Włączanie grafik do tekstów w LaTeX 2ε
wer. 1.8

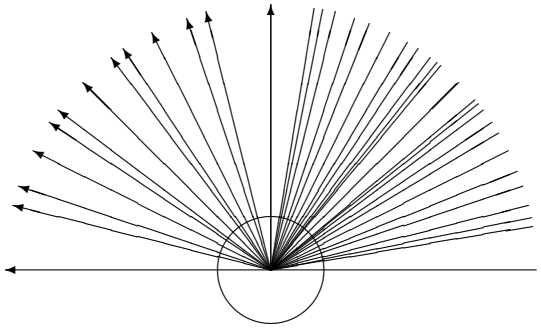
25 lutego 2000 roku

Spis treści

[1] © 1999 by Krzysztof Pszczoła.
[2] © 1999 by Krzysztof Pszczoła.
[3] © 1999 by Krzysztof Pszczoła.
[4] © 1999 by Krzysztof Pszczoła.

1. Wstęp

System LaTeX nie był nigdy pomyślany jako program w którym **tworzy się** grafikę. W czasach kiedy powstawał (La)TeX nie znane były jeszcze powszechnie używane dziś formaty graficzne (PostScript, GIF, JPEG). Dostępny w otoczeniu picture zestaw poleceń (`\line`, `\vector`, `\circle`, ... `\box`, `\oval`) pozwala jedynie na rysowanie linii prostych (o ograniczonych nachyleniach), wektorów (linia prosta[5] z grotem na końcu), okręgów (o średnicach ograniczonych do ok. 40pt), kół (o średnicach z jeszcze węższego zakresu), prostokątów i prostokątów o zaokrąglonych rogach.



Rysunek 1: Nachylenia linii i wektorów dostępne w LaTeX 2ε (oraz okrąg o największej średnicy)

Do rysowania krzywych o nieco bardziej wymyślnych kształtach służy polecenie `\qbezier` składające je z pojedynczych punktów.[6] (Przykład użycia polecenia `\qbezier` zamieszczam na stronie 19.)

Bardziej szczegółowy opis otoczenia picture można znaleźć na przykład w 2. rozdziale książki Rafajłowicza i Myszki [20] albo w rozdziale 10.2 podstawowego podręcznika systemu LaTeX 2ε [5].

Wszystko co jest bardziej skomplikowane musi być przygotowane jakimś innym programem i **włączone** do tekstu jako **zewnętrzny** obiekt albo zlecone do wykonania innym programom (na przykład wszystkie polecenia pakietu PSTricks). Poza zakresem naszych zainteresowań jest **wybór** programu użytego do zrobienia wykresu, szkicu czy schematu, chociaż podamy kilka przykładów takich programów.

Pomysł tej broszurki powstał gdy pomagałem kolegom włączać do ich prac różne rysunki i wykresy przygotowane w różnych okresach czasu i bardzo różnymi narzędziami. Dziś wiem, że były one, bardzo często, przygotowane źle. Może warto poświęcić trochę czasu, żeby ilustracje przygotować porządnie?

Pierwotna wersja tego tekstu dostępna była, również w wersji elektronicznej: `http://www.immt.pwr.wroc.pl/~myszka/grafika/` pod tytułem: „Włączanie grafik w formacie EPS do tekstów w LaTeX 2ε i parę innych uwag". Oprócz informacji dotyczących włączania grafik zawarte tam były również inne treści: związane z polonizacją systemu LaTeX czy instalacją nowej wersji „formatów".

Na zakończenie jedna, ważna uwaga. Na codzień (w domu) korzystam z MiKTeXa. Wszystko o czym tu piszę sprawdziłem i przetestowałem w tym środowisku. Jest ono bardzo podobne do środowiska Web2c dostępnego na płycie TeX Live (którego używam w pracy) czy teTeXa (bardzo często z „dokładnością" do kartotek). Większość tego o czym tu piszę powinna działać również z innymi implementacjami LaTeXa (w tym również z emTeXem).

[5] Zestaw nachyleń wektorów jest jeszcze uboższy!

[6] Tak na marginesie: można również używać polecenia `bezier`, które zachowano w celu zapewnienia zgodności z systemem LaTeX2.09. Patrz również `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/doc/html/usrguide/node31.html`.

Dodatkowo zakładam użycie programu dvips jako narzędzia przetwarzającego pliki `.dvi` do postaci „drukowalnej”.

I na koniec mała uwaga o charakterze formalnym:

This program may be distributed under the conditions of the LaTeX Project Public License, either version 1.1 of this license or (at your option) any later version. The latest version of this license is in `http://www.latex-project.org/lppl.txt` *and version 1.1 or later is part of all distributions of LaTeX version 1999/06/01 or later.*

Tekst przygotowany został z wykorzystaniem systemu MiKTeX. Używałem „eksperymentalnej” polskiej klasy dokumentów *mwart*: `http://www.mimuw.edu.pl/~wolinski/mwcls.html`.

Gdzie on–line znaleźć można tę broszurkę?

Dziwne pytanie: przecież ją czytasz! Jeżeli jednak tego potrzebujesz to sprawdź: `http://www.gust.org.pl/doc.html` oraz `http://www.immt.pwr.wroc.pl/~myszka/grafika/` — być może jest tam wersja ciut bardziej aktualna.

2. Włączanie grafik przygotowanych przez inny program

Jak już napisałem (La)TeX generalnie nie służy do tworzenia rysunków. Zatem w jaki sposób mogą być one włączane do tekstów składanych w tym systemie?

Donald Knuth[7] rozwiązał problem w ten sposób, że wbudował w system TeX polecenie `\special`. Działanie jego jest takie, że wstawia do pliku `.dvi` swój argument. Może on być, na przykład, nazwą pliku zawierającego pewien rysunek.

