TEX-OWA KUCHNIA

Przepis na wygodne używanie komendy \parshape

Janusz Marian Nowacki

O co chodzi?

Przypuszczalnie najrzadziej używaną komendą TEX-ową jest polecenie \parshape. Nie dzieje się tak wyłącznie z powodów estetycznych. Komenda ta należy chyba do najbardziej uciążliwych dla użytkownika. Aby zastosować ją w standardowy sposób, trzeba się liczyć z wielokrotnym powtarzaniem cyklu TEX-owego aż do uzyskania pożądanego efektu.

Wielokrotnie zastanawiałem się nad sposobem obejścia tych problemów, spowodowania choć częściowego automatyzmu. Uważam, że częściowo się to udało. Prezentuję tu nie tyle gotowe rozwiązanie na wszystkie okazje, ile metodę postępowania. Tekst niniejszy nie jest też pełnym opisem możliwych zastosowań tej komendy. Zadanie to pozostawiam „prawdziwym” specjalistom TEX-owym.

Dla przypomnienia \parshape służy do regulowania położenia i długości kolejnych wierszy składanego akapitu. Tekst można formatować w dość dowolny sposób, co może się przydać do tzw. oblewania tekstem np. grafiki o nieregularnych kształtach. Ogólna postać zastosowania jest następująca:

\parshape=n x_1 y_1 x_2 y_2 . . . .

gdzie n jest ilością wierszy, w których komenda będzie działała, natomiast $x_1 y_1$ $x_2 y_2$ itd. są parami rozmiarów (dimen) określających: pierwszy – wcięcie od lewego marginesu, drugi – długość składanego wiersza.

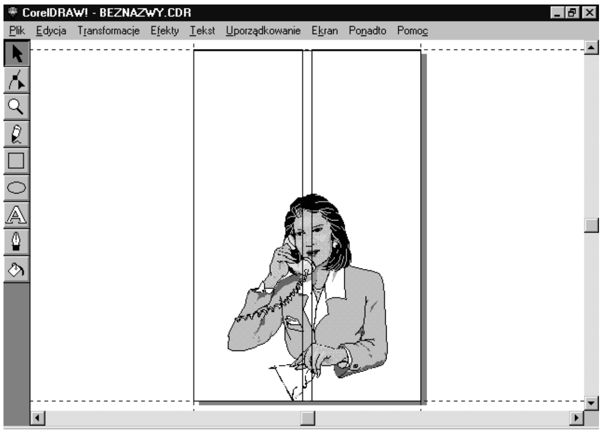
Ilość par rozmiarowych powinna odpowiadać ilości wierszy przeznaczonych dla \parshape. Jeżeli ilość tych par jest mniejsza, ostatnie linie zostaną złożone wg. ostatniej zdefiniowanej pary, jeżeli jest ich więcej – będą ignorowane. Domyślnie w pliku Plain polecenie ma ustawioną wartość zerową, która jest przywracana automatycznie na końcu każdego akapitu. Tak więc jedną instrukcją \parshape nie można objąć kilku akapitów. Lecz i na to są sposoby.

Skorzystajmy np. z CorelDraw!

Tak, tak. CorelDraw! może być przydatny. Wszystkie opisywane przykłady i ilustracje dotyczą wersji 3.0 CorelDraw!. Wersje nowsze, szczególnie piąta i szósta mają interesujące z naszego punktu widzenia funkcje `trim` i `intersection`, które w znacznym stopniu mogą zautomatyzować, interesujące dla naszych potrzeb, czynności w samym CorelDraw!.

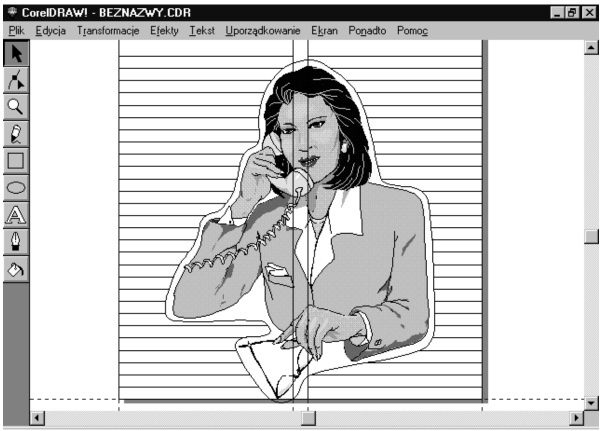
A więc do roboty.

