Piórkiem i METAPOST-em czyli Antykwa Półtawskiego

Janusz Marian Nowacki
J.Nowacki@gust.org.pl

Streszczenie

Wieloletnie próby opracowania polskiej czcionki ukoronował w latach 1923–1928 grafik i typograf Adam Półtawski. W artykule omówiono problemy związane z opracowaniem elektronicznej wersji tego kroju pisma.

Pierwsza polska czcionka – Antykwa Półtawskiego

Joachim Lelewel w pracy „Bibliograficznych ksiąg dwoje" pisał: *»Dobrze by było, żeby przyszedł czas, żebyśmy się przeświadczyli, że ani francuski, ani niemiecki druk do naszego polskiego języka nie są stosowne, że dla polskiego języka wcale własnych potrzeba druków«*.[1] Przez *»druk«* autor rozumiał tu kształt czcionki.

Próby stworzenia polskiej czcionki czynione były niemal od początku drukarstwa polskiego. Zajmowali się tym drukarze krakowscy: Ungler (gotyko-antykwa) i Wietor (kursywa polskiego kroju). W końcu XVI wieku Jan Januszowski projektował „Nowy Karakter Polski". Ciekawostka: była też książka „Nowy Karakter Polski", złożona tymże Nowym Karakterem Polskim, zawierająca traktaty o ortografii Janusza Januszowskiego, Jana Kochanowskiego i Łukasza Górnickiego. Również Stanisław Wyspiański wspólnie z drukarzem Władysławem Teodorczukiem planowali projektowanie czcionek.

Po odzyskaniu niepodległości w gronie współpracowników czasopisma „Grafika Polska" zastanawiano się również nad tym zagadnieniem. A. Półtawski tak wspomina te dyskusje: *»Wszyscy czuliśmy potrzebę takiej czcionki, ale nie umieliśmy sprecyzować, na czym jej polskość powinna i może polegać«*.

Prof. Adam Półtawski cały czas zajmował się badaniami i projektowaniem nowego pisma w oparciu o XVIII wieczne antykwy. Swoje prace ukończył w 1928 roku w Krościenku. Produkcji nowego pisma podjęła się odlewnia czcionek Jana Idźkowskiego[2] (stało się to w 1931 r.). Wytwórnia reklamowała Antykwę Półtawskiego jako krój uwzględniający nowoczesną technikę drukarską, łączący czytelność i przejrzystość druku w języku polskim z estetycznym wyglądem graficznym. Pismo odlewano w stopniach od 6 do 48 punktów.

[1] Cytaty pochodzą z pracy Janusza Sowińskiego „Adam Półtawski – Typograf artysta", Ossolineum, 1988.

[2] Była to właściwie odlewnia Tadeusza Drozdowskiego, do roku 1921 wspólnika w firmie Idźkowskiego.

Narzędzia do generowania fontów

Do dyspozycji mamy wiele programów, za pomocą których możemy generować fonty PostScript-owe. Najczęściej używany i znany jest chyba Fontographer. Wielu zwolenników posiadają też: FontLab, TypeDesigner czy FontMonger. Jak wszystko, co windowsowe, są „lekkie, łatwe i przyjemne", lecz nie do końca...

Wymienione programy są bardzo dobre do dokonywania poprawek (np. polonizacji) w już istniejących fontach. Trudniej jednak stworzyć w nich font od początku do końca. Ich możliwości „rysunkowe" są znacznie mniejsze od znanego wszystkim Corela. W praktyce więc poszczególne litery można rysować w Corelu, eksportować do formatu `*.eps`, następnie importować w np. programie Fontographer.

Praktycznie każdy font składa się ze znaków zbudowanych z powtarzających się elementów (fragmenty krzywych, szeryfy itp.). Tworząc nowy font nie wiemy do końca jaka powinna być np. szerokość pionowych kresek liter (*stemów*), jaki dokładnie kształt szeryfów itd. Często bywa tak, że po zakończeniu całej pracy stwierdzamy, że *stemy* powinny być o 5% szersze. W programach windowsowych czeka nas w takiej sytuacji kilkaset kliknięć we wszystkich znakach. Po czym stwierdzamy, że lepiej byłoby jednak poszerzyć je nie o 5% lecz o 10%. Zabawa zaczyna się od początku.