Zatem ciężar włączania grafik został z programu TeX przerzucony na programy interpretujące plik `.dvi`. Działanie takie ma jedną wadę: dokładna składnia argumentów polecenia `\special` zależy od używanego przez nas programu do oglądania plików `.dvi` czy konwersji ich do postaci drukowalnej.

Pakiety graphics/graphicx w systemie LaTeX 2ε oferują pewną zunifikowaną metodę włączania plików graficznych: polecenie: `\includegraphics`.

Polecenie pozwala na włączenie praktycznie dowolnego obiektu graficznego. Dokument przygotowany dla wielu implementacji (La)TeXa) może wyglądać identycznie. Ale rodzaje włączanych grafik zależą ciągle od używanej implementacji.

I tak, używany przez nas do niedawna emTeX pozwalał na łatwe włączanie rysunków w postaci czarno-białych plików graficznych PCX albo BMP.

Dosyć popularny w środowiskach Windows 9x/NT program MiKTeX oprócz plików BMP (kolorowych!) pozwala na włączanie plików EPS, WMF i EMF.

System teTeX (używany na maszynach Unixowych) obsługuje praktycznie wyłącznie grafiki w postaci EPS.

Inne implementacje dają jeszcze inne możliwości (bardziej szczegółowe informacje na ten temat zawarte są w rozdziale 5). Wydaje się jednak, że pewnego rodzaju standardem staje się przygotowywanie grafik w formacie EPS.

Względna niezależność od implementacji pakietów graphics/graphicx została uzyskana przez rozdzielenie kodu na dwie części: zależnej od implementacji (pliki `.def`) i niezależnej (pliki `.sty`).

Dodatkowe pliki (`color.cfg` i `graphics.cfg`) definiują domyślny tryb (czy model pracy) pakietów.

[7] http://Sunburn.Stanford.EDU/%7Eknuth/

3. Zalety formatu EPS

Format EPS (Encapsulated PostScript) ma szereg zalet:
— Wiele programów pozwala tworzyć grafiki wektorowe[8] i zapisywać je jako pliki EPS.
— Włączane obiekty graficzne mogą być kolorowe.
— Obiekty można łatwo skalować i obracać, przy czym uzyskany efekt będzie bardzo różny w zależności od sposobu przygotowania obiektu graficznego: obiekty rastrowe skalują się (obracają) bardzo źle, obiekty wektorowe skalują się (obracają) bardzo dobrze.
— Każdy bitowy (rastrowy) plik graficzny może być do tego formatu przekształcony.

Wadą używania grafik w postaci EPS jest konieczność wydruku na drukarce PostScriptowej (albo korzystanie ze specjalnych programów, które przekształcają PostScript do postaci zrozumiałej przez drukarkę — najczęściej ghostscript i ghostview/GSview/gv).

Kolejną wadą jest bardzo duża objętość plików EPS. O możliwościach ich kompresji piszę na stronie 31 w opisie programu cep.

4. Tworzenie plików EPS przez inne programy

Pragnę zwrócić uwagę na problem polskich liter w tworzonych plikach PostScriptowych (EPS). Problem jest o tyle trudny, że w „standardowych" (cokolwiek to oznacza) czcionkach PostScriptowych polskich liter nie ma! Dostępne są oczywiście (tak darmowe jak i komercyjne) zestawy polskich czcionek Type1. Nie zawsze jednak używane aplikacje potrafią z nich skorzystać. W każdym przypadku pozostanie pewnym problemem również sposób kodowania (polskich) liter. Unix używa ISO-8859-2, Windows 9x/NT — CP1250 (a w DOSie dodatkowo mamy jeszcze dwa (najpopularniejsze) kodowania: Mazovia i CP852). A jak są kodowane poszczególne czcionki? W jaki sposób (jeżeli jest to niezbędne) kodowanie to zmienić? Problem wymaga dokładniejszego opisania przez fachowców.

Jeżeli miałbym coś podpowiadać to widzę trzy możliwości:
1. Zainstalowanie jakichś polskich fontów Type1 i zmuszanie używanej aplikacji do korzystania z nich. Udaje się to, na przykład, z programem gnuplot w środowisku Windows 9x.
2. Pakiet ogonkify Juliusza Chroboczka który polonizuje różne dziwne pliki PostScriptowe w których użyto polskich liter nie dbając o ich brak w samym foncie: `http://www.dcs.ed.ac.uk/~jec/programs/ogonkify/`.
3. Pakiet psfrag przedstawiony w rozdziale 10.

(Tak na marginesie: różnym aspektom polonizacji komputerowego środowiska pracy poświęcona jest Polska Strona Ogonkowa[9].)

4.1. Programy systemu Windows

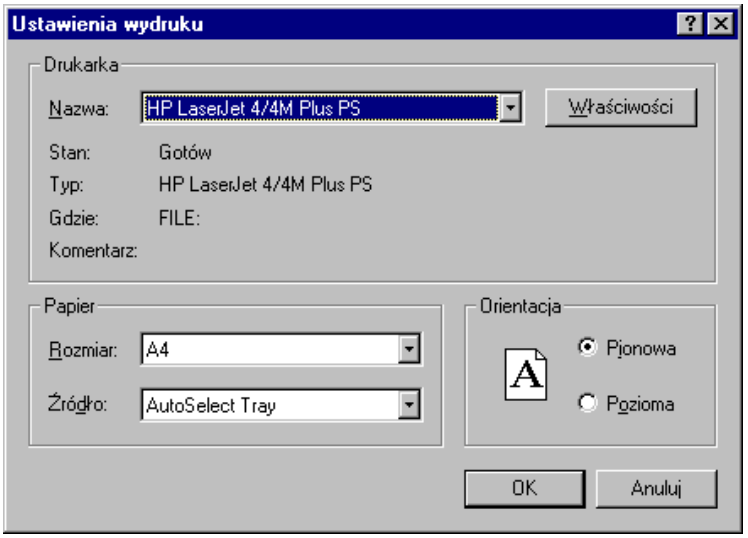
Aby w systemie Windows 9x/NT najpewniej (choć nigdy ze 100% pewnością) tworzyć pliki EPS, najwygodniej jest zainstalować najnowszy sterownik drukarki PostScriptowej z serwera Adobe.[10]

Opis w jaki sposób doprowadzić do utworzenia „dobrego" pliku EPS przedstawiamy poniżej na przykładzie programu SigmaPlot w systemie Windows NT.

