Praktyka

Problemy z ATM dla Windows NT

Aleksander Kasicki

Mój pierwszy kontakt z Adobe Type Managerem był bolesny – bodajże z CorelDRAW 3.0 dołączona była wersja ATM która nie działała. Dowiedziałem się o tym w momencie, gdy na rano miałem przygotować logo firmy, które bazowało na foncie postscriptowym. Pozornie wszystko było OK – program się instalował, widział fonty, ale nie udostępniał ich innym programom. Dopiero następna wersja działała prawidłowo.

Podobnych kłopotów nie miałem przez długi czas. Bez problemu przesiadłem się na Windows 95, najpierw używając starego, 16 bitowego ATM-a dla Windows 3.11, potem dostałem wersję 32 bitową, kolejną i kolejną i ciągle wszystko było OK. Niestety, po którymś totalnym padzie W95 i utracie kilku godzin pracy postanowiłem przesiąść się na Windows NT 4.0. Nie będę nikogo przekonywał, że zrobiłem dobrze. Wartość tego systemu doceni każdy, kto używał zawieszających się ciągle programów pod W95.

Wadą tej przesiadki był brak ATM. Wersja dla W95 nie działała, a wersja dla NT jeszcze nie istniała, co stwierdziłem z pewnym zdziwieniem. No cóż, obejść się to dało. Prace tworzyłem pod NT, teksty dopisywałem pod W95, zamieniałem na krzywe, wracałem do NT, dopieszczałem.

Z radością powitałem ATM Delux dla NT. Pomyślałem przez chwilę, że wszystko wraca do normy, ale problemy pokazały się natychmiast. Używam fontów firmy ThETA (tzw. „biblioteka Theta") w wersji TEX-owej (płyta ta, jako jedyna, została wydana w dwóch wersjach – TEX-owej i Windowsowej), nie poprawionej ThETAFixem. Gdy je kupowałem, na temat fontów wiedziałem jedno – że ich potrzebuję. Dziś wiem trochę więcej. Między innymi to, że w świecie Windows nie istnieje coś takiego, jak stuprocentowo interpretowalny PostScript.

Co takiego się dzieje z fontami postscriptowymi w Windows NT? Ano różne dziwne rzeczy. Zacznijmy od tego, że niektóre fonty nie włączają się w legalny sposób do pliku. W programach graficznych można to obejść każąc zamienić fonty na krzywe przy tworzeniu pliku PS. Ale w programach do składu (np. Page Maker) jest to niemożliwe. Taki plik PS z zaszytymi fontami albo się nie interpretuje, bo wszyty font ma błąd, albo RIP nie znajduje fontów w pliku i podmienia je jakoś. Zwykle nie krzycząc zbyt głośno, o czym sam się przekonałem. Pikanterii niech doda fakt, że znam fonty, które nie dają się wbudować do pliku pod Windows 95/NT, ale w TEX-u wszystko jest dobrze.

Spotkałem się również z fontami które nie włączały się do pliku w sposób legalny i nie dawały się zamienić na krzywe – podczas zamiany na krzywe polskie diakrytyki były zastępowane innymi znakami. Problem w tym, że wszystkie te błędy są niewidoczne na ekranie. Tu wszystko wygląda tak jak powinno, jak również wydruki są prawidłowe.

Oczywiście żadnych problemów nie ma, gdy używa się programów do obróbki bitmap – tu można pisać po polsku zawsze. I jest to jakieś wyjście w sytuacjach kryzysowych.

Z rozmów z różnymi osobami i moich obserwacji wynika, że problemem jest to, iż Windows NT używa Unicode i wie, że na świecie używa się wielu różnych języków, w tym także języka polskiego. Efekt jest taki, że jeżeli aplikacja nie potrafi przekonać systemu, że coś ma być wyświetlane po polsku, to nie da się tego zrobić. Zjawisko to występuje gdy używa się niektórych źle napisanych polskich programów (zwłaszcza encyklopedii i słowników) – nie można wpisywać polskich liter lub nie ma polskich liter w oknach. Podobne efekty dają niektóre stare aplikacje zagraniczne. Jeżeli mamy dokument napisany w Word Perfect for Windows lub AmiPro, to jedynym sposobem zachowania polskich diakrytyków przy przechodzeniu na nowe wersje oprogramowania jest konwersja w Wordzie 6, który potrafi się połapać w różnych pomysłach Microsoftu. Wersja Worda 97 już nie zobaczy polskich liter w starych tekstach i zastąpi je znakami `notdef`.

Wydaje się, że podobnie dzieje się z fontami. Jeżeli font jest w polskiej wersji, to nie można używać niektórych niepolskich znaków narodowych istniejących w foncie, bo system twierdzi, że takie znaki nie są potrzebne. I na odwrót – jeżeli font nie umie przekonać systemu, że jest w polskiej wersji, to polskie litery będą niedostępne. Problem ten firma URW++ rozwiązała dostarczając fonty w dwóch wersjach – polskiej i zachodnioeuropejskiej i w ten sposób wszystkie znaki są zawsze dostępne.