To oczywiście nie wszystkie (w moim odczuciu) wady wspomnianych narzędzi. Poprzestańmy jednak na tym, gdyż tematem niniejszego referatu nie jest krytyka programów windowsowych. Wspomnieć jednak warto, że w świecie Linuxowo/Unixowym jest jeszcze gorzej, gdyż brak tam jakichkolwiek narzędzi do tworzenia fontów PostScript-owych (wg mojej wiedzy).

Konkluzja może być tylko jedna. Poszukać trzeba innego sposobu generowania fontów. Poszu-

Polska Grupa Użytkowników Systemu TEX, 1999 (http://www.GUST.org.pl)

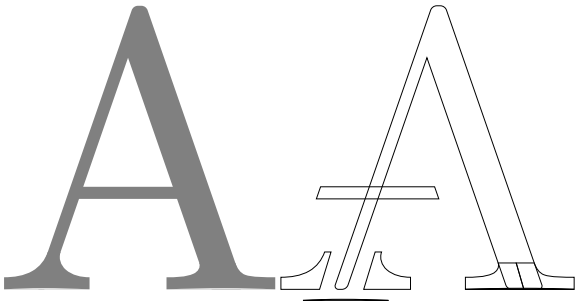
kiwania te doprowadziły mnie do programu METAPOST. Jeżeli w świecie TEX-owym problem rozwiązuje METAFONT, jeżeli METAPOST jest odpowiednikiem METAFONT-a, tyle że na wyjściu mamy kod PostScript-owy, to chyba to jest to.

Czym jest METAPOST?

METAPOST jest narzędziem programowania, opracowanym przez Johna D. Hobby'ego, służącym do generowania rysunków wektorowych. Zasady języka są bardzo zbliżone do systemu METAFONT, którego autorem jest Donald E. Knuth. Zasadniczą różnicą między obu programami jest format danych wyjściowych. METAFONT tworzy mapy bitowe, natomiast METAPOST generuje pliki PostScript-owe, zbliżone formalnie do formatu `*.eps`. Pliki te mogą być bez problemu dołączane do dokumentów TEX-owych.

METAPOST ułatwia użytkownikowi dostęp do wszystkich funkcji języka PostScript. Można w nim również łączyć tekst z grafiką. METAPOST, mając rodowód METAFONT-owy, jest wspaniałym generatorem rysunków dla potrzeb fontów zgodnych ze standardem Adobe Type 1.

Trzeba jednak przestrzec potencjalnych użytkowników przed bezkrytycznym przenoszeniem metod tworzenia liter z systemu METAFONT do programu METAPOST. Musimy pamiętać o podstawowej zasadzie konstrukcji fontów PostScript-owych: „Każda litera składa się z jednej tylko ścieżki, mogącej się składać z podścieżek", oraz że poszczególne podścieżki nie mogą na siebie częściowo zachodzić (*overlap*) i przecinać. D. Knuth nie musiał tych zasad przestrzegać, tworząc bitmapowe fonty Computer Modern. Efekt jest taki, że np. majuskuła „A" składa się z ośmiu podścieżek z nakładkami. Ilustruje to poniższy przykład:



METAPOST w przykładach

METAPOST, jak każdy niemal język programowania nie jest łatwym narzędziem dla początkującego użytkownika. Jednak po praktycznym przyswojeniu nawet niewielu jego funkcji i zasad, uzyskujemy wiele satysfakcji z wykonanych zadań.

Oczywiście tekst niniejszy nie jest podręcznikiem użytkownika programu. Na kilku prostych przykładach chcę jedynie pokazać jakie są jego możliwości. Szczególnie interesuje mnie kontekst fontowy zagadnienia.

W pierwszej kolejności przypisujemy wartości zmiennym, za pomocą których będziemy budować poszczególne litery. Już ten pierwszy krok daje nam wiele możliwości wpływania na kształt poszczególnych znaków. Po zaprogramowaniu obwiedni liter będziemy mogli, zmieniając jedynie wartości przypisane zmiennym, zmieniać kształty, nie zmieniając programu liter. A więc do dzieła:

```
vstem=120;  % szerokość pionowych stemów
hstem=30;   % wysokość poziomych stemów
oczko=280;  % szerokość oczka liter
EX=750;     % wysokość majuskuł
prz=12;     % przestrzał pionowy liter
            % np. ,,O'' (ang. overshoot)
```

W METAPOST możemy stosować jednostki miary takie jak w TEX-u. Jeżeli nie określimy jednostki, zostanie zastosowana domyślna, tj. bigpoint (`bp`).