[8] Wektorowe, to znaczy takie, które złożone są ze stosunkowo prostych obiektów (proste, łuki okręgów, krzywe również wyższego stopnia, wypełnienia) zadanych parametrycznie: za pomocą współrzędnych (początku, końca, środka,...) i pewnych dodatkowych parametrów (promień, kąt, długość).

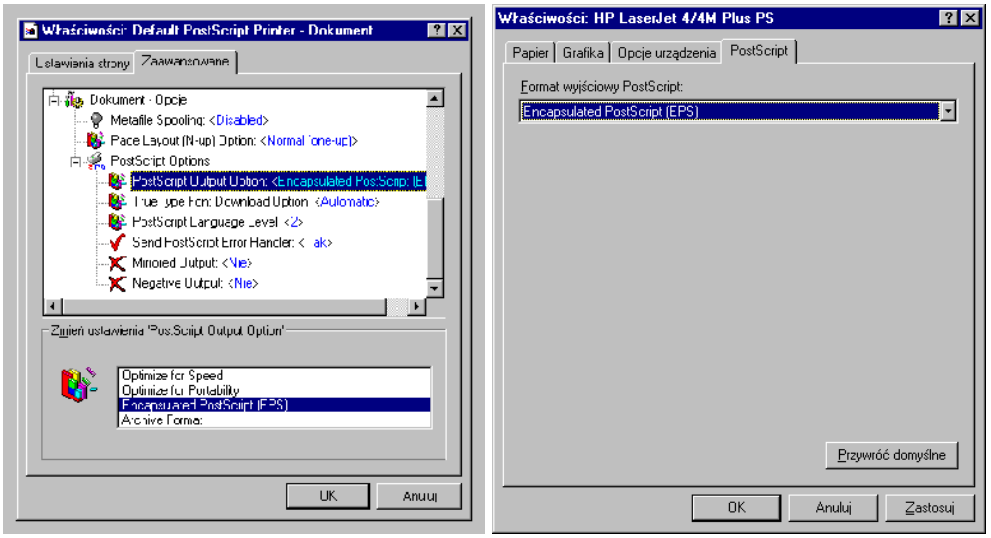
[9] http://www.agh.edu.pl/ogonki/

[10] `http://www.adobe.com/`



Rysunek 2: Wybór drukarki w systemie Windows NT/9x

1. Rysunek przygotowywujemy w programie SigmaPlot tak jak zwykle. Na „stronie” powinien być umieszczony tylko jeden rysunek bez żadnych dodatkowych napisów (numer strony, nagłówki...).
2. Na rysunku można nanosić teksty, korzystać z kolorów i wszystkich (chyba) możliwości, które daje SigmaPLot. Nie znalazłem jeszcze sposobu na użycie polskich liter.
3. Gotowy obrazek musimy teraz zapisać na dysk we właściwym formacie. W tym celu wykonujemy następujące czynności: File|Print... Jako drukarkę wybieramy „Default PostScript Printer”. Stawiamy „ptaszka” w okienku „Drukuj do pliku”.
   Naciskamy klawisz „Właściwości” i wybieramy z „Dokument — Opcje” znajdujące się na samym dole „PostScript Options” naciskając maleńki plusik. Musimy ustawić: „Postscript Output Option: <Encapsulated PostScript (EPS)>
   W przypadku Windows 9x postępowanie jest analogiczne: po naciśnięciu klawisza „Właściwości” należy wybrać zakładkę PostScript i w okienku wybrać „Encapsulated PostScript (EPS)”.
   Naciskamy klawisz „OK” i jeszcze raz „OK” żeby wydrukować. System powinien zapytać nas o nazwę pliku. Podajemy cokolwiek, na przykład `rysunek.ps`.
4. Otrzymany plik PS otwieramy programem GSview. Ghostscript posłuży nam do ostatecznego przekształcenia pliku do właściwej postaci. W tym celu wybieramy File|PStoEPS i w otwartym okienku sprawdzamy czy ptaszek jest przy „Automatically calculate Bounding Box” i naciskamy „Yes”. Program poprosi o podanie nazwy pliku, podajemy `rysunek.eps`.
5. Otrzymany plik .eps można już włączać do tekstów pisanych w LaTeX 2ε poleceniem:
   `\includegraphics{rysunek}`
6. Jeżeli okaże się, że pojawiają się jakieś problemy z kolorami (wbrew oczekiwaniom rysunek jest szary zamiast kolorowy) mamy problem. Można z nim sobie radzić tworząc specjalny plik opisu drukarki (`.PPD`). Zakładam przy tym, że zainstalowany został sterownik Adobe dla drukarek PostScriptowych (patrz str. 32). Jest on tak skonstruowany że pozwala dodawać kolejne modele drukarek tworząc plik parametryczny (`.PPD`).
   W kartotece w której zainstalowaliśmy program instalacyjny znajdować powinien się plik `DEFPRTR2.PPD`. Możemy potraktować go jako wzorzec dla „nowej” drukarki. Kopiujemy ten



Rysunek 3: Ustawienie właściwości przy drukowaniu do pliku

plik pod inna nazwą (`test.ppd`, na przykład) i otwieramy w edytorze tekstowym. Bardzo ostrożnie będziemy musieli zmodyfikować poniższe linie:[11]

```
...
*PCFileName: "DEFPRTR2.PPD"
...
*Product: "(Default PostScript Printer)"
...
*ModelName: "Default PostScript Printer"
*NickName: "Default PostScript Printer v(2010)"
...
*ColorDevice: False
*DefaultColorSpace: Gray
...
```

do postaci:

```
...
*PCFileName: "TEST.PPD"
...
*Product: "(Default PostScript Printer kolor)"
...
*ModelName: "Default PostScript Printer kolor"
*NickName: "Default PostScript Printer kolor v(2010)"
...
*ColorDevice: True
*DefaultColorSpace: RGB
...
```

[11] Jeżeli zdecydujemy się zainstalować program wmf2eps, o którym piszę nieco dalej, możemy korzystać ze sterownika, który instalowany jest z tym programem.

