How to Create Your Own Symbols in METAFONT and for use in LaTeX Documents

Kuang-Hsu (Richard) Lin
Department of Software Development
Monash University
PO Box 197, Caulfield East
Australia 3145

email: rlin@fcitds.fcit.monash.edu.au
phone: 61–3–573 2975
fax: 61–3–573 2745

December 21, 1993

Abstract

The aim of this report is to explain how to create your own symbols using METAFONT, and incorporate these with LaTeX. Thereafter, users can create their own style files and do not have to depend on existing ones. As many features which have been introduced in METAFONT, TeX and LaTeX are not explained here, readers should refer to the references for further details.

Contents

1 What is METAFONT? 3

1.1 Introduction . . . 3

1.2 How to bring METAFONT up . . . 3

1.3 How to QUIT METAFONT . . . 4

1.4 Some tips for METAFONT beginners . . . 5

2 Programming in METAFONT 6

2.1 Organization of Files . . . 6

2.2 Number and type of PK files needed . . . 7

2.3 The program file . . . 8

2.3.1 Brief introduction to METAFONT language . . . 8

2.3.2 The file: GISSYMBOLS.MF . . . 8

3 How to Glue all the pieces together 12

3.1 What is TeX? . . . 12

3.2 What is LaTeX? . . . 12

3.3 Document styles . . . 12

3.4 Example . . . 14

1 What is METAFONT?

1.1 Introduction

METAFONT is a relatively low level font building system, suited to raster-based devices that print or display text. The characters that you are now reading were all designed with METAFONT.

A METAFONT user writes a program for each character or symbol of a typeface. These programs are different from ordinary computer programs, because they are essentially *declarative* rather than *imperative* (Knuth 1986b). Hopefully, METAFONT will incorporate *object orientation* in the near future. If any one tackles this task, it would be interesting and beneficial to all users.

In METAFONT language you specify where the major components of a desired shape are to be located, and how they relate to each other, but you do not have to work out the details of exactly where the lines cross; the computer takes over the work of solving equations as it deduces the consequences of your specifications (Knuth 1986b).

This report does not describe all the METAFONT features, because to do so would be redundant.Readers seeking basic information should consult The METAFONTbook (Knuth 1986b).

1.2 How to bring METAFONT up

In this subsection, I will explain how to bring METAFONT up, based on my experiments (try and error).

The default values for all of METAFONT environment variables were set at the time of compilation. Users can refer to the local user guide for details, if there is one. Once you change or add other fonts into a different directory, then you may have to update some of these variables, which are described bellow:

- MFINPUTS: Search path for input and opening files (*.mf).
- MFINPUTS_SUBDIR: Search path for directories with sub directories of input files.
- MFBASES: Search path for base files (*.base).
- MFPOOL: Search path for METAFONT internal strings.
- MFEDIT: Command template for switching to editor.
- MFTERM: Determines the on-line graphics display. (e.g., *sun* for SunView, *xterm* for either X10 or X11)

To invoke METAFONT, simply type *mf* in your computer. If METAFONT has been initialized., you will see ** in front of you, rather than a ** . When you only get one *, do not *input plain* as indicated in The METAFONTbook.

Thereafter, you might think you have succeeded. You should then type *showit;*. If you have succeeded, the graphical panel will show up "immediately". If not, please check whether you have set the environment variable $TERM$ to *xterm* or not (assume you are using X11). If not, simply type:

setenv TERM xterm

into your computer or add it into your *.cshrc* or *.login* file and *execute* it. The variable TERM is more important than the variable MFTERM, based on my experiments.

There is another tricky bit. If you use *rlogin* to open another session and try to bring up METAFONT in the new session, you may not succeed in setting the environment variable TERM correctly.

For example, let us assume that your host system is in *krista* [1], and you open an additional session to *nellads*[2]. As required, you have to set another environment variable $DISPLAY$. If your host machine is *nashi* [3], and you open another session to $nella07$ [4], normally, you will set DISPLAY to $nashi.fcit.monash.edu.au:0.0$ by typing:

setenv DISPLAY nashi.fcit.monash.edu.au:0.0

This setting, however, may disappoint you by failing to bring up the graphical window.

Because of the lower-level nature of METAFONT, you will need to set the variable DISPLAY to the physical host address. In this case, what you need is *nashi*'s actual address. Simply type *nslookup*(`^D` to quit) to get the actual physical address of *nashi*. Then set the variable DISPLAY as follows:

setenv DISPLAY 130.194.224.221:0.0

After you have completed this, the METAFONT graphical panel should appear on your screen. If not, please ask your local guru for help.

1.3 How to QUIT METAFONT

You might find it is hard to properly quit from METAFONT. In this case, guesswork is as good as any other means. My experience suggests the `\end` is for quitting the METAFONT, and `^D` is for *Emergency stop* (and quit as well). Notice that if you get a question mark "?" after testing some commands as The METAFONTbook suggests, you simply type ? for help. Lower case q and x are used for quitting in ? mode, which the METAFONT 'help' message does not indicate.