1. W CorelDraw! otwieramy lub importujemy grafikę, która ma być umieszczona w naszym tekście. Rozmiar kartki powinniśmy zdefiniować taki, jaki w TEX-u będzie miało pole zadruku bez pagin (\hsize i \vsize). Jeżeli będzie to skład dwułamowy (większej ilości łamów nie polecam na początek) możemy liniami prowadzącymi zaznaczyć również rozmiary łamów. Grafika powinna być umieszczona w miejscu strony, w którym znajdzie się w naszym dokumencie. Punkt zerowy linijek miarowych powinien się znajdować w lewym dolnym narożniku kartki.



2. Wokół grafiki rysujemy krzywą zamkniętą odpowiadającą polu, w który nie powinien się pojawić tekst w dokumencie TEX-owym. Warto zwrócić uwagę na odpowiedni odstęp między grafiką a przewidywanym tekstem (zazwyczaj chcemy wykonać zbyt mały odstęp).
3. Na całe pole zajmowane przez samą grafikę, poczynając od dołu, wprowadzamy poziome linie o długości naszego \hsize i w odstępach pionowych wielkości naszego \baselineskip.
4. Ręcznie (CorelDraw! 3.0) lub automatycznie (CorelDraw! 5 i 6) wycinamy w liniach „okienko” na tekst. Dobrze jest po tej

operacji wszystkie linie poziome połączyć w jedną krzywą a następnie rozłączyć. Pierwsza i ostatnia linia „łamu" powinny mieć pełną długość. W ten sposób w prezentowanym przykładzie otrzymujemy dwa „łamy" linii, obrys grafiki i właściwą grafikę.



5. Wykonujemy eksport w formacie EPS samej grafiki, a następnie „lewego" łamu (również z zaznaczoną opcją „tylko wybrane") i kasujemy te obiekty. „Prawy" łam przesuwamy do lewego marginesu strony i również wykonujemy eksport. Następnie robimy jego lustrzane odbicie i ostatni już eksport. Na dysku mamy więc jeden plik grafiki oraz trzy pliki z liniami.
6. Możemy się pożegnać z CorelDraw! i wrócić do ulubionych zajęć DOS-owych. Gdy obejrzymy pliki EPS z liniami, znajdziemy sekcje z następującym przykładowym tekstem:

```
%%CURV 2
0.00 180.43 m
283.46 180.43 L
```

gdzie interesuje nas pierwsza cyfra wiersza zakończonego na literę `m` lub `L` (w tym wypadku chodzi o cyfrę 283.46). Sekcji takich w pliku PostScriptowym jest tyle ile linii poziomych utworzyliśmy w CorelDraw! i tyle też cyfr musimy wyciąć z każdego z trzech plików.

To już tradycja – zaprzęgamy AWK-a

Grzebanie edytorem tekstowym w pliku EPS jest możliwe, lecz czasochłonne. Dlatego do pomocy zaprosimy AWK-a z jednym z następujących skryptów:

Skrypt select-m.awk

```
/ m$/ {X[++n]=$1}
END {
  print n
  for (i=1; i<=n; ++i) print X[i] "bp"
    }
```

Skrypt select-l.awk

```
/ L$/ {X[++n]=$1}
END {
  print n
for (i=1; i<=n; ++i) print X[i] "bp"
    }
```

Po utworzeniu tych plików uruchamiamy poleceniem DOS-owym procedurę

```
mawk -f select-l.awk (ew. select-m.awk)
                     naszplik.eps > nazwa.txt
```

Otrzymane pliki zawierają kolumny cyfr będących argumentami komedy \parshape, czyli to na czym bardzo nam zależało. Wystarczy w pierwszej linii, na początku dopisać \parshape=. Najłatwiej utworzyć na podstawie otrzymanych danych polecenie dla „lewego" łamu naszego przykładu. Będzie ono wyglądało następująco:

```
\parshape=10 % liczba wierszy
 0bp 0.00bp
0bp 44.57bp
0bp 61.34bp
0bp 70.06bp
0bp 88.70bp
0bp 90.50bp
0bp 80.71bp
0bp 72.94bp
0bp 46.66bp
 0bp 0.00bp
Nasz akapit . . . . . .
```

Jeżeli wcięcia wierszy mają nastąpić z prawej strony to we wszystkich parach argumentów pierwszym rozmiarem jest 0bp (z lewej strony nie ma żadnych dodatkowych wcięć).