Później instalujemy nową drukarkę. Powinno pomóc.[12]

Niezbyt złożoną procedurę „drukowania do pliku” można nieco uprościć jeżeli program którego używamy pozwala na zapisaniu wyniku pracy w postaci pliku `.WMF`.[13]

W takiej sytuacji użyć możemy programu wmf2eps. Pozwala on na konwersję plików WMF/EMF do postaci EPS. Co więcej, nawet jeżeli używany przez nas program nie pozwala na zapis w postaci WMF program można wykorzystać również do konwersji wektorowej informacji znajdującej się w „schowku” (*clipboard*).[14]

Tak więc wystarczy zaznaczyć wykres utworzony na przykład w programie Excel, skopiować go do schowka aby po uruchomieniu programu wmf2eps „wkleić” (*paste*) do pola roboczego programu a później skonwertować do postaci EPS.

Poniżej podaję (niepełną) listę aplikacji Windowsowych generujących poprawne (na tyle, na ile byłem to w stanie osobiście sprawdzić) pliki EPS:

— gnuplot (gnu) — patrz również str. 9,
— tkpaint (gnu)
— SigmaPlot,
— Mathcad 8 (za wyjątkiem wykresów 3D!),
— Excel,
— PowerPoint,
— PageDraw (freeware — nosi obecnie nazwę MayuraDraw 2.04),
— MayuraDraw (shareware).
— CorelDraw! posiada możliwość eksportu grafiki w formacie EPS. Należy wybrać menu File -> Export do pliku EPS, po czym włączyć opcję: „Eksportuj tekst jako krzywe” (*Export text as curves*) i (**koniecznie**) wyłączyć opcję „Dołącz nagłówek” (*Add header*). Postępowanie takie można traktować jako „regułę kciuka” także dla eksportu w formacie EPS z innych programów graficznych działających w Windows.

Pomijam bardzo wiele komercyjnych i zazwyczaj bardzo drogich aplikacji pracujących w środowisku Windows — niestety, nie zawsze mam do nich dostęp :-)

gnuplot, tkpaint i MayuraDraw posiadają możliwość zapisu (lub eksportu) tworzonych grafik bezpośrednio w postaci plików EPS.

Do konwersji grafik bitmapowych do postaci EPS można używać na przykład programu ImageMagick (free), lub Paint Shop Pro (shareware) lub jakiegoś innego. W każdym przypadku, należy pamiętać o **wyłączeniu** właściwości, która nazywa się „Preview”.

Również oprogramowanie dostarczane wraz ze skanerami ma możliwość zapisu plików w postaci EPS.

Wiele cennych uwag na temat technik przetwarzania i „ulepszania” obrazów stosowanych podczas skanowania znaleźć można w `http://www.scantips.com/`.

4.2. Programy w systemie Unix

— Mathematica,

W środowisku tekstowym postępujemy tak: gdy już mamy przygotowany obrazek (niech nazywa się on wykres):

```
In(1):=wykres=Plot[Sin[x], {x, 1, 10}];
```

[12] Powyższa procedura nie ma żadnego uzasadnienia teoretycznego, ale w pewnych sytuacjach praktycznych - działa. Wszystkie zmiany każdy dokonuje na własną odpowiedzialność. Warto też zapoznać się z *End User License Agreement*.

[13] WMF: *Windows Meta–File* specyficzny dla systemu Windows sposób zapisu grafik. Podobnie jak EPS pozwala na zapis grafik wektorowych (i rastrowych oczywiście). Obok niego istnieje również format EMF: *Enhanced Metafile*.

[14] O ile tylko program potrwfi przekazać informacje w formacie wektorowym do schowka!

zapisujemy go do pliku `wykres.eps` w formacie EPS poleceniem:

```
In(2):=Display["!psfix -epsf > wykres.eps", wykres]
```

— Matlab,

Aby wykres zapisać w postaci pliku EPS należy wydać polecenie `print` o następującej postaci:
`print -d`⟨*devicetype*⟩ ⟨*filename*⟩
Jako ⟨*devicetype*⟩ podać możemy: `eps` aby zapisać rysunek jako czarnobiały plik EPS lub `epsc` — rysunek kolorowy; można użyć również `eps2` lub `epsc2` (PostScript Level 2).
⟨*filename*⟩ oznacza nazwę pliku w którym zostanie zapisany rysunek.