[1] A series of DEC stations at Monash University

[2] Another series of DEC stations at Monash University

[3] One of the krista series

[4] One of the nellads series

1.4 Some tips for METAFONT beginners

Most users when learning new systems always follow the manual faithfully, and test one example after another, but this is not always a reliable method. The most promising book on METAFONT is "The METAFONTbook" (Knuth 1986b). However, the author assumes that the *mf* currently in use is still the original version (*inimf*), which is no longer the reality.

For example, the *mf* in our system inputs *plain.mf* and *mobius.mf* as its pre-loading files. Therefore, some variable names may have been pre-set. Therefore, typing in the examples that The METAFONTbook suggests might not work as you expect. Fortunately, most of the examples work well. As you became more familiar with METAFONT, you can simply correct faulty examples.

In METAFONT, you are not supposed to update the *relation* of variables by adding another equation, if you have previously specified one. Once a variable is bound with a constant, then it is locked and cannot be changed. For example, if you first type x=100, and then add x=200 afterwards, you will receive an error message such as "! Inconsistent equation ...". This is because the equation sign = is used for declaring that the value on the left is equal to the one on the right. The assignment sign := is also available in METAFONT, but its use is not recommended by Knuth. Assignment (:=) is a basic operator of the *imperative* style whilst Knuth recommends that the *decalarative* style be used in METAFONT.

Another important thing is that you should not input the same file more than once. If you do so, you may not be able to consume all the question marks that METAFONT generates for you because of redundant equations. This is the reason why you cannot input *plain.mf* when there is only one * in front of you, as mentioned earlier.

The easiest way to learn to design your own fonts is to look at the examples that already exist in your system. For example, Computer Modern Fonts, and AMSFonts are very useful. You can make another copy of such *.*mf* files and then rename and update them. There are at least four types of files with the same extension filename .*mf*. Read the next section if you want to experiment with these.

If you want to look at some fascinating fonts that are created in Computer Modern System, type in *mf cmr*10 [5], sit back and relax, and pretend that you are a font connoisseur.

[5] Make sure the search paths have been set up properly.

2 Programming in METAFONT

2.1 Organization of Files

Normally the programs for certain font types (e.g., Computer Modern and AMSFonts) consist of many little pieces of programs that are grouped together in dozens [6] of files. There are four main categories of files:

- The *parameter file* defines parameters for specific typefaces. For example, *gis10.mf* [7] is the parameter file for GISFonts.
- The *base file* contains macros and declarations that are used by all programs. For example, I have adopted *cmbase.mf* [8] as the base file for GISFonts.
- The *driver file* controls the font-generation process and define ligature and kerning conventions. For example, *gis.mf* [9] is the driver file for GISFonts.
- The *program file* contains many short programs for individual characters or symbols. For example, *gissymbols.mf* is the program file for GISFonts.

For further information, the reader should refer to Computer Modern Typefaces (Knuth 1986a). The GISFonts sample programs are listed in Appendix A, B and C.

When you want to generate a new font for your system, simply invoke METAFONT with a command line like

```
mf \\mode=localfont\; mag=1.2\; input gis10\;
or
mf '\mode=localfont; mag=1.2; input gis10;'
```

localfont is the name of the mode that makes fonts for your standard output device. $mag = 1.2$ means that this run will produce a font that magnifies 1.2 times the resolution [10] of your device. *input gis10* means read *gis10.mf* as input file.

Your system will begin by reading the parameter file *gis10.mf*. Next, *gis10.mf* will call in the base file *cmbase.mf*, and follow by switching to the driver file *gis.mf*. The driver file will then input the program files *gissymbols.mf*, generating symbols one by one. Your terminal will be something like this at the end of the run:

```
This is METAFONT, C Version 2.0
(gis10.mf (/usr/local/lib/mf/inputs/cmbase.mf)
(gis.mf (gissymbols.mf [0] [1] [2] [3] [4] [5] [6] [7] [8]
```

[6] The small example font in this manual are GISFonts, which only consist of 3 files and generate 27 symbols.

[7] Modified from *msam10.mf* in AMSFonts

[8] is from Computer Modern

[9] Modified from *amsya.mf* in AMSFonts

[10] The system used here is 300dpi.

```
[9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]
[21] [22] [23] [24] [25] [26] [27]) ) )
```

Font metrics written on gis10.tfm.
Output written on gis10.360gf (28 characters, 12224 bytes).
Transcript written on gis10.log.

The system will generate both $gis10.tfm$ and $gis10.360gf$ files. Then you can use a utility program *gftopk* (*GFtoPK*) to convert $gis10.360gf$ file to $gis10.360pk$. You can also use *gftodvi* (*GFtoDVI*) to convert $gis10.360gf$ to $gis10.dvi$. Thereafter, use *xdvi* to look at the symbols, which are created by GISFonts.

2.2 Number and type of PK files needed

You might find there are many PK files with different numbers in the TEXPKS and XDVIFONTS directories. How many different scaled PK files have to be generated?

