.m :.d "\  FORM YCOR+:.y0 :.m :.d "\  )
    IFELSE PENDOWNP [MAKE ".act "true PRINT "lineto][PRINT "moveto]
  ]
END

TO .porg
  IF :.draw [
    (PRINT FORM XCOR+:.x0 :.m :.d FORM YCOR+:.y0 :.m :.d "moveto )
  ]
END

TO .delim :s
  (LOCAL "t "c) MAKE "t "
  REPEAT COUNT :s [ MAKE "c ITEM REPCOUNT :s
    IF MEMBERP :c "|(\)| [MAKE "t (WORD :t "\\ ) ]
    IF :c = "\\ [MAKE "t (WORD :t "\\ ) ]
    MAKE "t (WORD :t :c)]
  OUTPUT :t
END

TO PSExit
  MAKE "REDEFP "true
  ERASE [forward fd back bk setpos setxy setx sety home
    ellipsearc clearscreen cs setpencolor setpc setpensize
    setscreencolor label settextfont begpic endpic pichead
    pictail begshape endshape psinit .conv .delim .move .porg ]
  ERASE [ .act .x0 .y0 .m .d .draw .xur .yur .eps .fs .p
    .v .w .pc .fc]
  COPYDEF "FORWARD ".forward  COPYDEF "FD      ".fd
  COPYDEF "BACK    ".back     COPYDEF "BK      ".bk
  COPYDEF "SETPOS  ".setpos   COPYDEF "SETXY   ".setxy
  COPYDEF "SETX    ".setx     COPYDEF "SETY    ".sety
  COPYDEF "HOME    ".home     COPYDEF "CS      ".cs
  COPYDEF "SETPC   ".setpc    COPYDEF "LABEL   ".label
  COPYDEF "CLEARSCREEN    ".clearscreen
  COPYDEF "ELLIPSEARC     ".ellipsearc
  COPYDEF "SETPENCOLOR    ".setpencolor
  COPYDEF "SETPENSIZE     ".setpensize
  COPYDEF "SETSCREENCOLOR ".setscreencolor
  COPYDEF "SETTEXTFONT    ".settextfont
  ERASE [.forward .fd .back .bk .setpos .setxy .setx .sety
    .home .ellipsearc .clearscreen .cs .setpencolor .setpc
    .setpensize .setscreencolor .label .settextfont]
  MAKE "REDEFP "false
END

PSInit
```

The last version of `LogoPS` is available on:
`http://vlado.fmf.uni-lj.si/educa/logo/logo2ps/logo2ps.zip`

References

[1] Adobe Systems Inc.: *PostScript Language, Reference Manual.* Second Edition. Addison-Wesley, Reading, MA, 1990.

[2] Adobe Systems Inc.: *PostScript Language, Tutorial and Cookbook.* Addison-Wesley, Reading, MA, 1985.

[3] Adobe Systems Inc., Reid G.C.: *PostScript Language, Program Design.* Addison-Wesley, Reading, MA, 1988.

[4] Aladdin Enterprises: *Ghostscript*:
`http://www.cs.wisc.edu/~ghost/aladdin/index.html`

[5] Batagelj V.: *Combining TEX and Postscript.* Proceedings of Eighth European TEX Conference, September 26-30, 1994, Gdansk, Poland, p. 83-90. Updated version, march 1995; Maps #14 (95.1), Nederlandstalige TEX Gebruikersgroep, p. 40-46.

[6] McGilton H., Campione M.: *PostScript by Example.* Addison-Wesley, Reading, MA, 1992.

[7] Harvey B.: *Berkeley Logo*: `http://http.cs.berkeley.edu/~bh/`

[8] Lang R.: *GSView*: `http://www.cs.wisc.edu/~ghost/gsview/index.html`

[9] Mills G.: *Logo (Berkeley) for Windows*, ver. 5.2c. Program help file, 1996.
`http://www.ultranet.com/~mills/logo2.html`