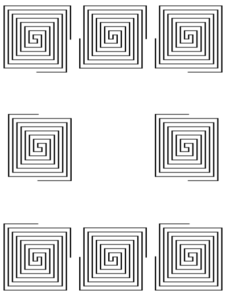
METAFONT

Ramki spiralne

Bogusław Jackowski*
i Tomasz Przechlewski**

Projektowanie ramek to jedno z prostszych zadań, które można zrealizować wykorzystując METAFONT-a. Przedstawione niżej projekty ramkek, z uwagi na zwartość kodu, są prezentowane w całości. Z tego też powodu dodano w kilku miejscach komentarze. Do wykreślenia spiralnej ramki należy zaprojektować osiem następujących znaków:



Program METAFONTowy rozpoczyna się (pomijając sakramentalne mode_setup) od nadania wartości czterem parametrom:

- px określa szerokość linii poziomych,
- py określa szerokość linii pionowych,
- leap odległość między zwojami,
- spir liczba elementów projektowanego znaku.

```
mode_setup;
px=round(.4pt); % pen x-dimen
py=round(1.2pt); % pen y-dimen
leap=round(2.5pt); spir=15;
```

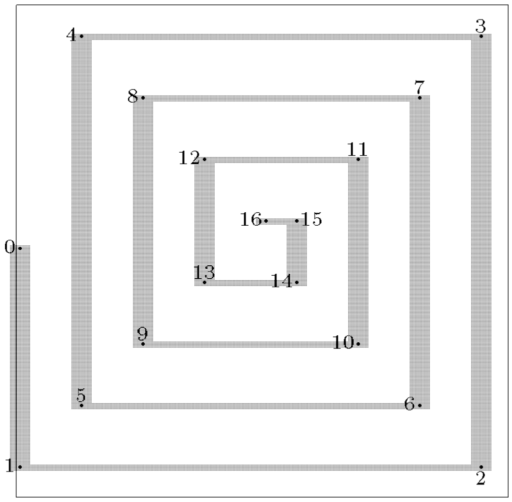
Na podstawie tych informacji obliczana jest wielkość znaku. Jest ona zależna od rozdzielczości urządzenia, na które ramka jest generowana. Przykładowo dla urządzenia o rozdzielczości 300dpi designsize$\approx$ 38,54 pt, dla 600dpi designsize$\approx$ 40,47 pt, a dla rozdzielczości 72dpi nawet $\approx$ 32,12 pt. Dzięki tej procedurze designsize (określający w tym przypadku szerokość i wysokość znaków) jest krotnością piksela. Chodziło w tym wszystkim o uniknięcie zjawiska przesuwania znaków przez sterowniki przy ich pozycjonowaniu. Gdy szerokość znaku nie stanowi całkowitej wielokrotności rozmiaru piksela występuje zjawisko określane potocznie jako „pływanie" znaków. Eliminując pływanie gwarantujemy, że sąsiadujące ze sobą znaki będą względem siebie ustawiane przez sterownik identycznie, niezależnie od ich położenia na stronie.

```
font_size:=((spir+1)*leap)/hppp;
message "designsize =
        " & decimal designsize & "pt";
message "";
```

Wszystkie znaki są generowane poprzez wywołanie makrodefinicji fix_spir. Począwszy od brzegu kwadratu linia labiryntu jest zakręcana do środka.

```
vardef fix_spir@# text t =
% |t| is an appropriate rotation
 n:=0; pair uv;
% wrapping:
 z0=(.5px,.5h) t;
 z[incr n]=(.5px,leap-.5px) t;
 uv:=(right t)-(origin t);
 for i:=spir*leap step -leap until leap:
  z[incr n]=z[n-1]+i*uv;
  uv:=uv rotated 90;
 endfor
```

Na tym etapie wygenerowany znak wyglądałby następująco:



Ale to nie koniec. Teraz linia jest rozwijana od środka z powrotem do brzegu, za pomocą następnej instrukcji iteracyjnej.

