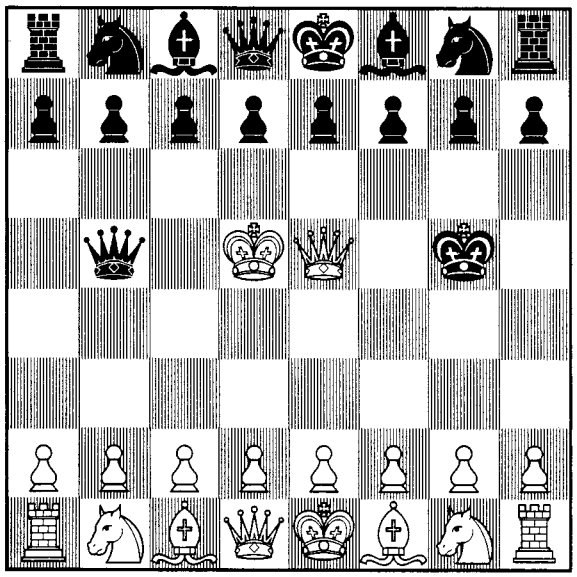
An Improved Chess Font

David Tofsted

In the July 1989 issue of *TUGboat* an article by Zalman Rubinstein discussed how to display chess board positions using a font created with METAFONT. Upon reading this article I was interested enough in the idea of producing a high quality chess font that I felt I had to try my hand at improving on Professor Rubinstein's "first cut." The model I used for the set was derived from a combination of the chess characters used by the *New York Times* and those of a vintage chess tutorial I have at home. The results of this work are presented here along with some comments concerning METAFONT peculiarities that were overcome in developing the characters. I do hope there is enough interest in the TEX community to use this new font. Perhaps it will also inspire others as to the highly versatile nature of the METAFONT program.

To begin, consider the net results. Figure 1 shows the 26 possible characters in this font. These include the six different pieces of each side on two differently colored squares each, plus an empty black and an empty white square. The commands used to present the chess board can be obtained from either Prof. Rubinstein's article or the article by Mr. Wolfgang Appelt (December 1988, *TUGboat*).

BLACK



WHITE

Figure 1.

In describing how these characters are formed, it must first be noted that the original printing format was a LaserJet, Series II. At 300 dpi the characters are amazingly sharp, but the hatched background (lines 28-33)* had to be handled carefully. It was designed such that every fifth column of dots was turned on. This limited the METAFONTness of the resulting font because changes were required at each magnification. Having a column of filled-in dots every five columns required adjusting the looping factor on the `hatch` subroutine. Without this adjustment the METAFONT program performs roundoff that causes the line spacing to be uneven. Also, the LaserJet tends to overlap the space between dots. This overfilling meant that the erase draws performed in the interior of the black characters had to be two dots wide rather than one, to compensate for the overfilling.

Aside from this minor detail, the characters are drawn similarly for both black and white pieces. The same set of control points can be used for each (the pair function `w[]`). Thus the array assignments only need to be specified once (lines 46-59) for each piece type. These points are then converted into `z` points (lines 64, 99, 119, and 143) within the character description. Additionally, the total number of characters is 26, but the number of chess pieces that must actually be drawn is only 12. This is possible because of Donald Knuth's `keepit` "dirty trick" (*The METAFONTbook*, p. 295) (lines 4-14, 88, and 98 for example). Using Knuth's technique, the chess piece is first drawn on a white background and saved (88, 98), then the next character (same piece on a black background) can be drawn simply by calling the `hatch` subroutine into a fresh picture drawing area (105), erasing out the area of the hatched character that will be filled in by the previously drawn piece (106-112), and then using the `addto` command (113) to combine the original chess piece with the new background.

Space would not allow a full listing and description of all the characters, so I will discuss in detail the steps required to generate only the Bishop, leaving the rest as an exercise for the reader. Before proceeding though, it would seem appropriate to comment on some of the key aspects of each of the other pieces as well. The two simplest characters were the Rook and the Queen. Both are symmetric about the vertical center line. Both were composed almost entirely of straight line segments. The Rook was particularly simple in that it used several points along the vertical center line and then achieved the brick corners via `penpos` commands.

*This notation refers to line numbers in the listing at the end of this article. Other references to the listing will be designated similarly.

A third piece, the King, was only difficult because of the number of points involved. Over 50 were needed. One problem area that did arise was in producing the curved lines describing the King's upper crown. Control points had been chosen in the arc around the two upper lobes of the crown, but when drawing the crown the tension command was needed on either side of the center to keep the lines from overlapping as in the example below. (A tension of 1.2 was found sufficient to correct the problem.)

