The Scrabble font

By Jacques André

This issue's **P.S** feature describes the SCRABBLE font which displays alphabetic characters as stylised Scrabble tiles, arranged either horizontally or vertically according to whether the character in the string to be displayed is upper or lower case.

The SCRABBLE font has been designed to illustrate some interesting properties of POSTSCRIPT fonts:

- Characters may have a horizontal width or a vertical width
- Fonts using `setcharwidth` rather than `setcachedevice` within the `BuildChar` procedure may be dynamic *(see the* **cover art** *feature in Volume 2, number 3 of the Journal).*
- One font may display characters from a different font.

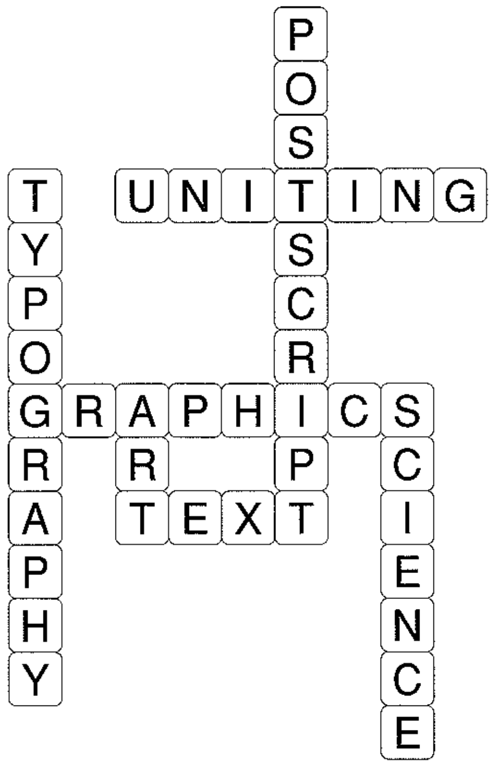
Although SCRABBLE uses the classic program for creating an analytic font (given in the Adobe 'Blue Book', program 20), some differences should be noted:

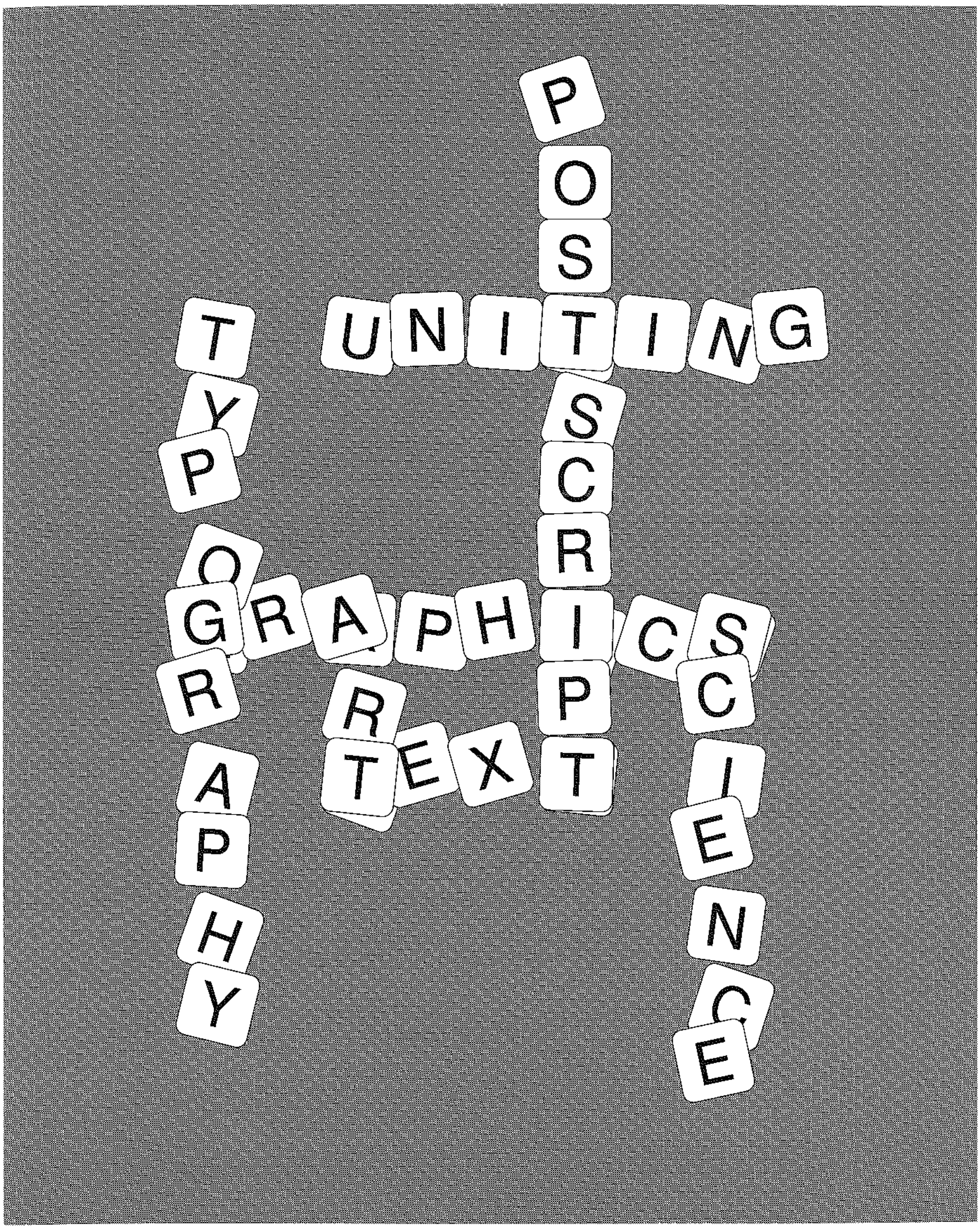
- There is no `/Metrics` or `/BBox` entry in the font dictionary.
- Encoding names are computed from the character code rather than being typed in to the text of the program.
- There is no `CharacterDefs` dictionary. Instead, `BuildChar` calls a unique procedure (`Piece`) which takes the character code as a parameter.
- Because the `definefont` operator makes the font dictionary's access read-only, the font uses an annexed dictionary (`DYFADIC`) which does have write access and so may be used to hold the values used in computations.