*: METAFONTologia
**: Reszta

```
20. % unwrapping:
21. z[incr n]=z[n-1]+leap*uv;
22. uv:=uv rotated -90;
23. for i:=leap step leap until
24.     (spir if (str @#)="’": +1 fi)*leap:
25.  z[incr n]=z[n-1]+i*uv;
26.  uv:=uv rotated -90;
27. endfor
28. z[incr n]=if (str @#)="’":
29.    (.5w,-.5px) else: (w+.5px,.5h) fi t;
30. labels(range 0 thru n);
```

Na końcu makrodefinicji `fix_spir` definiowana jest ścieżka `skeleton` zawierająca wszystkie, niezbędne do wykreślenia znaku punkty.

```
31. path skeleton;
32. skeleton=z0 for i:=1 upto n:
33.                --z[i] endfor;
34. enddef;
```

Druga makrodefinicja rysuje zwoje znaku „przygotowane” w poprzedniej:

```
35. def draw_spir =
36. pickup pensquare xscaled py;
37.                  yscaled px;
38. draw skeleton;
39. enddef;
```

a trzecia, służąca wyłącznie do skrócenia kodu, określa transformację, przekazywaną jako drugi argument w makrodefinicji `fix_spir`.

```
40. def trs(expr a)(expr c,d) =
41.    rotatedaround ((.5w, .5h), a)
42.    shifted (c,d);
43. enddef;
```

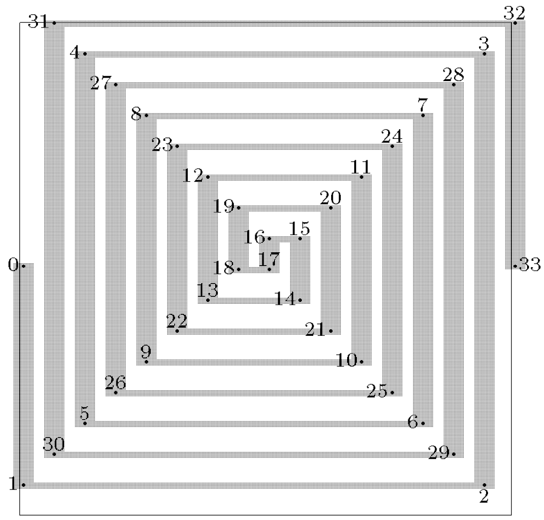
Górny środkowy znak jest zdefiniowany następująco:

```
44. beginchar("0",((spir+1)*leap)/hppp,
45.           ((spir+1)*leap)/hppp,0);
46. fix_spir;
47. draw_spir;
48. endchar;
```

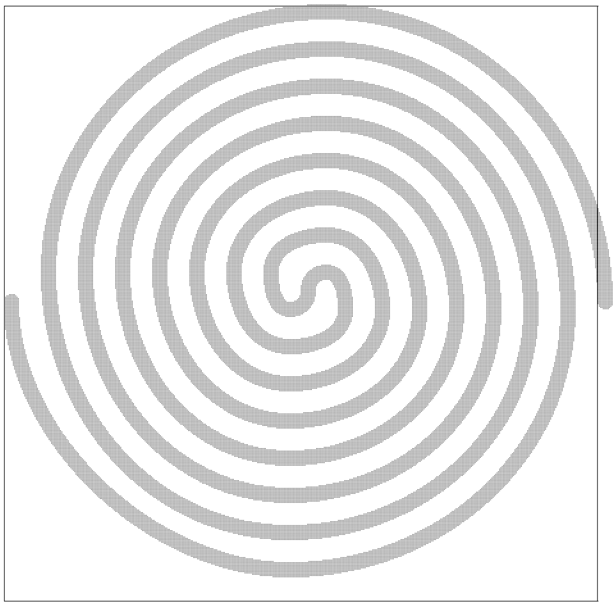
Pozostałe znaki są definiowane identycznie za wyjątkiem linii 46 zawierającej makro `fix_spir` (oczywiście zmienia się także pierwszy argument makra `beginchar` — pozycje poszczególnych znaków są jednak zupełnie nieistotne i mogą być dowolnie ustalone, w zależności od *gustu* programującego). Makro `fix_spir` ma dwa paramety, z których pierwszy jest typu `suffix` a drugi `text`. Suffiksem jest albo prim `’` albo jest on pusty (pełni on tutaj rolę zmiennej logicznej — por. wiersze 24 i 28). Argument tekstowy w dwóch przypadkach także jest pusty, a w pozostałych jest nim odpowiedni obrót wokół środka znaku, oraz przesunięcie.