Of the remaining three pieces, producing the Knight was rather simple because of its asymmetric shape. The asymmetry meant there was a wider range of shapes that "looked" right. I must however credit my wife Laura for the original drawing of the horse used in the Knight. The Pawn I finally adopted is possibly too simple, but this was a design decision since it focuses more attention on the stronger pieces.

This leaves the Bishop. Oddly enough more time was spent getting this character right than any of the others. At first a doubled miter was tried, but this construct resulted in a lopsided look. Also I had originally defined twelve points to use as turning points in drawing the cross in the miter. But this approach led to problems on the 300 dpi machine. I cannot reproduce this problem for this article because of the higher printer resolution, but suffice it to say that the vertical lines in the upper extension of the cross did not match the line in the lower cross portion. The vertical section therefore appeared crooked in the middle. This problem was remedied by using `fill` and `unfill` commands (lines 82-85) instead of a single draw command.

A final problem was discovered after the other characters had been produced. This problem is illustrated by the Bishop below on the left.

The Bishop on the left has a small overlap of the lines at the miter's base. The solution was to use the up directional command at that point (75 and 78).

Given this description, the complete code used to compose all four versions of Bishop is included below.

```
1. %Components needed for D. Knuth's
2. %KEEPIT dirty trick.
3. %
4. picture extrapic;
5. boolean currentnull, extranull;
6. def clearit = currentpicture:= extrapic;
7.  currentnull:=extranull;
8.  extrapic:=nullpicture;
9.  extranull:=true; enddef;
10.
11. def keepit =
12.  cull currentpicture keeping (1,infinity);
13.  extrapic:=currentpicture;
14.  extranull:=currentnull; enddef;
15.
16. %Define LaserJet device parameters.
17. %
18. mode_setup;
19.  em#:=1/3in#;
20.  thin#:=.01em#; thick#:=.02em#;
21.  define_pixels(em,cap,ext,dep);
22.  define_blacker_pixels(thin,thick);
23.  curve_sidebar=round 1/18em;
24.
25. %Draw vertical hatching background for
26. %pieces on black squares.
27. %
28. def hatch(expr dummy) =
29.  pickup pencircle scaled thin;
30.  for i=0 upto 22:
31.   draw (w*i/22,0)--(w*i/22,h);
32.  endfor
33.  enddef;
34.
35. %Point locations as a percentage of the
36. %full character width.  #'s 1-5 define
37. %the left cloth strip, 6-10 the right
38. %cloth strip, 11-16 the left side of
39. %the miter, 16-17 the top circle, 18-20
40. %and 26-27 the right side of the miter,
41. %21-25 were are an unused second miter
42. %section, 28-29 details at the base of
43. %the miter, 30-37 cross in the center
44. %of the miter.
45. %
46. pair w[];
47.  w1=(7,9); w2=(10,15);
48.  w3=(16,16); w4=(38,10);
49.  w5=(46.5,20); w6=(53.5,20); w7=(62,10);
50.  w8=(84,16);   w9=(90,15);   w10=(93,9);
51.  w11=(35,20);  w12=(35,30);  w13=(29,43);
52.  w14=(29,55);  w15=(38,75);  w16=(50,86);
53.  w17=(50,95);  w18=(62,75);  w19=(71,55);
54.  w20=(71,43);  w21=(65,75);  w22=(61,73);
55.  w23=(72,55);  w24=(69,55);  w25=(69,43);
56.  w26=(65,30);  w27=(65,20);  w28=(42,30);
57.  w29=(58,30);  w30=(39,56);  w31=(61,56);
58.  w32=(50,39);  w33=(50,70);  w34=(41,56);
59.  w35=(59,56);  w36=(50,41);  w37=(50,68);
```

```
%The White Bishop on a White Square
%
beginchar(4,em#,em#,0); "White Bishop";
 for i=1 upto 37: z[i]=h/100*w[i]; endfor
 penpos1(4thick,-20); penpos2(4thick,-50);
 penpos3(4thick,-70); penpos4(4thick,-70);