Następnie tworzymy METAPOST-owy opis znaków, każdą literę opisując między parą `beginfig` i `endfig`. Liczba wpisana jako argument `beginfig` stanie się rozszerzeniem wynikowego pliku PostScript-owego. Jego nazwą będzie nazwa naszego pliku METAPOST-owego.

Najłatwiejsza chyba jest konstrukcja majuskuły „O". Można to zrobić na wiele sposobów. Najprościej byłoby tak:

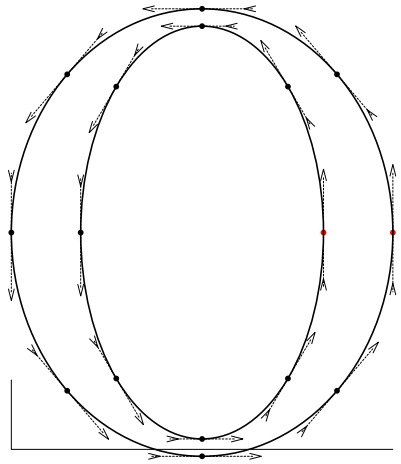
```
beginfig(1);
width_char=2vstem+1.5oczko;
draw (fullcircle xscaled width_char
        yscaled (EX+2prz)
        shifted (.5width_char,.5EX)
        );
draw (fullcircle xscaled (width_char-2vstem)
        yscaled (EX+2prz-2hstem)
        shifted (.5width_char,.5EX)
        );
endfig;
```

Wynikiem przetworzenia takiego zapisu będzie plik `nazwa.1`, który możemy obejrzeć poniżej. Dla wyjaśnienia, co poleciliśmy METAPOST-owi wykonać.

Przypisaliśmy wartość zmiennej `width_char`, która określa szerokość litery (2 szerokości pionowego stema plus 1.5 oczka). Następnie poleciliśmy narysowanie (`draw`) okręgu przeskalowanego w poziomie (`xscaled`) na szerokość litery i w pionie (`xscaled`) na wysokość majuskuł powiększoną o górny i dolny przestrzał. Całość należy przesunąć (`shifted`), gdyż inaczej nasze „O" byłoby wycentrowane wokół zerowej pozycji układu współrzędnych.

Polska Grupa Użytkowników Systemu TEX, 1999 (http://www.GUST.org.pl)

Podobnie skonstruowana jest wewnętrzna ścieżka litery, z uwzględnieniem poziomych i pionowych stemów.



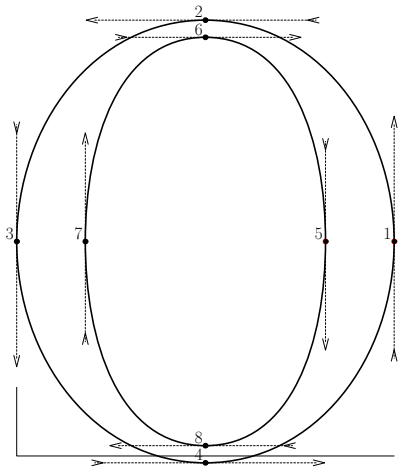
Uzyskany kształt znaku jest oczywiście poprawny. Mamy jednak za dużo odcinków krzywych Béziera, tak jednak METAPOST rozumie realizację polecenia `fullcircle`. [3]

Do prawidłowego opisania litery „O” wystarczą nam po cztery węzły krzywych Beziera na jedną ścieżkę. Ich pozycję określamy przypisując zmiennym „z” wartości współrzędnych `x` i `y`. Następnie od węzła do węzła rysujemy ścieżkę.

```
z1=(width_char,.5EX);
z2=(.5width_char,EX+prz);
z3=(0,y1);
z4=(x2,-prz);
z5=(x1-vstem,y1);
z6=(x2,y2-hstem);
z7=(x3+vstem,y3);
z8=(x4,y4+hstem);
%
draw z1..z2..z3..z4..cycle;
draw reverse (z5..z6..z7..z8..cycle);
```