Z „prawym” łamem jest nieco więcej zabawy, bo i zachowania CorelDraw! są czasem dziwne, czasem nieprzewidywalne, zależnie od wersji programu. Tak jak dla „lewych” łamów zazwyczaj odpowiedni do wycinania danych jest skrypt `select-l.awk` tak dla „prawych” zazwyczaj będziemy korzystali z `select-m.awk`. Nasze gotowe polecenie, wykorzystujące dane z dwóch ostatnich EPS-ów, będzie wyglądało teraz tak:

```
\parshape=10
    0bp 170.00bp
44.57bp 125.43bp
61.34bp 108.66bp
70.06bp  99.94bp
88.70bp  81.30bp
90.50bp  79.50bp
80.71bp  89.29bp
72.94bp  97.06bp
46.66bp 123.34bp
    0bp 170.00bp
Nasz akapit . . . . . .
```

Z przykładu tego widać, że jeżeli wcięcia dla grafiki mamy z lewej strony składu to suma każdej pary parametrów musi się równać `\hsize` wyrażonej w bp (tylko dlatego w bp, że dane otrzymane z pliku EPS też są w tych jednostkach). Nie zawsze tak jest i trzeba przeprowadzić drobne korekty. Jeżeli będziemy pamiętać, że rozmiary z lewej kolumny liczb dotyczą głębokości wciecia, a z prawej długości linii tekstowych, bez problemu wybierzemy z plików utworzonych AWK-iem odpowiednie zestawy.

Gdzie to umieścić?

Polecenia `\parshape` z wszystkimi wymaganymi argumentami umieszczamy tuż przed akapitem w którym będą się miały pojawić wcięcia. Jeżeli mają się one zacząć dopiero od jakiegoś konkretnego wiersza to dodajemy na początku zerowe pary argumentów (przy prawych wcięciach będzie to `0bp 0bp`, przy lewych wcięciach `0bp \hsize`.

Jeżeli `\parshape` ma objąć więcej niż jeden akapit powinniśmy je oddzielać niestandardowo poleceniem

```
\hfill\break\hglue\parindent
```

dzięki czemu będą nadal jednym akapitem podzielonych na kilka fragmentów. W zasadzie powinniśmy używać `\parskip=0pt` gdyż w przeciwnym wypadku zmianie ulegnie organizacja wierszy na stronie i argumenty wstawione do komendy `\parshape` będą nieodpowiednie, chyba że je ręcznie zmienimy.

Otrzymujesz kartę kredytową i pełną swobodę; od tej pory sam możesz decydować o wyborze trasy i środka lokomocji. Tak oto rozpoczyna się niezwykle frapująca interaktywna *przygoda internetowego podróżnika.

Przed Tobą, pełna zaskakujących niespodzianek, droga z Amsterdamu do Australii. Sidney – młodzieniec nieco zepsuty i zarozumiały – lubi bowiem sprawiać kłopoty: jest nudny i przemądrzały, znika na wiele godzin, gdy tylko spuścisz go z oka lub testuje stan Twojej wiedzy zadając Ci podchwytliwe pytania, na które musisz odpowiedzieć, by można było ruszyć dalej.

Podróż przebiega w czasie realnym. Jeśli zdecydujesz się np. lecieć z Paryża do Londynu, musisz odczekać ponad dwie godziny zanim do Twojej skrzynki trafi list powiadamiający, że dotarłeś na miejsce i tym samym możesz na nowo włączyć się do gry i dalej zwiedzać świat.

W każdym mieście sporo czasu przeznaczysz na zwiedzanie; opóźni to podróż i zmniejszy Twoje konto. Przemądrzały Sidney nie przepuści żadnej okazji, by zadać Ci parę pytań, np. kiedy zburzono Mur Berliński, kto zbudował Akropol albo co to są Smoki Barcelońskie.

Otrzymujesz kartę kredytową i pełną swobodę; od tej pory sam możesz decydować o wyborze trasy i środka lokomocji. Tak oto rozpoczyna się niezwykle frapująca interaktywna przygoda internetowa.

Przed Tobą, pełna zaskakujących niespodzianek, droga z Amsterdamu do Australii. Sidney – młodzieniec nieco zepsuty i zarozumiały – lubi bowiem sprawiać kłopoty: jest nudny i przemądrzały, znika na wiele godzin, gdy tylko spuścisz go z oka lub testuje stan Twojej wiedzy zadając Ci podchwytliwe pytania, na które musisz odpowiedzieć, by ruszyć dalej. Jeśli zdecydujesz się na przykład lecieć samolotem z Amsterdamu do Paryża, musisz odczekać ponad trzy godziny zanim do Twojego komputera trafi list powiadamiający, że dotarłeś na miejsce i tym samym możesz na nowo włączyć się do gry.

W każdym mieście sporo czasu przeznaczysz na zwiedzanie; opóźni to podróż i zmniejszy Twoje konto. Przemądrzały Sidney nie przepuści żadnej okazji, by zadać Ci parę pytań, np. kiedy zburzono Mur Berliński,

Prezentowany przykład przygotowałem stosując polecenie:

```
\hbox{\hsize=60mm\vtop{
  tekst lewego łamu z komendami \parshape i \objectput
}\vtop{ tekst prawego łamu z komendą \parshape }}
```

Grafikę wstawiamy do dokumentu za pomocą następującego makra:

```
\def\objectput#1#2#3{%
\vadjust{
 \vbox to0pt{\kern#2
       \moveright#1\hbox{\rlap{#3}}
 \vss}
 }%
}
```

gdzie: #1 – offset poziomy obiektu, #2 – offset pionowy obiektu, #3 – obiekt oblewany tekstem.