The system of magsteps is identical to the system used by TeX , with values generally in the range 0, 0.5, 1.0, 2.0, 3.0, 4.0 and 5.0. A listing of PK numbers for 118-dpi, 240-dpi and 300-dpi fonts is shown below:

MAGSTEP	118 *dpi*	240 *dpi*	300 *dpi*
$mag = magstep(0)$	118	240	300
$mag = magstep(0.5)$	129	263	329
$mag = magstep(1)$	142	288	360
$mag = magstep(2)$	170	346	432
$mag = magstep(3)$	204	415	518
$mag = magstep(4)$	245	498	622
$mag = magstep(5)$	294	597	746

Magnification can also be specified not as a *magstep* but as an arbitrary value, such as 1.315, to create special character sizes.

Does any reader find that these PK numbers do not increase as a geometrical series? For instance, when $magstep = 1$, the PK number is 360. It does not double in size to the PK number 300(magstep =0). The scale may seem strange, but if you look at the *magstep* macros in *plain.tex* [11] file, you will not be surprised. The macros are as follows:

```
\def\magstephalf{1095}
\def\magstep#1{\ifcase#1 1000\or
  1200\or 1440\or 1728\or 2074\or 2488\fi\relax}
```

This solves mystery, doesn't it?

[11] New version 3.0 called *lplain.tex*.

2.3 The program file

2.3.1 Brief introduction to METAFONT language

Although we are not going to reintroduce the METAFONT programming, some pre-knowledge is necessary for understanding the next sub subsection.

The location of a point is defined by specifying its x and y coordinates. In METAFONT (x_n, y_n) use to present the n-th point of a symbol. The abbreviation of (x_n, y_n) is z_n which saves large amount of typing time. For example, a set of commands like

$(x_1, y_1) = (0, 100)$;
$(x_2, y_2) = (100, 100)$;
draw $z_1 - -z_2$;
showit;

denotes that line is drawn from point z1 to z2, and will be shown it on the graphical window. Sometimes, if the symbols you have created is bit too tall, METAFONT provides some remedy keywords, e.g., *top*, *bot*, *lft* and *rt*, which let you adjust your symbol's look.

METAFONT programs often use another means to express relations between points. For example:

$z_4=\frac{1}{4}$ [z_1, z_2] represents the point z_4 one-fourth of the way from z_1 to z_2.
z_3=.5[z_1, z_2] represents the point z_3 half way from z_1 to z_2.

Some often used variables provide the following specific information

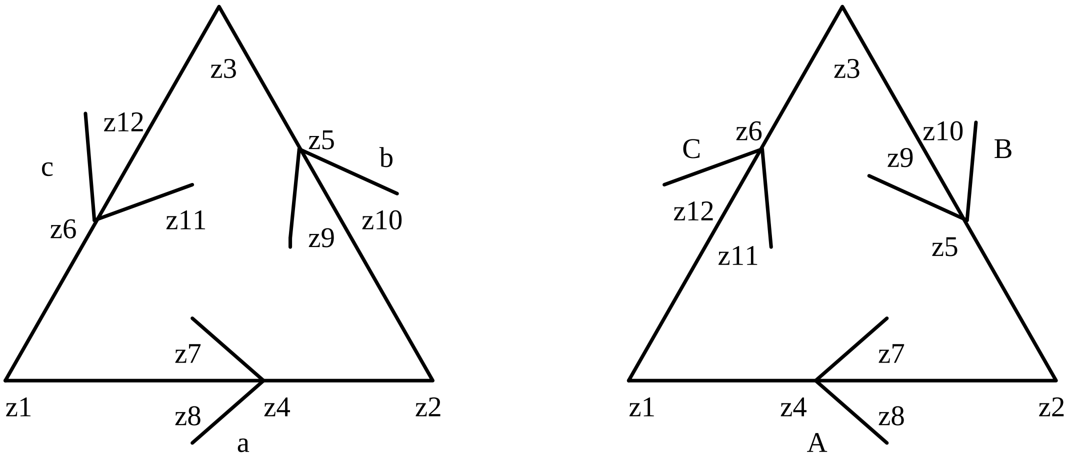
- h: The height of the box in which you specify your symbols.
- w: The width of the box in which you specify your symbols.
- o: The number of pixels which are allowed to draw over the box.

Having above information should help you to understand the following program.

2.3.2 The file: GISSYMBOLS.MF

In this subsection, we will discuss the real source code of the triangle symbols. Readers can refer to the the program lists in Appendix C.

```
def maketriangle (suffix i,j,k) =
bot y.i=0; y.j=y.i;
top y.k=h+o;
.5[x.i,x.j]=x.k=good.x .5w;  %compute x.k to the approximation of .5w
```



A. Triangle abc B. Triangle ABC

Figure 1: GIS Symbols

```
center_on(x.k);               %macro defined in cmbase.mf
lft x.i=hround(.5w-u*sqrt48);%refer to p264 of The METAFONTbook
%% user must draw the triangle him/herself.
enddef;
```