Zapis `z1..z2..z3..z4 itd.` nakazuje METAPOST-owi narysowanie krzywych Beziera kolejno między tymi punktami, z automatycznym wyliczeniem punktów kontrolnych krzywych, `cycle` natomiast to nic innego jak zamknięcie ścieżki. Komenda `reverse` odwraca kierunek przebiegu krzywych. W fontach PostScript-owych kolejne, nakładające się ścieżki muszą mieć przeciwny kierunek. Firma ADOBE zaleca, aby pierwsza ścieżka podlegająca wypełnieniu miała kierunek odwrotny do ruchu zegara. Otrzymamy teraz taki wynik:

[3] Całkiem ładną literę „O” można skonstruować używając tylko czterech łuków Béziera, podczas gdy do dobrego (i elastycznego) przybliżenia elipsy potrzeba ośmiu. Brak obrotu wewnętrznej elipsy względem zewnętrznej mógłby drażnić wysublimowane powonienie typografów. Nie we wszystkich krojach.



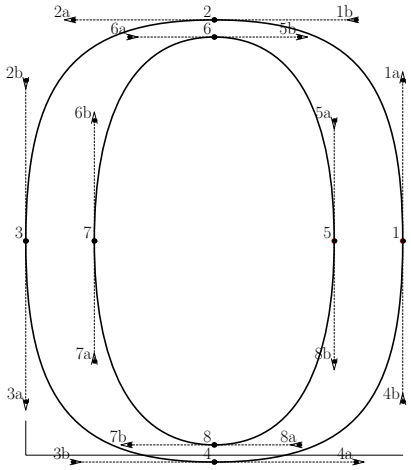
Taki kształt majuskuły „O” może nam odpowiadać lub nie, ale jego opis PostScript-owy jest jak najbardziej zgodny z normami określonymi przez standard Adobe Type 1.

Gdybyśmy potrzebowali krzywych innego kształtu, możemy METAPOST-owi powiedzieć nie tylko gdzie znajdują się poszczególne węzły, ale również jakie ma być położenie punktów kontrolnych krzywych Beziera.

```
hnode=.375width_char;
vnode=.375EX;
%
z1=(width_char,.5EX);
z1a=(x1,y1+vnode); z1b=(x2+hnode,y2);
z2=(.5width_char,EX+prz);
z2a=(x2-hnode,y2); z2b=(x3,y3+vnode);
z3=(0,y1);
z3a=(x3,y3-vnode); z3b=(x4-hnode,y4);
z4=(x2,-prz);
z4a=(x4+hnode,y4); z4b=(x1,y1-vnode);
%
z5=(x1-vstem,y1);
z5a=(x5,y5+.75vnode);  z5b=(x6+.75hnode,y6);
z6=(x2,y2-hstem);
z6a=(x6-.75hnode,y6);  z6b=(x7,y7+.75vnode);
z7=(x3+vstem,y3);
z7a=(x7,y7-.75vnode);  z7b=(x8-.75hnode,y8);
z8=(x4,y4+hstem);
z8a=(x8+.75hnode,y8);  z8b=(x5,y5-.75vnode);
%
draw (reverse (z1..controls z1a and z1b..
               z2..controls z2a and z2b..
               z3..controls z3a and z3b..
               z4..controls z4a and z4b..
               cycle));
draw (z5..controls z5a and z5b..
     z6..controls z6a and z6b..
     z7..controls z7a and z7b..
     z8..controls z8a and z8b..
     cycle);
```


Polska Grupa Użytkowników Systemu TEX, 1999 (http://www.GUST.org.pl)

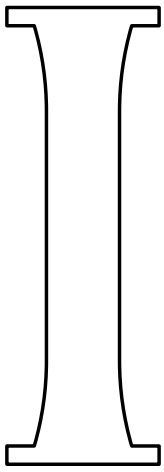
Myślę, że zapis ten nie wymaga już szerszego omówienia, szczególnie gdy porównamy go z wynikiem przetworzenia METAPOST-owego:



To tylko trzy recepty na majuskułę „O”. W istocie, jak to w językach programowania, sposobów tych jest ∞.