— gnuplot,

Typ „terminala” musimy zdefiniować jako:
`set terminal postscript eps` ⟨*color*⟩ ⟨*dashed*⟩ `"`⟨*fontname*⟩`"` ⟨*fontsize*⟩
lub
`set terminal mp` ⟨*color*⟩ ⟨*dashed*⟩ `[`notex`] [mag` ⟨*magsize*⟩`] "`⟨*fontname*⟩`"` ⟨*fontsize*⟩
gdzie:

⟨***color***⟩ przyjmuje wartości `color` lub `monochrome` (wartość domyślna — `monochrome`),

⟨***dashed***⟩ przyjmuje wartości `solid` lub `dashed` (wartość domyślna `dashed`); parametr decyduje o typie linii użytych do rysowania wykresów: ciągłe lub przerywane,

⟨***fontname***⟩ nazwa fontu PS używanego do opisów (wartość domyślna Helvetica w przypadku terminala
`postscript` lub cmr10 w przypadku terminala `mp` — chyba, że wybrano opcję `notex`, wówczas: pcrr8r),

⟨***fontsize***⟩ wielkość fontu (domyślnie 14pt w przypadku terminala `postscript` lub 10pt w przypadku terminala `mp`),

⟨***notex***⟩ przy konstruowaniu wykresu nie będą nigdzie używane konstrukcje TEXowe (pozwala na używanie znaków zastrzeżonych, na przykład % czy $; w przeciwnym razie znaki takie muszą być poprzedzone pojedynczym lub podwójnym[15] znakiem *backslash*, a w pewnych przypadkach trzeba stosować bardziej zaawansowane techniki znane z LATEXa),

`mag` ⟨***magsize***⟩ definiuje współczynnik „powiększenia” (pozwala to, na przykład, na powiększenie symboli matematycznych),

Aby wykres zapisać do pliku używamy polecenie `set output "`⟨*filename*⟩`"`. Każdy wykres powinien być zapisywany do osobnego pliku!

Tak na marginesie — pragnę zwrócić uwagę na pakiet egplot pozwalający na włączanie w tekst źródłowy (w LATEXu) poleceń, które zostaną zapisane do pliku, a po przetworzeniu pliku przez gnuplot, w kolejnym przebiegu włączone jako grafiki EPS. Narzędzie pozwala na wygodne zintegrowanie kodu programów generujących rysunki z tekstem który się do nich odwołuje.

Istnieje również możliwość tworzenia wykresów w innych formatach „zrozumiałych” dla systemu LATEX. Jako terminal wybrać można:

— `latex` (rysunek tworzony jest za pomocą elementarnych poleceń języka LATEX: circle, rule, line, vector); uwaga: pliki wynikowe są bardzo obszerne!
— `pslatex` (ew. `pstex`) wykresy tworzone są z udziałem specjalnych poleceń języka PostScript włączanych do wynikowego pliku za pomocą poleceń `\special`;
— `eepic` — wymagają użycia pakietu eepic i specjalnego sterownika drukarki;
— `tpic` wymaga sterowników rozumiejących polecenia tpic;
— `pstricks` wymaga użycia pakietu pstricks (por. rozdział 11.3);
— `texdraw` do wykorzystania z pakietem texdraw;
— `mf` — przygotowywuje program który powinien być później przekształcony do pliku `.pk` — fontu. Niedogodnością wykorzystywania programu mf do przygotowywania rysunków

[15] Gdy teksty zamykane są znakami podwójnego cudzysłowu.

jest konieczność generowania osobnego obrazka w rozdzielczości właściwej dla każdego używanego urządzenia drukującego;
— `mp` metapost (począwszy od wersji 3.7.1).
Wydaje się, że rysunki w postaci EPS będą rozwiązaniem najwygodniejszym, choć `metapost` pozwoli, być może, zapanować nad polskimi literami...
— tkpaint.
Bardzo sympatyczny program do przygotowywania rysunków wykorzystujący zestaw narzędzi Tcl/Tk (wymaga ich zainstalowania). Pod wielu względami program jest podobny do programu Xfig. Sam program został opracowany w środowisku Windows (tam dostępny jest również jako samodzielny program, nie wymagający Tcl/Tk) i „przeniesiony" do pracy w środowisku Unix.
— SDRC I–DEAS,
Zwracam uwagę, że mimo, iż oprogramowanie (w wersji 4) potrafi wyprodukować kolorowe pliki EPS, są z nimi różne problemy:
— pliki są bardzo duże w przypadku obrazów „cieniowanych",
— PostScript jest trochę niestandardowy — poszczególne wiersze zakończone są dziwaczną kombinacją znaków co przeszkadza niektórym programom,
— może zachodzić konieczność „ręcznego" wyspecyfikowania wymiarów rysunku (*Bounding Box*).
— Xfig. Program pozwala na eksport rysunków do postaci EPS.

Do konwersji grafik bitmapowych do postaci EPS można używać programu convert z pakietu ImageMagick lub xv.

Do „poprawiania" plików EPS mających źle wyznaczony BoundingBox (a również do konwersji[16] plików typu PS do EPS) można użyć programu ps2epsi lub psfixbb.

4.3. Analizator HP 35655A oraz konwersja plików HPGL

(Poniższe uwagi mogą dotyczyć innych urządzeń HP z „wbudowaną inteligencją" i dużej części plików typu HPGL.[17]) Jeżeli zachodzi potrzeba załączenia „zrzutów" ekranowych analizatora, najlepiej zapisać je na dyskietce tak jak pliki przekazywane na ploter.[18]

Postępowanie prowadzące do zapisania na dyskietce pliku we właściwym formacie jest następujące:
1. Na panelu czołowym naciskamy klawisz „Print/Plot".
2. Naciskamy klawisz koło monitora opisany „More Setup".
3. Ustawiamy „Device is PLOT".
4. Ustawiamy „Output to File".
5. Naciskamy klawisz „Return".
6. Nazwę pliku możemy ustalić po naciśnięciu klawisza „Output Filename" (standardowo analizator nadaje kolejnym plikom nazwy: PLOT1, PLOT2,...)
7. Naciskamy „Start Plot/Print"

Plik taki (w języku HPGL) może być przekształcony do postaci EPS za pomocą programu hp2xx (dostępne są wersje pracujące w środowiskach UNIX lub DOS).

Polecenie ma następującą postać:
`hp2xx -m eps -f` ⟨*filename*⟩ ⟨*file*⟩

[16] Zwracam uwagę, że poprawny plik EPS nie może zawierać wielu poleceń języka PostScript. Zatem nie każdy plik PS może być przekształcony do EPS.