The above code is a macro for defining an ordinary triangle. The parameters, i, j, and k determine which point [12] in the triangle. (X_n, Y_n) denotes the n-th point (using 2D coordinates system) in the triangle. The functions, e.g., *hround* and *center_on*, can be found in The METAFONTbook (Knuth 1986b) and Computer Modern Typefaces (Knuth 1986a).

```
def makearrow(suffix a,b,i,j,k,c)=
z13 = .15(w,0) ;          %The vector of one side of an arrow
if (i=4): theta:=0;       %The degree of bottom side
elseif (i=5): theta:=120; %The degree of right-hand side
else: theta:=240;         %The degree of left-hand side
fi;
if (c=1):
    z.i=.45[z.a,z.b];     %The position of the arrow
    z.j =z13 shifted z.i rotatedaround(z.i,theta+40);
    %The arrow has been translated and rotated
    z.k =z.j reflectedabout(z.a,z.b);
    %The other side of the arrow just mirroring
```

[12]Starting from left-bottom point and in counter-clockwise order.

```
    %The previous one
else:
    z.i=.55[z.a,z.b];
    z.j =z13 shifted z.i rotatedaround(z.i,theta+220);
    z.k =z.j reflectedabout(z.a,z.b);
fi;
enddef;
```

The above code is a macro for an arrow. The parameters, a and b, are two points that determine a certain side of the triangle. The parameter i denotes which side in the triangle, and $z.i$ becomes the peak point of the arrow. The $z.j$ and $z.k$ are part of the arrow, as portrayed in the above figure. The parameter c determines whether the direction is clockwise or counter-clockwise.

```
cmchar "Triangleabc";
beginchar(oct"032",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;         % Drawing a triangle
makearrow(1,2,4,7,8,0);
draw z7--z4--z8;                % The arrow a
makearrow(2,3,5,9,10,0);
draw z9--z5--z10;               % The arrow b
makearrow(3,1,6,11,12,0);
draw z11--z6--z12;              % The arrow c
labels(1,2,3,4,5,6,7,8,9,10,11,12);
endchar;
```

```
cmchar "TriangleABC";
beginchar(oct"023",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;         % Draw a triangle
makearrow(1,2,4,7,8,1);
```

```
draw z7--z4--z8;                 % The arrow A
makearrow(2,3,5,9,10,1);
draw z9--z5--z10;                % The arrow B
makearrow(3,1,6,11,12,1);
draw z11--z6--z12;               % The arrow C
labels(1,2,3,4,5,6,7,8,9,10,11,12);
endchar;
```

The above two parts of programs represent two triangles, *abc* and *ABC*, as can be seen in figure 1. Readers with some programming experience should be able to understand the code. In order to avoid insulting experienced users, tedious explanations will not be given here. For less experienced users, it is time for you to study <u>The METAFONTbook</u> now.

3 How to Glue all the pieces together

3.1 What is TEX?

TEX is a typesetting system intended for the creation of beautiful books, and especially for books that contain large amount of mathematics. By preparing a manuscript in TEX format, you will be telling a computer exactly how the manuscript is to be transformed into pages whose typographic quality is comparable to that of the world's finest printers (Knuth 1986c).

The inconvenience is that what you write is not what you will see right away. However TEX is so powerful, you can do almost anything you want in typesetting much like C language (in the category of programming language).

The best source of knowledge about TEX is The TEXbook (Knuth 1986c).

3.2 What is LATEX?

The LATEX document preparation system, like AMS-TEX, is a special version of Knuth's TEX program. As mentioned earlier, TEX is a sophisticated program designed to produce high-quality typesetting, especially for mathematical texts. LATEX adds to TEX a collection of commands that makes typesetting simple by letting the user concentrate on the structure of the text, rather than on formatting commands.

The LATEX version 3.0 takes *lplain.tex* as its preloading file, *lplain.tex* then calls *hyphen.tex*, *lfonts.tex* and *latex.tex* to construct the LATEX configuration. All the macros and tex-font setting can be easily found in these files . If the reader would like to know more about LATEX, who can trace these files.

3.3 Document styles

Some standard LATEX styles for ordinary documents are the *article*, *report* and *book* styles. In addition to choosing the main style, you can also select from among certain document-styles, for example, 11*pt*, 12*pt* and *twocolumn*. Your system may have some other special styles, e.g., *zed* and *oz*, please check your local guide (if you have one). Do not forget to add those directories, which contain special styles, into your *TEXINPUTS* search paths.

The intention here to help you create your own style and incorporate the symbols that are already available. Certainly, the easiest way to do this is to use the style that exist in your system.

terrain.sty, for example, is inspired by *oz.sty* [13] and which is the successor of *zed.sty* [14].

Readers can refer to *terrain.sty* in Appendix B, apart from the following legend.

The *terrain.sty* was named by Mr. Bohdan Durnota. According to his research into

[13] Written by Paul King, Queensland University

[14] Created by Mike Spivey, Oxford University

geographical information system, 27 triangles need to be created as symbols so that these symbols can be used in any document, which incorporates *terrain.sty*.

The style files consists of two sections i.e., FONTS and SYMBOLS. We start from the FONTS section.

```
\font\sevgis=gis10   \@magscale0  %300pk
\font\egtgis=gis10   \@halfmag    %328pk
⋮
\font\fvtngis=gis10  \@magscale5  %746pk
```

The above command is to load the fonts *gis10.mf* with different scales into TEXś memory, the system will then select these fonts for typesetting.