Podobnie na kilka sposobów można skonstruować majuskułę „I”.

```
z1=(-.5vstem,kor_v_stem);
z2=(x1-kor_h_stem,hstem);
z3=(x2-dserif,y2);  z4=(x3,0);
z5=(-x4,0);   z6=(-x3,y3);
z7=(-x2,y2);  z8=(-x1,y1);
hserif=2dserif+2kor_h_stem+vstem;
path serif_path;
serif_path = z1{down}..z2--z3--z4--z5--
             z6--z7..{up}z8;
def serif (expr x,y,z)
          serif_path rotated z˜shifted (x,y)
          enddef;
draw serif(.5hserif,0,0)--serif(.5hserif,EX,
          180)--cycle;
```



Wykorzystanie wyników pracy METAPOST-a

Po zaprojektowaniu i zaprogramowaniu wszystkich występujących w foncie znaków, METAPOST wygeneruje nam poszczególne pliki wyjściowe, każdy z opisem innej litery. Aby z nich skorzystać, zajrzyjmy do ich treści. Opis PostScript-owy tam zawarty jest tylko surowcem, na podstawie którego będziemy mogli uzyskać kod wymagany przez pliki `*.pfb`.

Po pierwsze METAPOST jest zbyt dokładny. Pliki `*.pfb` tolerują tylko liczby całkowite.

Pod drugie METAPOST wykorzystuje jedynie podstawowe operatory PostScript-owe jak `lineto` i `curveto` do rysowania, natomiast pliki `*.pfb` wymagają bardziej zaawansowanych poleceń. Stosowane są zarówno te bardziej znane, jak `rlineto`, jak i stworzone specjalnie dla opisu fontowego `hlineto` czy `rrcurveto` itp., nie jest zaś stosowana np. komenda `curveto`.

Generalnie, argumenty stosowanych w plikach `*.pfb` poleceń rysowania są obliczane w odniesieniu do poprzednio występującego argumentu.

Dla przykładu fragment wyprodukowanego przez METAPOST-a kodu PostScript-owego opisujący literę „I” wygląda następująco:

```
newpath 250 0 moveto
250 30 lineto
205 30 lineto
191.73401 75.4834 185 122.62146 185 170 curveto
185 170.03337 185 579.96663 185 580 curveto
185 627.37854 191.73401 674.5166 205 720 curveto
250 720 lineto
250 750 lineto
0 750 lineto
0 720 lineto
45 720 lineto
58.26599 674.5166 65 627.37854 65 580 curveto
65 579.96663 65 170.03337 65 170 curveto
65 122.62146 58.26599 75.4834 45 30 curveto
0 30 lineto
0 0 lineto
 closepath
```

Natomiast dla potrzeb fontu PostScript-owego powinno być to zapisane w taki sposób:

```
/I{
        35 320 hsbw
        75 100 vstem
        250 hmoveto
        30 vlineto
        -60 hlineto
        -10 45 -5 47 0 46 rrcurveto
        364 vlineto
        0 46 5 47 10 45 rrcurveto
        60 hlineto
```

```
30 vlineto
-250 hlineto
-30 vlineto
60 hlineto
10 -45 5 -47 0 -46 rrcurveto
-364 vlineto
0 -46 -5 -47 -10 -45 rrcurveto
-60 hlineto
-30 vlineto
closepath
endchar
} ND
```

Oczekiwany przez nas wynik osiągniemy najszybciej za pomocą programu AWK (GAWK), przygotowując odpowiedni, niezbyt skomplikowany skrypt.

Pliki `*.pfb` w zasadzie składają się z bardzo rozbudowanego prologu, skromnego zakończenia oraz takich j.w. definicji poszczególnych znaków zawartych w foncie. Definicji tych może być znacznie więcej niż 256, co znaczy tyle, że zmieniając tabelę encodingu, wybieramy 256 znaków (w zasadzie nieco mniej), z których aktualnie będzie można korzystać.

Do plików wynikowych METAPOST-a możemy również wysyłać dodatkowe informacje o np. *kerningu*, szerokości znaków, marginesach itp. W zasadzie można w nich umieścić wszystkie informacje potrzebne do utworzenia fontów (plików `*.pfb` i `*.afm`). Mając te dwa pliki, korzystając z pakietu TOIL uzyskamy potrzebne w TEX-u pliki `*.tfm`, `*.enc` i `*.map`. Jeżeli potrzebny jest nam plik `*.pfm` konieczny w systemie Windows, możemy skorzystać z programu afm2pfm.

Efekt jaki jest, każdy widzi

Pisanie programów METAPOST-owych jest, szczególnie dla początkujących, dosyć trudne. Prawidłowo skonstruowany program da nam jednak wiele korzyści. Modyfikując wartość nawet jednej zmiennej można już zmienić krój pisma i otrzymać nową wersję fontu. Modyfikując kilka zmiennych można stworzyć całą kolekcję fontów. Byle nie przedobrzyć.