The source code for the SCRABBLE font is given below:

```
%! SCRABBLE: a sample font which exhibits
%  some properties of PostScript fonts
%  by Jacques Andre - IRISA/INRIA-Rennes, France.
%  The font uses the definition given in
%  the 'Blue Book', but it has a dynamic
%  aspect (using setcharwidth instead of
%  setcachedevice.)
%  Note the use of a DYFADIC (dynamic font annex
%  dictionary), the way the Encoding dict is built
%  and the way the BuildChar proc is defined.

%  This font is regular if /regular is
%  set to true, and random if not.

/DYFADIC 100 dict def
DYFADIC begin
/regular true def

% seed random number
25038 srand

/nbal {
      /alea rand def
      alea alea 20 idiv 20 mul sub 10 sub 2 mul
} def

% allow for a-z + A-Z + 2 spaces + notdef
/dictsize 55 def
/car (X) def

/Piece { % draw the Scrabble piece
      userdict begin
      /code exch def
      /letter car dup 0 code put def

      % use caps for vertical
      way not
      { % if
            letter (_) eq
            { % ifelse
            userdict begin /letter ( ) def end
            }
            { %else
            letter 0 code (A) 0 get (a)
            0 get sub add put
            } ifelse
      } if

      % is it random?
      regular not
      { nbal rotate } if % small variation on posn

      % draw Scrabble piece
      2 2 scale
      0 150 moveto
      0 250 200 250 36 arcto 4 { pop } repeat
      250 250 250 50 36 arcto 4 { pop } repeat
      250 0 50 0 36 arcto 4 { pop } repeat
      0 0 0 200 36 arcto 4 { pop } repeat
      closepath
      gsave 1 setgray fill grestore
      3 setlinewidth
      stroke

      % show letter
      /Helvetica findfont 200 scalefont setfont
      125 letter stringwidth pop 2 div sub 50 moveto
      letter show
      end
} def % Piece

end % DYFADIC

% begin font dictionary definition
/Myfont 9 dict def
Myfont begin
/FontType    3 def
/FontMatrix  [.001 0 0 .001 0 0] def
/FontBBox    [0 -500 500 500] def

% create Encoding array
/Encoding    256 array def
0 1 255 { Encoding exch /.notdef put } for

% insert character names
Encoding dup ( ) 0 get /Hspace put
(_) 0 get /Vspace put

% see below for the other values
/Horizontal
DYFADIC begin
      dictsize
end
dict def  %True if horizontal way

Horizontal begin
      /.notdef  false def
      /Hspace true def
      /Vspace false def
end

% compute character names for A-Z
(ABCDEFGHIJKLMNOPQRSTUVWXYZ) {
      /code exch def % from A -> Z
      DYFADIC begin car 0 code put end % (X)
      Encoding dup DYFADIC begin
            car 0 get car cvn put
      end % encoding /X
      Horizontal begin
            DYFADIC begin car cvn true end def
      end         % Horizontal
 } forall % loop CAP

% compute character names for a-z
 (abcdefghijklmnopqrstuvwxyz) {
      /code exch def % from a -> z
      DYFADIC begin car 0 code put end % (x)
      Encoding dup DYFADIC begin
            car 0 get car cvn put
      end % encoding /x
      Horizontal
      begin
            DYFADIC begin car cvn false end def
      end          % Vertical
 } forall % loop LOWERCASE

/BuildChar % cf the blue book
{
      0 begin
      /ccar exch def          % char code
      /DictPolice exch def
      /NomCar DictPolice /Encoding get ccar get def
      DictPolice begin
            Horizontal NomCar get
            DYFADIC begin /way exch def end % direction
            DYFADIC begin way end
            % charwidth is always the same size
            { 500 0 }{ 0 -500 } ifelse setcharwidth

            % call draw algorithm
            ccar DYFADIC begin Piece end
      end
end
} def %BuildChar

/BuildChar load 0 3 dict put

/UniqueID 123 def
end   % Myfont

/Scrabble Myfont definefont pop

% --- END OF DEFINITION OF SCRABBLE-FONT

% --- Now let us try it:
20 20 scale
/Scrabble findfont 2 scalefont setfont
3 11 moveto (UNITING) show
1 7 moveto (GRAPHICS) show
3 5 moveto (TEXT) show
1 11 moveto (typography) show
3 7 moveto (art) show
6 14 moveto (postscript) show
8 7 moveto (science) show
showpage
```

The sample shown below is produced with the value `/regular` set to `true`; the one shown on the inside back cover is done with `/regular` set to `false`.




POSTSCRIPT
UNITING
TYPOGRAPHY
GRAPHICS
SCIENCE
ART
TEXT