➢

Dzięki uprzejmości kilku osób przetestowałem zgodność fontów różnych firm z Windows NT. Całkowicie zgodne są fonty firm Adobe, FontFont, Linotype i URW++ – dają się włączać do pliku i zamieniać na krzywe. Fonty firmy Elsner+Flake dają się zamienić na krzywe, ale niestety nie dają się włączyć do pliku. Najśmieszniej jest z fontami firmy ThETA, a przynajmniej z dostępnymi mi fontami z „biblioteki Theta": wersja TEX-owa fontów nie włącza się do pliku i podczas konwersji na krzywe gubione są polskie litery; wersja Windowsowa, która pod W3.11 i W95 daje się włączać i zamieniać na krzywe, pod NT również daje się włączać i zamieniać na krzywe, tyle że wszystkie znaki zastępowane są pustym znakiem.

Wydawało by się, że to wszystko świadczy przeciwko fontom Type1 i jedynym wyjściem jest używanie fontów TrueType, ale całe szczęście nie jest tak pięknie. Nowe fonty TT, używające Unicode, nie naświetlają się na naświetlarkach, których RIP-y nie wiedzą co to jest Unicode.

I co w takiej sytuacji ma zrobić człowiek chcący używać TEX-a i Windows 95/NT? Dobrym wyjściem jest używanie fontów firmy URW++ – dostarczane są bowiem w postaci TrueType i Type1 w obu wersjach językowych (zachodnioeuropejskiej i polskiej). W ten sposób zawsze można jakoś wybrnąć z kłopotów. (Jedyną wadą tych fontów jest cena: pakiety EuroWorks (500 fontów) kosztują ponad 2000 zł + VAT!.)

◇ Aleksander Kasicki
akasicki@ameba.nencki.gov.pl

Jak przerobić font na KAPITALIKOWY

Jacek Mierczyński

Niedawno na liście dyskusyjnej GUST poruszana była sprawa tworzenia z danego fontu postscriptowego fontu, w którym małe litery zastąpione są kapitalikami tego samego kroju, ale nieco mniejszego stopnia. Poniżej przedstawiono prostą propozycję użycia do tego celu narzędzi zawartych w standardowych dystrybucjach TEX-a i dvips-a. Wykorzystuje ona koncepcję fontów wirtualnych i dotyczy fontów zgodnych z układem QX. Jako przykładowy wybrano qplr.

1. Załóżmy, że mamy poprawnie zainstalowany font qplr oraz programy afm2tfm z dvips-a i vptovf z dystrybucji TEX-a. Do wspólnego katalogu kopiujemy pliki qplr.afm i qpl.enc. Przyda się też plik l.qx z pakietu TOIL (BOP).

2. Po usunięciu nagłówka i komentarzy z pliku l.qx przerabiamy w nim każdy wiersz (p. instrukcja do dvips), np.

```
hyphen hyphen endash
```

zapisujemy jako:

```
% LIGKERN hyphen hyphen =: endash ;
```

Należy zwrócić uwagę na spację przed średnikiem.

Tak zmieniony l.qx dopisujemy na końcu pliku qpl.enc zmieniając w nim dla porządku również pierwszy wiersz na (przykładowo)

```
/enc-qplx[
```

oraz nazwę na qplx.enc.

Mała uwaga: definiowanie ligatur takich jak f+i → fi dla fontu kapitalikowego nie ma sensu. Na szczęście nie trzeba się tym przejmować. Przy tworzeniu KAPITALIKÓW zostaną one zignorowane przez afm2tfm, który uwzględni tylko myślniki i cudzysłowy – natomiast plik qplx.enc z definicją kompletu MEX-owych ligatur może się jeszcze przydać.

3. Mając qplr.afm i qplx.enc uruchamiamy polecenie:

```
afm2tfm qplr -t qplx.enc -V qplrc rqplr
```

w którym -V powoduje utworzenie fontu z małymi literami jako kapitalikami o wielkości 80% stopnia fontu normalnego (jest to wielkość domyślna, można ją zmienić opcją -c). Powinny powstać pliki qplrc.vpl i rqplr.tfm. Dodatkowo na ekranie pojawi się tekst:

```
rqplr QuasiPalladio-Regular
```

Tekst ten dopisujemy do pliku psfonts.map, koniecznie dodając na końcu wiersza <qplr.pfb (bez spacji za znakiem <).

4. Uruchamiamy kolejny program:

```
vptovf qplrc.vpl qplrc.vf qplrc.tfm
```

tworzący dwa pliki o nazwach jak w ostatnim poleceniu.

5. Pozostaje SPRAWDZIĆ, JAK TOTO DZIAŁA, a następnie wpisać pliki qplrc.tfm, rqplr.tfm oraz qplrc.vf do odpowiednich katalogów.

Na zakończenie jeszcze jedna uwaga. Autor nie rozumie, dlaczego czasem afm2tfm tworzy plik VPL, w którym brakuje opisu znaku P (małe „p"). W takiej sytuacji ponowne uruchomienie programu afm2tfm, bez żadnych zmian, da już poprawny wynik — brakująca litera znajdzie się na swoim miejscu (!?).

◇ Jacek Mierczyński
mier@ibwpan.gda.pl