Polecenie to powinno być użyte w dowolnym miejscu tekstu pierwszego wiersza poddawanego działaniu `\parshape`. W naszym przykładzie dotyczy to „lewego" łamu, znakiem * zaznaczyłem miejsce wstawienia `\objectput`. Dla dokładnej lokalizacji grafiki w poziomie i w pionie musimy wstawić odpowiednią wielkość offsetów zaczynając np. od 0mm.

Można też stosować opisane metody w składzie wielołamowym np. z pakietem `\mimulcol`. Będzie trzeba jednak poeksperymentować z miejscem umieszczenia komendy `\parshape` w jednym jak i drugim łamie.

W prezentowanym przykładzie tekst został połączony z grafiką wektorową. Ciekawe efekty daje oblewanie tekstem bitmapy w formacie EPS. Bitmapy czarno-białe nie zawierające półtonów w miejscach białych są przeźroczyste, natomiast bitmapy zawierające półtony lub kolory w każdym miejscu posiadają jakieś wypełnienie. Można to wykorzystać w ten sposób, że część ilustracji stanie się tłem dla naszego tekstu dopasowanego do fragmentu tej ilustracji.

Znawcy AWK-a stwierdzą zapewne, że można nim wygenerować ostateczne, pełne komendy. Nie zawsze będą one jednak uniwersalne. TEX-owiec musi jednak trochę samemu myśleć. CorelDraw! 3.0, na którym się oparłem jest mało dokładny w obliczaniu współrzędnych umieszczanych w plikach EPS, podobno jego nowsze wersje są lepsze. Myślę jednak, że prezentowana tu metoda może może znacznie ułatwić korzystanie z nielubianej komendy `\parshape`.

Niezwykle ostrożnie należy podchodzić do oblewania tekstem obiektów umieszczonych wewnątrz jednego łamu (gdy jednołamowy tekst znajduje się z lewej i prawej strony np. grafiki). Łam w takiej sytuacji w pewnym miejscu się rozdziela na dwa jakby łamy o nielogicznej kolejności czytania. Szczególnie utrudnione jest wtedy czytanie dokumentów o dużej szerokości składu np. 180mm. Trudno odszukać kontynuację lewego wiersza po prawej stronie. Jest to niezgodne z regułami sztuki drukarskiej. Dlatego też programy takie jak np. Quark mają tę możliwość zablokowaną, grafika może być oblewana tekstem tylko z lewej albo tylko z prawej strony. Dlatego też negatywnie oceniam przykład zaprezentowany u dołu strony.

Mimo, że wygląda to bardzo efektowne, to trudno przeczytać tekst. Mamy więc do czynienia ze sztuką dla sztuki, nic więcej. Publikacje składamy jednak po to, by je czytać.

Ps. Dziękuję B. Jackowskiemu za pomoc w przygotowaniu prezentowanych skryptów.

◇ Janusz Marian Nowacki
jnowacki@to.onet.pl

TEX (wym. *tech*) jest systemem profesjonalnego składu drukarskiego. Może służyć do składu książek, czasopism, gazet jak też do edycji i drukowania prostych tekstów, np. listów, ulotek, informatorów, notatek. Wśród systemów służących do podobnych celów wyróżnia się dbałością o jakość wyników. Nie ma sobie równych przy składaniu trudnych tekstów naukowych i technicznych — ze wzorami, tabelami i rysunkami. System TEX, którego twórcą jest wybitny matematyk i informatyk amerykański profesor Donald E. Knuth, został zaprojektowany w taki sposób, że może być przystosowywany do pracy w różnych językach. Przystosowanie takie obejmuje co najmniej rozszerzenie zestawu fontów TEXa o znaki narodowe i wbudowanie reguł automatycznego przenoszenia wyrazów. Często też zachodzi potrzeba dołączenia do standardu specyficznych dla danego języka zwyczajów drukarskich. TEX został w ten sposób przygotowany do pracy w większości języków europejskich. Istnieją również egzotyczne — z naszego punktu widzenia — wersje językowe, m. in. japońska, arabska, hebrajska, perska, grecka, turecka.

Powyższy przykład opracowano w oparciu o makra Alana Hoeniga, opublikowane w *TUGboat*, vol. 8 (1987) No. 2 p. 211. Wykorzystane tu makra można znaleźć w ftp://ftp.gust.org.pl/GUST/bulletin/08/glueball.tex