Antykwa Półtawskiego została zaprojektowana w wersjach: normalnej, pogrubionej i kursywie. Wprowadzane do rozpowszechniania fonty również występują w tych odmianach. Dzięki możliwościom METAPOST-a wygenerowałem dodatkowo pogrubioną kursywę. Zapewne niebawem będą następne odmiany.

Udostępniane fonty zawierają znaki według znanego układu QX. Tak jak i Antykwa Toruńska są dobrem publicznym i mogą być używane bezpłatnie.

Antykwa Półtawskiego odmiana normalna

ff fi fl ffi ffl ı ȷ ` ´ ˇ ˘ ¯ ° ¸ ß æ œ ø Æ Œ Ø ¸ ! ” # $ % & ’ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ¡ = ¿ ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [“] ˆ ˙ ‘ a b c d e f g h i j k l m n o p q r s t u v w x y z – — ˝ ˜ ¨ Ą Ć Ę Ł Ń Ś Ş Ţ \ ° Ź fk Ż · § ą ć ~ ę " ' | ł ń « » { ś } ş ţ × ÷ ź ż ‡ ∢ Ø À Á Â Ä Å © Ç È É Ê Ë Ì Í Î Ï Ð < > Ó Ô ± Ö Ų Ù Ú Û Ü Þ ® à á â ¤ ä å ç è é ê ë ì í î ï ð ⩽ ⩾ ó ô õ ö ų ų ù ú û ü ¶ þ „

Antykwa Półtawskiego odmiana pogrubiona

ff fi fl ffi ffl ı ȷ ` ´ ˇ ˘ ¯ ° ¸ ß æ œ ø Æ Œ Ø ¸ ! ” # $ % & ’ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ¡ = ¿ ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [“] ˆ ˙ ‘ a b c d e f g h i j k l m n o p q r s t u v w x y z – — ˝ ˜ ¨ Ą Ć Ę Ł Ń Ś Ş Ţ \ ° Ź fk Ż · § ą ć ~ ę " ' | ł ń « » { ś } ş ţ × ÷ ź ż ‡ ∢ Ø À Á Â Ä Å © Ç È É Ê Ë Ì Í Î Ï Ð < > Ó Ô ± Ö Ų Ù Ú Û Ü Þ ® à á â ¤ ä å ç è é ê ë ì í î ï ð ⩽ ⩾ ó ô õ ö ų ų ù ú û ü ¶ þ „

Antykwa Półtawskiego odmiana kursywa

*ff fi fl ffi ffl ı ȷ ` ´ ˇ ˘ ¯ ° ¸ ß æ œ ø Æ Œ Ø ¸ ! ” # $ % & ’ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ¡ = ¿ ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [“] ˆ ˙ ‘ a b c d e f g h i j k l m n o p q r s t u v w x y z – — ˝ ˜ ¨ Ą Ć Ę Ł Ń Ś Ş Ţ \ ° Ź Ż · § ą ć ~ ę " ' | ł ń « » { ś } ş ţ × ÷ ź ż ‡ À Á Â Ä Å © Ç È É Ê Ë Ì Í Î Ï Ð < > Ó Ô ± Ö Ų Ù Ú Û Ü Þ ® à á â ¤ ä å ç è é ê ë ì í î ï ð ⩽ ⩾ ó ô õ ö ų ų ù ú û ü ¶ þ „*

Antykwa Półtawskiego odmiana kursywa pogrubiona

ff fi fl ffi ffl ı ȷ ` ´ ˇ ˘ ¯ ° ¸ ß æ œ ø Æ Œ Ø ¸ ! ” # $ % & ’ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ¡ = ¿ ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [“] ˆ ˙ ‘ a b c d e f g h i j k l m n o p q r s t u v w x y z – — ˝ ˜ ¨ Ą Ć Ę Ł Ń Ś Ş Ţ | ° Ź fk Ż · § ą ć ~ ę " ' | ł ń « » { ś } ş ţ × ÷ ź ż ‡ ∢ Ø À Á Â Ä Å © Ç È É Ê Ë Ì Í Î Ï Ð < > Ó Ô ± Ö Ų Ù Ú Û Ü Þ ® à á â ¤ ä å ç è é ê ë ì í î ï ð ⩽ ⩾ ó ô õ ö ų ų ù ú û ü ¶ þ „

Polska Grupa Użytkowników Systemu TEX, 1999 (http://www.GUST.org.pl)