As you can see from the right-hand side comments for each line, these are the PK numbers that you have to create for this font. The method for this was explained in the previous section.

```
\newfam\gisfam
```

The new font is then registered into TEX. TEX community welcomes this font as a new family and assigns a unique counter register to *gisfam*.

```
\def\@ixpt{\textfont\gisfam=\ningis \scriptfont\gisfam=\sevgis}
⋮
\def\@xivpt{\textfont\gisfam=\fvtngis \scriptfont\gisfam=\tengis}
```

The commands `\texfont` and `\scriptfont` are used to specify the members of each family.

```
\def\@famletter#1{\ifcase #1 0\or 1\or 2\or 3\or 4\or 5\or
 6\or 7\or 8\or 9\or A\or B\or C\or D\or E\or F\fi}
\edef\@fx{\@famletter\gisfam}
```

The register number which *gisfam* receives is a decimal number. To specify a symbol in a textfont table, however, you need to use a hexadecimal number. Therefore, these commands simply do the converting from decimal to hexadecimal numbers. Also shorten the textfont family number to `\@famletter\gisfam` to `\@fx`.

The second section is SYMBOLS. In this section symbols are definded and correspond with their individual locations in the font table (refer to Appendix E).

```
\let\@mc=\mathchardef      %alias
\@mc \Triangle             "0\@fx00
:
\@mc \Triangleabc          "0\@fx1A
```

The triangle symbols are classified into math mode, and TEX uses a 15-bit number to specify each symbol. The biggest number is 7FFF. The first hexadecimal digit stands for one of eight classes which are as follows:

0 : *Ordinary*
1 : *Large operator*
2 : *Binary operator*
3 : *Relation*
4 : *Opening*
5 : *Closing*
6 : *Punctuation*
7 : *Variable family*

The second hexadecimal digit stands for the particular family the symbol belongs to. The other two bytes stand for the location of the family where the symbol resides.

For further details, readers can refer to chapters 17 and 26 of The TEXbook (Knuth 1986c).

3.4 Example

How do you use the symbols that have been created? The following is a TEX file example:

```
This is a testing file:\\
Triangle:     \Triangle\\
TriangleA:    \TriangleA\\
Trianglea:    \Trianglea\\
```

```
TriangleB:    \TriangleB\\
\vdots\\
TriangleAB:   \TriangleAB\\
TriangleAb:   \TriangleAb\\
TriangleaB:   \TriangleaB\\
\vdots\\
TriangleaBc:  \TriangleaBc\\
TriangleabC:  \TriangleabC\\
Triangleabc:  \Triangleabc\\
```

---

which is translated as follows:

This is a testing file:

$Triangle:$ 

$TriangleA:$ 

$Trianglea:$ 

$TriangleB:$ 

$\vdots$

$TriangleAB:$ 

$TriangleAb:$ 

$TriangleaB:$ 

$\vdots$

$TriangleaBc:$ 

$TriangleabC:$ 

$Triangleabc:$ 

As mentioned earlier, this work is based on my experiments. The information may not be sufficient for some particular systems. Readers are most welcome to give me comments and suggestions.

THE END

That's all, folks.

Happy METAFONTing!

References

American Mathematical Society (1991, August). *User's Guide to AMSFonts Version 2.1.* Providence: American Mathematical Society.

Knuth, D. E. (1986a). *Computer Modern Typefaces*, Volume E of Computers & Typesetting. Reading: Addison-Wesley Publishing Co.

Knuth, D. E. (1986b). *The* METAFONT *book*, Volume C of Computers & Typesetting. Reading: Addison-Wesley Publishing Co.

Knuth, D. E. (1986c). *The TEXbook*, Volume A of Computers & Typesetting. Reading: Addison-Wesley Publishing Co.

Lamport, L. (1986). LATEX*: A Document Preparation System.* Reading: Addison-Wesley Publishing Co.

Tobin, G. (1993, August). *METAFONT for Beginners.* Latrobe University, Australia.

Appendix A.