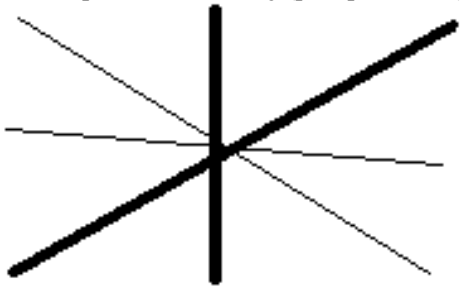
[17] HPGL to specjalny język wykorzystywany do programowania ploterów.

[18] Program hp2xx może być wykorzystany również do konwersji innych plików zapisanych w standardzie HPGL do innych postaci.

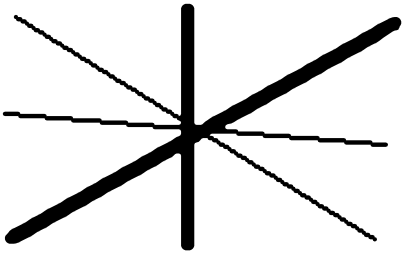
gdzie: ⟨*filename*⟩ to nazwa pliku wyjściowego (w przypadku pominięcia zostanie przyjęta jako ⟨*file*⟩`.eps`) a ⟨*file*⟩ nazwa pliku wejściowego.

4.4. Konwersja map bitowych do postaci skalowalnej

Jeżeli chcemy rysunek/obrazek w postaci bitowej przekształcić do postaci EPS możemy wykorzystać jeden z kilku dostępnych programów:
— convert z pakietu ImageMagick (środowisko Unix lub Windows 9x/NT),
— xv w środowisku Unix,
— gws w środowisku DOS lub Windows,
— Paint Shop Pro w środowisku Windows.
(nie jest to w żadnym wypadku pełna lista programów). W środowisku Windows możemy użyć właściwie dowolnego programu, który potrafi rysunek wydrukować — trzeba jedynie zastosować metodę opisaną wcześniej (por. punkt 4.1).



Rysunek 4: Bitmapowy (Rastrowy) EPS uzyskany za pomocą programu Paint Shop Pro

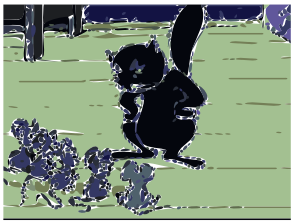


Rysunek 5: Wektorowy (Skalowalny) EPS uzyskany za pomocą programu kvec

Powyższe programy zapisują mapę bitową z wykorzystaniem poleceń języka PostScript ale nie zmieniają charakteru grafiki — ciągle składa się ona z pojedynczych punktów. W przypadku konieczności skalowania (a zwłaszcza obrotów lub przekształceń) — możemy mieć do czynienia ze wszystkimi efektami skalowania map bitowych.

Kolejną wadą tak uzyskanych plików EPS jest ich ogromny rozmiar. Pewnym rozwiązaniem może być użycie **stratnej** kompresji obrazka do postaci JPEG a następnie użycie programu jpeg2ps który wykorzystuje fakt, że PostScript Level 2 potrafi sobie radzić z odpowiednio zakodowanym plikiem JPEG albo programu tiff2ps realizującego podobną ideę w przypadku plików TIFF. Gdy zależy nam na **bezstratnej** kompresji lepszym rozwiązaniem jest użycie programu cep [24] do kompresji rastrowych EPSów (czytaj również na stronie 31).

Jeżeli jednak zechcemy przekształcić grafikę rastrową do postaci wektorowej — napotkamy spore kłopoty.



Znalazłem program o nazwie kvec (Shareware — patrz również na stronie 32) który stara się dokonać tego dzieła, jednak zadowalające efekty uzyskamy jedynie w ograniczonej liczbie przypadków... Najlepsze efekty uzyskuje się w przypadku dwubarwnych symboli o niezbyt skomplikowanym kształcie.

Zamieszczam przykłady pliku przekształconego do postaci „rastrowego” EPS (rysunek 4) a w drugim do postaci „wektorowego” EPS (rysunek 5). W żadnym wypadku efekt nie jest idealny — niektóre „linie” ciągle nie mają charakteru linii a raczej „schodów”; efekty skalowania widać dopiero przy bardzo dużym pomniejszeniu, ale... I wreszcie przykład zdjęcia przekształconego programem kvec. Kolejny przykład na stronie 17.

Podobno[19] najlepszym narzędziem do przekształcania grafiki bitmapowej w wektorową jest Adobe Streamline; trudno jednak polecać komercyjny program kosztujący ok. 300 USD do zastosowań nieprofesjonalnych.

5. Pakiet graphicx[20]

Do manipulowania obiektami graficznymi służą polecenia zdefiniowane w pakietach graphicx i graphics. Pakiety te różnią się składnią poleceń `\includegraphics` i `\rotatebox`. Ponieważ polecenia pakietu graphicx dają więcej możliwości i mają bardziej elegancką składnię — dalej będę się zajmował tylko tym pakietem.