 penpos5(4thick,0);   penpos6(4thick,0);
 penpos7(4thick,60);  penpos8(4thick,60);
 penpos9(4thick,50);  penpos10(4thick,40);
 penpos30(3thick,90); penpos31(3thick,90);
 penpos32(3thick,0);  penpos33(3thick,0);
 penpos34(thick,90);  penpos35(thick,90);
 penpos36(thick,0);   penpos37(thick,0);
 pickup pencircle scaled thick;
 draw z1r..z2r..z3r..z4r..{up}.5[z5,z6]--
      z5l..z4l..z3l..z2l..z1l--cycle;
 draw z6r..z7r..z8r..z9r..z10r--z10l..z9l..
      z8l..z7l..{up}.5[z5,z6]--cycle;
 draw z11--z12..z13..z14..z15..z16--z16..
      z18..z19..z20..z26--z27--cycle;
 draw z16..z17..cycle;
 fill z30r--z31r--z31l--z30l--cycle;
 fill z32r--z33r--z33l--z32l--cycle; cullit;
 unfill z34r--z35r--z35l--z34l--cycle;
 unfill z36r--z37r--z37l--z36l--cycle;
 pickup pencircle scaled thin;
 draw z12--z28--z5l--z28--z29--z6r--z29--z26;
showit; keepit;
endchar;

%The White Bishop on a Black Square
%(All that is necessary is hatching,
%unfilling the area to be filled be the
%previously drawn character, and using
%addto.)
%
beginchar(10,em#,em#,0); "Wht Bish on Blk";
 keepit; currentpicture:=nullpicture;
 for i=1 upto 37: z[i]=h/100*w[i]; endfor
 penpos1(4thick,-20); penpos2(4thick,-50);
 penpos3(4thick,-70); penpos4(4thick,-70);
 penpos5(4thick,0);   penpos6(4thick,0);
 penpos7(4thick,60);  penpos8(4thick,60);
 penpos9(4thick,50);  penpos10(4thick,40);
 hatch(1);
 unfill z1r..z2r..z3r..z4r..z5r--z5l..z4l..
        z3l..z2l..z1l--cycle;
 unfill z6r..z7r..z8r..z9r..z10r--z10l..
        z9l..z8l..z7l..z6l--cycle;
 unfill z11--z12..z13..z14..z15..z16--z16..
        z18..z19..z20..z26--z27--cycle;
 unfill z16..z17..cycle; cullit;
 addto currentpicture also extrapic;
 pickup pencircle scaled thick;
showit;
endchar;

beginchar(16,em#,em#,0); "Black Bishop";
 for i=1 upto 33: z[i]=h/100*w[i]; endfor
 penpos1(4thick,-20); penpos2(4thick,-50);
 penpos3(4thick,-70); penpos4(4thick,-70);
 penpos5(4thick,0);   penpos6(4thick,0);
 penpos7(4thick,60);  penpos8(4thick,60);
 penpos9(4thick,50);  penpos10(4thick,40);
 penpos30(2thick,90); penpos31(2thick,90);
 penpos32(2thick,0);  penpos33(2thick,0);
 fill z1r..z2r..z3r..z4r..z5r--z5l..z4l..
      z3l..z2l..z1l--cycle;
 fill z6r..z7r..z8r..z9r..z10r--z10l..
      z9l..z8l..z7l..z6l--cycle;
 fill z11--z12..z13..z14..z15..z16--z16..
      z18..z19..z20..z26..z27--cycle;
 fill z16..z17..cycle;
 cullit;
 unfill z30r--z31r--z31l--z30l--cycle;
 unfill z32r--z33r--z33l--z32l--cycle;
 pickup pencircle scaled thick;
 erase draw z12--z26--z29--z6r--z5l--z28;
showit; keepit;
endchar;

beginchar(22,em#,em#,0); "Blk Bish on Blk";
 for i=1 upto 33: z[i]=h/100*w[i]; endfor
 penpos5(4thick,0); penpos6(4thick,0);
 penpos30(2thick,90); penpos31(2thick,90);
 penpos32(2thick,0);  penpos33(2thick,0);
 hatch(1);
 unfill z30r--z31r--z31l--z30l--cycle;
 unfill z32r--z33r--z33l--z32l--cycle;
 pickup pencircle scaled thick;
 erase draw z12--z26--z29--z6r--z5l--z28;
showit;
endchar;

stop"";
end;
```

To conclude, the chess font described was designed to be useful for a variety of chess publication needs. It is hoped the font provided is robust enough to avoid others having rework this same problem later. I also hope this example of the METAFONT program's capabilities should stimulate interest in other applications along these lines. One such application might extend TEX into the area of archeology by way of a font for hieroglyphics or cuneiform.

Any comments, suggestions, or requests regarding this font are welcome. I can be reached at the address below. Unfortunately I do not have access to electronic mail.

◇ David Tofsted
P. O. Box 6926
Las Cruces, NM 88006