GIS10.MF

```
% GIS10.MF                      Nov 1993
% This Metafont file is part of GISFonts 1.0.
% Copyright (C) 1993, SFT Dept. Monash University Australia.
% All rights reserved.  Copying of this file is authorized only if either:
%   (1) you make absolutely no changes to your copy including filename; OR
%   (2) if you do make changes, you first rename it to some other filename.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is GIS10.MF in text format.
% 10pt
% development starting Nov 3, 1993

if unknown cmbase: input cmbase fi

font_identifier:="GIS"; font_size 10pt#;

u#:=20/36pt#;        % unit width
width_adj#:=0pt#;    % width adjustment for certain characters
serif_fit#:=0pt#;    % extra sidebar near lowercase serifs
cap_serif_fit#:=5/36pt#;  % extra sidebar near uppercase serifs
letter_fit#:=0pt#;   % extra space added to all sidebars

body_height#:=270/36pt#;  % height of tallest characters
asc_height#:=250/36pt#;   % height of lowercase ascenders
cap_height#:=246/36pt#;   % height of caps
fig_height#:=232/36pt#;   % height of numerals
x_height#:=155/36pt#;     % height of lowercase without ascenders
math_axis#:=90/36pt#;     % axis of symmetry for math symbols
bar_height#:=87/36pt#;    % height of crossbar in lowercase e
comma_depth#:=70/36pt#;   % depth of comma below baseline
desc_depth#:=70/36pt#;    % depth of lowercase descenders

crisp#:=0;           % diameter of serif corners
tiny#:=8/36pt#;      % diameter of rounded corners
fine#:=7/36pt#;      % diameter of sharply rounded corners
thin_join#:=7/36pt#; % width of extrafine details
hair#:=9/36pt#;      % lowercase hairline breadth
stem#:=24/36pt#;     % lowercase stem breadth
curve#:=29/36pt#;    % lowercase curve breadth
ess#:=27/36pt#;      % breadth in middle of lowercase s
flare#:=32/36pt#;    % diameter of bulbs or breadth of terminals
dot_size#:=38/36pt#; % diameter of dots
cap_hair#:=11/36pt#; % uppercase hairline breadth
```

```
cap_stem#:=30/36pt#; % uppercase stem breadth
cap_curve#:=35/36pt#;% uppercase curve breadth
cap_ess#:=33/36pt#;  % breadth in middle of uppercase s
rule_thickness#:=.4pt#;   % thickness of lines in math symbols

dish#:=1/36pt#;      % amount erased at top or bottom of serifs
bracket#:=20/36pt#;  % vertical distance from serif base to tangent
jut#:=30/36pt#;      % protrusion of lowercase serifs
cap_jut#:=37/36pt#;  % protrusion of uppercase serifs
beak_jut#:=10/36pt#; % horizontal protrusion of beak serifs
beak#:=70/36pt#;     % vertical protrusion of beak serifs
vair#:=8/36pt#;      % vertical diameter of hairlines
notch_cut#:=10pt#;   % maximum breadth above or below notches
bar#:=11/36pt#;      % lowercase bar thickness
slab#:=11/36pt#;     % serif and arm thickness
cap_bar#:=11/36pt#;  % uppercase bar thickness
cap_band#:=11/36pt#; % uppercase thickness above/below lobes
cap_notch_cut#:=10pt#;    % max breadth above/below uppercase notches
serif_drop#:=4/36pt#;     % vertical drop of sloped serifs
stem_corr#:=1/36pt#;      % for small refinements of stem breadth
vair_corr#:=1/36pt#;      % for small refinements of hairline height
apex_corr#:=0pt#;    % extra width at diagonal junctions

o#:=8/36pt#;         % amount of overshoot for curves
apex_o#:=8/36pt#;    % amount of overshoot for diagonal junctions

slant:=0;            % tilt ratio $(\Delta x/\Delta y)$
fudge:=1;            % factor applied to weights of heavy characters
math_spread:=0;      % extra openness of math symbols
superness:=1/sqrt2;  % parameter for superellipses
superpull:=1/6;      % extra openness inside bowls
beak_darkness:=11/30;% fraction of triangle inside beak serifs
ligs:=2;             % level of ligatures to be included

square_dots:=false;  % should dots be square?
hefty:=false;        % should we try hard not to be overweight?
serifs:=true;        % should serifs and bulbs be attached?
monospace:=false;    % should all characters have the same width?
variant_g:=true;     % should an italic-style g be used?
low_asterisk:=false; % should the asterisk be centered at the axis?
math_fitting:=true;  % should math-mode spacing be used?

% the following font parameters are explained in {\sl The \TeX book}, Appendix G
subs:=7/10;          % subscripts to 10pt will normally be in 7pt type
rth#:=.4pt#;         % assume that rules will come from cmex10
fontdimen 8:         % parameters $\sigma_8$ through $\sigma_{22}$ will now be listed
 math_axis#+3.51rth#+54/36pt#+subs*desc_depth#,      % num1
```

```
 math_axis#+1.51rth#+30/36pt#,                          % num2
 math_axis#+1.51rth#+48/36pt#,                          % num3
 -(math_axis#-3.51rth#-subs*fig_height#-124/36pt#),  % denom1
 -(math_axis#-1.51rth#-subs*fig_height#-30/36pt#),   % denom2
 8.99pt#-subs*asc_height#,             % sup1
 8.49pt#-subs*asc_height#,             % sup2
 104/36pt#,                            % sup3
 54/36pt#,                             % sub1
 -(8.49pt#-2subs*asc_height#-3.1rth#), % sub2
 subs*asc_height#-36/36pt#,            % sup\_drop
 18/36pt#,                             % sub\_drop
 23.9pt#,                              % delim1
 10.1pt#,                              % delim2
 math_axis#;                           % axis\_height

%% include these parameters to shrink R and S so they
%% fit inside a circle
%
lower.u#:=16/36pt#;            % unit width
lower.width_adj#:=0pt#;        % width adjustment for certain characters
lower.cap_serif_fit#:=4/36pt#; % extra sidebar near uppercase serifs
lower.letter_fit#:=2.4/36pt#;  % extra space added to all sidebars

lower.body_height#:=200/36pt#; % height of tallest characters
lower.cap_height#:=185/36pt#;  % height of caps
lower.x_height#:=116/36pt#;    % height of lowercase without ascenders
lower.bar_height#:=65/36pt#;   % height of crossbar in lowercase e
lower.comma_depth#:=52/36pt#;  % depth of comma below baseline

lower.flare#:=25/36pt#;        % diameter of bulbs or breadth of terminals
lower.cap_hair#:=9.1/36pt#;    % uppercase hairline breadth
lower.stem#:=21/36pt#;         % lowercase stem breadth
lower.cap_stem#:=26/36pt#;     % uppercase stem breadth
lower.cap_curve#:=31/36pt#;    % uppercase curve breadth
lower.cap_ess#:=28/36pt#;      % breadth in middle of uppercase s

lower.cap_jut#:=26/36pt#;  % protrusion of uppercase serifs
lower.beak_jut#:=7/36pt#;  % horizontal protrusion of beak serifs
lower.beak#:=52/36pt#;     % vertical protrusion of beak serifs
lower.slab#:=9.1/36pt#;    % serif and arm thickness
lower.cap_bar#:=9.1/36pt#;     % uppercase bar thickness
lower.cap_band#:=9.1/36pt#;    % uppercase thickness above/below lobes
lower.cap_notch_cut#:=10pt#;   % max breadth above/below uppercase notches

lower.o#:=5/36pt#;         % amount of overshoot for curves
lower.apex_o#:=4/36pt#;    % amount of overshoot for diagonal junctions
```

```
lower.fudge:=1;              % factor applied to weights of heavy characters
generate gis                 % switch to the driver file gis.mf
```