Wywołując pakiet graphicx możemy zadeklarować, do jakiego DVI-procesora mają być dostosowane generowane polecenia `\special`; domyślnie jest to dvips. Poniżej podajemy listę obsługiwanych DVI-procesorów oraz ich możliwości (ograniczenia).

autor	nazwa	możliwości
T. Rokicki	dvips	wszystko
N. Beebe	dvialw	tylko dołączanie plików i skalowanie
S. Lesenko	dvipdf	wszystko
Arbortext	dvilaser	tylko dołączanie plików i skalowanie
Y& Y	dvipsone	wszystko
J. Clark	dvitops	wszystko oprócz zagnieżdżonych obrotów
H. Sendoukas	dviwin	tylko dołączanie plików
Y&Y	dviwindo	wszystko
	dvi2ps	tylko dołączanie plików i skalowanie
E. Mattes	emtex	tylko dołączanie plików bez skalowania
B.H. Kelly	ln	dołączanie plików dla drukarki DEC LN03
A. Trevorrow	oztex	dołączanie plików, kolor, obroty
PCTeX	pctexps	dołączanie plików, kolor, obroty
PCTeX	pctexwin	dołączanie plików, kolor, obroty
PCTeX	pctex32	wszystko
PCTeX	pctexhp	tylko dołączanie plików
A. Trevorrow	psprint	tylko dołączanie plików
Arbortext	pubps	dołączanie plików i obroty
Kinch	truetex	dołączanie plików i elementy koloru
Kinch	tcidvi	TrueTeX z dodatkową obsługą Scientific Worda
Blue Sky	textures	wszystkie funkcje dla Textures

Dodatkowo pakiet graphicx może być wywoływany z następującymi parametrami:

hiresbb high resolution bounding box; rezerwuje na rysunek miejsce o wymiarach podanych w pliku EPS jako %%HiResBoundingBox;
final odwrotnie niż draft;
hiderotate nie wyświetla obracanych elementów;
hidescale nie wyświetla skalowanych elementów.

6. Polecenie `includegraphics`

Do wstawiania plików graficznych w tekst tworzonego dokumentu służy polecenie `\includegraphics`. Jest ono tak skonstruowane, że:

[19] Uwaga zasugerowana przez Krzysztofa Pszczołę :-)
[20]

— może być umieszczone praktycznie w dowolnym miejscu tekstu,
— automatycznie „rezerwuje” odpowiednią ilość miejsca na wstawianą ilustrację.

Taka konstrukcja wymaga jednak posiadania pewnych informacji na temat rozmiarów wstawianego obiektu. Tak się dobrze składa, że pliki graficzne praktycznie wszystkich typów zawierają informację o rozmiarach pliku. Problem polega jednak na tym, że większość plików graficznych ma postać binarną, która jest dosyć trudna do obrabiania przez LaTeXa a zatem bezużyteczna.

Format EPS jest generalnie znakowy i tak skonstruowany, że nagłówek pliku zawiera informację na temat wielkości generowanej grafiki. Ma ona następującą postać:

```
%%BoundingBox: 50 50 410 302
```

dwie pierwsze liczby oznaczają *x*–ową i *y*–ową współrzędną lewego dolnego rogu grafiki, a dwie następne — prawego górnego rogu. Wszystkie wymiary podawane są w punktach używanych przez PostScript. Wielkość jednego punktu (`bp`) to $\frac{1}{72}$ cala czyli około 0.35278 mm. Punkt używany przez TeXa (`pt`) ma nieco inny wymiar: $\frac{1}{72.27}$ cala czyli 0.35146 mm.

W pewnych przypadkach, to znaczy wtedy gdy plik EPS zawiera tak zwany *preview*[21] czyli niskiej rozdzielczości rastrowy i binarny obraz[22] swojej „zawartości” — dotarcie do informacji o rozmiarach pliku może być utrudnione albo niemożliwe. W takich przypadkach (albo wtedy gdy włączamy grafiki typu rastrowego: pliki BMP, PCX, MSP) informację tę musimy dostarczyć — patrz informacje o opcji `bb` na stronie 15.

Aby użyć polecenia `\includegraphics` w tekście dokumentu, w jego preambule (to znaczy **przed** `\begin{document}` powinno znaleźć się polecenie:

```
\usepackage{graphicx}
```

Składnia polecenia `\includegraphics` jest następująca:

`\includegraphics[`⟨*parametry_dodatkowe*⟩`]{`⟨*nazwa_pliku_graficznego*⟩`}`

⟨*parametry_dodatkowe*⟩ mogą być następujące (przy czym nie jest to pełen wykaz; odsyłam do [21] i [2]):

`width` określa szerokość obiektu,

`height` określa wysokość obiektu (normalnie obiekty graficzne skalowane są tak, aby zachować proporcje oryginału pomiędzy wysokością a szerokością; wówczas wystarczy podać tylko jeden z powyższych parametrów),

`totalheight` określa wysokość pudełka w którym będzie umieszczony obiekt; istotne w przypadku dokonywania obrotów,

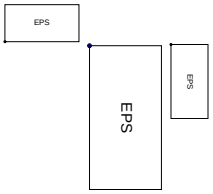
`keepaspectratio` w przypadku gdy podane są oba powyższe parametry, dodatkowe użycie `keepaspectratio` powoduje, że wstawiony obiekt będzie przeskalowany w taki sposób aby nie przekroczyć żadnego z zadanych rozmiarów i zachować proporcje oryginału,

`angle` określa kąt (w stopniach) obrotu obiektu, liczby dodatnie oznaczają obrót w kierunku przeciwnym do ruchu wskazówek zegara; trzeba pamiętać, że w przypadku dokonywania obrotów wielkość obracanego obiektu zależy od kolejności podawania parametrów `width` lub `height` i `angle`:

```
\includegraphics[width=1cm,angle=0]{eps}
\includegraphics[angle=-90,width=1cm]{eps}
\includegraphics[width=1cm,angle=-90]{eps}
```

[21] Tak zupełnie na marginesie: jeżeli rysunek EPS **nie zawiera** *preview* a koniecznie musimy go dołączyć –polecam pakiet **epstool** `http://sunsite.icm.edu.pl/pub/CTAN/support/ghostscript/rjl/`.

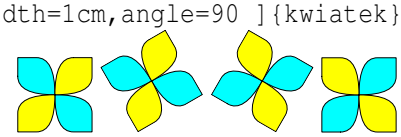
[22] Obraz ten jest wykorzystywane przez programy typu WYSIWYG do przedstawienia operatorowi przybliżonego wyglądu strony. Co gorsze, w pewnych przypadkach, *preview* wykorzystany jest również do drukowania na drukarkach innych niż PostScriptowe!