- górny prawy `fix_spir’`.
- boczny lewy `fix_spir  trs(-90,px,0);`
- dolny prawy `fix_spir’ trs(-90,px,0);`
- dolny środkowy `fix_spir trs(-180,px,-px);`
- dolny lewy `fix_spir’ trs(-180,px,px);`
- boczny prawy `fix_spir trs(-270,0,-px);`
- górny lewy `fix_spir’ trs(-270,0,-px);`

Efekt końcowy dla górnego środkowego znaku prezentuje się (w dużym powiększeniu) następująco:



Wystarczy niewielka modyfikacja (jaka?) końcowej części definicji `skeleton` i makrodefinicji `draw_spir` aby otrzymać następujący ciekawy rezultat:



Teraz kilka słów na temat wykorzystania fontu w TEXu. Niech ilustracją tego będzie ogłoszenie o konkursie. Oto kod TEXowy, nieco trickowy:

```
\font \S spirama
\font \BF pldunh10

\def\0{\hbox{\phantom{\S0}}}

\newbox\tline % górna ramka
\newbox\bline % dolna ramka
\newbox\rline % prawa ramka
\newbox\lline % lewa ramka
\newbox\frame
\newbox\text

\setbox\tline
 \hbox{\S 000000000a}
\setbox\rline
 \vbox{\offinterlineskip\def~#1{\hbox{#1}}%
 \S~1~1~1~1~1~1~1~1~1~1~1~1~1~1~b}
\setbox\bline\hbox{\S c222222222}
\setbox\lline
\vbox to\ht\tline{\offinterlineskip
 \def~#1{\hbox{#1}}%
 \S~d~3~3~3~3~3~3~3~3~3~3~3~3~3~3~3\vss}

\setbox\frame\vbox{%
\hbox{\copy\lline\copy\tline}
\hbox{\copy\bline\copy\rline}}
```

Teraz znając wymiary ramki, tekst zawieramy w pudełku o identycznych wymiarach:

```
\long\def\Lcenterline#1{\line{\hss#1\hss}}

\setbox\text\vbox to\ht\frame{%
\hsize=\wd\frame\baselineskip12pt
\vfil
\Lcenterline{\vbox{\hsize=.7\hsize%
\centerline{\BF Konkurs na najpiękniejszą}
\centerline{\BF ramkę lub inny...}}}
...
\vfil}
```

Pudełko `\frame` zawierające ramkę kopiujemy, zaś operacja `\llap{\box\text}` powoduje nałożenie pudełka z tekstem na pudełko ramki:

```
\noindent\copy\frame\llap{\box\text}
```

Pełny tekst ogłoszenia o konkursie obok. REDAKCJA OCZEKUJE LAWINY ZGŁOSZEŃ. Do dzieła!

Konkurs na najpiękniejszą
ramkę lub inny znak wykonany
METAFONTem

Redakcja Biuletynu ogłasza konkurs na najładniejsze znaki graficzne wykonane METAFONT-em. • Znaki mogą być wszelakiego rodzaju: ramki, loga, itp. • Termin nadsyłania propozycji 31 Stycznia 1994. • Najlepsze zostaną zaprezentowane w Zeszycie 2 (1) oraz nagrodzone książkami METAFONT-owo-TEX-owymi.

Znak powinien być oryginalnym pomysłem autora. • Można zgłaszać wiele prac. • Styl programu METAFONT-owego będzie ważył znacznie na ocenie. • Do źródła METAFONT-owego należy dołączyć opis programu.

Zgłoszenia kierować na adres redakcji

The Best Border Design
or Other Dingbat Character
Competition

GUST announces a competition for the best dingbat character created with METAFONT. • A character could be: border design, logo, etc. • The deadline for submissions: 31 January 1994. • The best works will be published in Zeszyt 2 (1) of GUST and rewarded with books on TEX and METAFONT.
The character should be of the author's original design. • Each submission can contain several characters. • METAFONT programming style will be considered while awarding prizes • Each submission shall be accompanied by description explaining its meaning and commenting interesting parts of METAFONT source code.

Submissions should be posted to the editors of GUST bulletin.