Appendix B.

GIS.MF

```
%% @metafontfile{
%%     filename="gis.mf",
%%     version="1.0",
%%     date="3-Nov-1993",
%%     filetype="Metafont: driver",
%%     copyright="Copyright (C) SFT Dept. Monash University Australia,
%%            all rights reserved.  Copying of this file is
%%            authorized only if either:
%%            (1) you make absolutely no changes to your copy
%%                including name; OR
%%            (2) if you do make changes, you first rename it to some
%%                other name.",
%%     author="Richard Lin",
%%     address="Software Development Department,
%%            Monash University,
%%            P.O. Box 197,
%%            Caulfield East, VIC 3145,
%%            AUSTRALIA",
%%     telephone="613-573-2975 ",
%%     fax="613-573-2745",
%%     email="rlin@fcitds.fcit.monash.edu.au",
%%     codetable="",
%%     checksumtype="line count",
%%     checksum="",
%%     keywords="",
%%     abstract="This is the driver file for use with GIS"
%%     }
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

font_coding_scheme:="Geographic Information System";
mode_setup; font_setup;

autorounding:=0;

font_slant slant; font_x_height x_height#;
font_quad 18u# if not monospace:+4letter_fit# fi;
slant:=mono_charic#:=0;  % the remaining characters will not be slanted
currenttransform:=identity yscaled aspect_ratio scaled granularity;
input gissymbols;        % the symbols
bye.
```

Appendix C.

GISSYMBOLS.MF

```
%% @metafontfile{
%%     filename="gissymbols.mf",
%%     version="1.0",
%%     date="3-Nov-1993",
%%     filetype="Metafont: program",
%%     copyright="Copyright (C) SFT Dept. Monash University Australia,
%%            all rights reserved.  Copying of this file is
%%            authorized only if either:
%%            (1) you make absolutely no changes to your copy
%%                including name; OR
%%            (2) if you do make changes, you first rename it to some
%%                other name.",
%%     author="Richard Lin",
%%     address="Software Development Department,
%%            Monash University,
%%            P.O. Box 197,
%%            Caulfield East, VIC 3145,
%%            AUSTRALIA",
%%     telephone="613-573-2975 ",
%%    fax="613-573-2745",
%%     email="rlin@fcitds.fcit.monash.edu.au",
%%     codetable="",
%%     checksumtype="line count",
%%     checksum="",
%%     keywords="",
%%     abstract="This is the progrsm file for use with GIS"
%%     }
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

def maketriangle (suffix i,j,k) =
bot y.i=0; y.j=y.i; top y.k=h+o;
.5[x.i,x.j]=x.k=good.x .5w; center_on(x.k); lft x.i=hround(.5w-u*sqrt48);
%% user must draw the triangle him/herself.
enddef;

def makearrow(suffix a,b,i,j,k,c)=
z13 = .15(w,0) ;
if (i=4): theta:=0;
elseif (i=5): theta:=120;
else: theta:=240;
fi;
```

```
if (c=1):
    z.i=.45[z.a,z.b];
    z.j =z13 shifted z.i rotatedaround(z.i,theta+40);
    z.k =z.j reflectedabout(z.a,z.b);
else:
    z.i=.55[z.a,z.b];
    z.j =z13 shifted z.i rotatedaround(z.i,theta+220);
    z.k =z.j reflectedabout(z.a,z.b);
fi;
enddef;

cmchar "Large triangle";
beginchar(oct"000",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % drawing triangle
labels(1,2,3); endchar;

cmchar "TriangleA";
beginchar(oct"001",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,1);
draw z7--z4--z8;         % arrow A
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;

cmchar "Trianglea";
beginchar(oct"002",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,0);
draw z7--z4--z8;         % arrow a
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;

cmchar "TriangleB";
beginchar(oct"003",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(2,3,5,9,10,1);
draw z9--z5--z10;        % arrow B
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;
⋮

cmchar "TriangleAB";
```

```
beginchar(oct"007",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,1);
draw z7--z4--z8;         % arrow A
makearrow(2,3,5,9,10,1);
draw z9--z5--z10;        % arrow B
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;

cmchar "TriangleAb";
beginchar(oct"010",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,1);
draw z7--z4--z8;         % arrow A
makearrow(2,3,5,9,10,0);
draw z9--z5--z10;        % arrow b
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;

⋮

cmchar "TriangleABC";
beginchar(oct"023",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,1);
draw z7--z4--z8;         % arrow A
makearrow(2,3,5,9,10,1);
draw z9--z5--z10;        % arrow B
makearrow(3,1,6,11,12,1);
draw z11--z6--z12;       % arrow C
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;

cmchar "TriangleABc";
beginchar(oct"024",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,1);
draw z7--z4--z8;         % arrow A
makearrow(2,3,5,9,10,1);
draw z9--z5--z10;        % arrow B
makearrow(3,1,6,11,12,0);
draw z11--z6--z12;       % arrow c
```

```
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;
```

⋮

```
cmchar "Triangleabc";
beginchar(oct"032",16u#,asc_height#,desc_depth#);
adjust_fit(0,0); pickup rule.nib;
maketriangle(1,2,3);
draw z1--z2--z3--cycle;  % triangle
makearrow(1,2,4,7,8,0);
draw z7--z4--z8;         % arrow a
makearrow(2,3,5,9,10,0);
draw z9--z5--z10;        % arrow b
makearrow(3,1,6,11,12,0);
draw z11--z6--z12;       % arrow c
labels(1,2,3,4,5,6,7,8,9,10,11,12); endchar;
```

Appendix D.

TERRAIN.STY

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% terrain.sty
%
\message{`TERRAIN Macros' <9 Nov 90>}
%
% This document is organised as follows:
%
% SECTION 1 FONTS
% SECTION 2 SYMBOLS
%
% Modification History:
%
% Nov, 93 27 triangles is created - Richrad Lin
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Please post any updates that you may make to this file to:
% Richard Lin -- rlin@fcitds.fcit.monash.edu.au
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% SECTION 1 FONTS
%
% The GIS symbol fonts are loaded.
%

\font\sevgis=gis10  \@magscale0 %300pk
\font\egtgis=gis10  \@halfmag   %328pk
\font\ningis=gis10  \@magscale1 %360pk
\font\tengis=gis10  \@magscale2 %432pk
\font\elvgis=gis10  \@magscale3 %518pk
\font\twlgis=gis10  \@magscale4 %622pk
\font\fvtngis=gis10 \@magscale5 %746pk

\newfam\gisfam

\def\@ixpt{\textfont\gisfam=\ningis \scriptfont\gisfam=\sevgis}
\def\@xpt{\textfont\gisfam=\tengis \scriptfont\gisfam=\sevgis}
\def\@xipt{\textfont\gisfam=\elvgis \scriptfont\gisfam=\egtgis}
\def\@xiipt{\textfont\gisfam=\twlgis \scriptfont\gisfam=\ningis}
\def\@xivpt{\textfont\gisfam=\fvtngis \scriptfont\gisfam=\tengis}

\def\@famletter#1{\ifcase #1 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or
```

```
8\or 9\or A\or B\or C\or D\or E\or F\fi}

\edef\@fx{\@famletter\gisfam}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% SECTION 2 SYMBOLS
%

\let\@mc=\mathchardef

\@mc \Triangle    "0\@fx00
\@mc \TriangleA   "0\@fx01
\@mc \Trianglea   "0\@fx02
\@mc \TriangleB   "0\@fx03
\@mc \Triangleb   "0\@fx04
\@mc \TriangleC   "0\@fx05
\@mc \Trianglec   "0\@fx06
\@mc \TriangleAB  "0\@fx07
\@mc \TriangleAb  "0\@fx08
\@mc \TriangleaB  "0\@fx09
\@mc \Triangleab  "0\@fx0A
\@mc \TriangleAC  "0\@fx0B
\@mc \TriangleAc  "0\@fx0C
\@mc \TriangleaC  "0\@fx0D
\@mc \Triangleac  "0\@fx0E
\@mc \TriangleBC  "0\@fx0F
\@mc \TriangleBc  "0\@fx10
\@mc \TrianglebC  "0\@fx11
\@mc \Trianglebc  "0\@fx12
\@mc \TriangleABC "0\@fx13
\@mc \TriangleABc "0\@fx14
\@mc \TriangleAbC "0\@fx15
\@mc \TriangleAbc "0\@fx16
\@mc \TriangleaBC "0\@fx17
\@mc \TriangleaBc "0\@fx18
\@mc \TriangleabC "0\@fx19
\@mc \Triangleabc "0\@fx1A
```

Appendix E.

Font Specimens

Test of gis10 scaled 2000 on November 17, 1993 at 1640