`scale` parametr mówi w jakich proporcjach ma być przeskalowany cały obiekt,

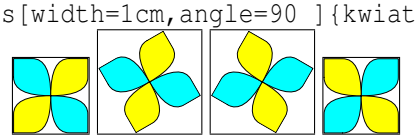
`origin` parametr określa współrzędne punktu, wokół którego obracany jest obiekt, normalnie jest to punkt wstawienia obiektu czyli jego lewy, dolny róg. Poniższy przykład pokazuje jakie jest znaczenie tego parametru.

```
\includegraphics[width=1cm,angle=0  ]{kwiatek}
\includegraphics[width=1cm,angle=30 ]{kwiatek}
\includegraphics[width=1cm,angle=60 ]{kwiatek}
\includegraphics[width=1cm,angle=90 ]{kwiatek}
```



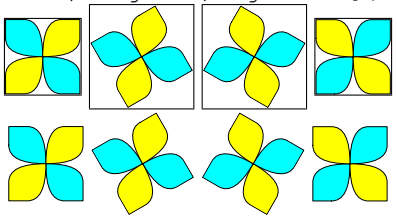
Jak są poukładane „kwiatki” będzie lepiej widać gdy dodamy BoundingBoxy rysunków:

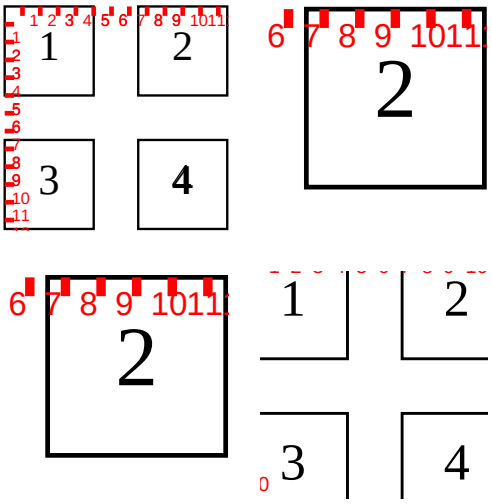
```
\fbox{\includegraphics[width=1cm,angle=0  ]{kwiatek}}
\fbox{\includegraphics[width=1cm,angle=30 ]{kwiatek}}
\fbox{\includegraphics[width=1cm,angle=60 ]{kwiatek}}
\fbox{\includegraphics[width=1cm,angle=90 ]{kwiatek}}
```



Jeżeli jednak dodamy parametr mówiący, że obiekty mają być obracane wokół swojego środka obraz się zmieni:

```
\fbox{\includegraphics[width=1cm,origin=c,angle=0  ]{kwiatek}}
\fbox{\includegraphics[width=1cm,origin=c,angle=30 ]{kwiatek}}
\fbox{\includegraphics[width=1cm,origin=c,angle=60 ]{kwiatek}}
\fbox{\includegraphics[width=1cm,origin=c,angle=90 ]{kwiatek}}

\includegraphics[width=1cm,origin=c,angle=0  ]{kwiatek}
\includegraphics[width=1cm,origin=c,angle=30 ]{kwiatek}
\includegraphics[width=1cm,origin=c,angle=60 ]{kwiatek}
\includegraphics[width=1cm,origin=c,angle=90 ]{kwiatek}
```






```
\includegraphics[width=3cm]{crop}
\includegraphics[width=3cm,
 bb=306 396 576 666,clip]{crop}
\includegraphics[width=3cm,
 viewport=270 270 540 540,clip]{crop}
\includegraphics[width=3cm,
 trim=50 50 50 50,clip]{crop}
```

Zwracam uwagę na parametr `clip`. Jest on niezbędny, gdy włączany obiekt „wystaje” poza BoundingBox lub obszar, który chcemy pokazać. W pliku `crop.eps` jest `%%BoundingBox: 36 126 576 666`

Rysunek 6: Przykład użycia różnych parametrów polecenia `includegraphics`

`clip` parametr żądający aby wszystko to co wykracza poza wymiary obiektu było obcinane,[23]

`bb` określa wymiary rysunku (*Bounding Box*); należy je podać jako cztery liczby oddzielone odstępami będące współrzędnymi lewego, dolnego i prawego górnego rogu obszaru. Liczby podawane są w jednostkach zwanych `bp` („duże punkty”, 1/72"). Parametr niezbędny gdy plik EPS jest pozbawiony tych informacji[24] lub gdy, LaTeX nie może tych danych z pliku EPS odczytać (na przykład z powodu zbyt długich wierszy albo w sytuacji gdy plik jest skompresowany.). Nawet mając plik EPS zawierający prawidłowy BoundingBox, manipulując wartością parametru `bb` można „wyciąć” z rysunku tylko ten detal, który jest nam potrzebny (patrz rysunek 6 b). Wydaje się jednak, że lepiej w tym celu użyć innych parametrów, opisanych poniżej.

`viewport` Parametr pozwala na wybranie z większego rysunku tylko pewnego jego fragmentu. Wymiary podaje się jako cztery liczby będące współrzędnymi lewego, dolnego i prawego górnego rogu obszaru. Współrzędne podawane są względem współrzędnej lewego dolnego rogu BoundingBox. Przykład zastosowania tego parametru podaje rysunek 6 c.

`trim` Jest to alternatywna metoda określania który fragment obiektu ma być drukowany. Wartością parametru są cztery liczby (w jednostkach `bp`) mówiące ile z rysunku należy odciąć z lewej strony, z dołu, góry i z prawej strony odpowiednio. Przykład na rysunku 6 d.

`draft` powoduje wstawienie zamiast obiektu graficznego tylko nazwy pliku i ramki określającej miejsce zajmowane przez obiekt, bardzo wygodne gdy ciągle pracujemy nad tekstem, a wstawiane grafiki są bardzo skomplikowane; odwrotnością `draft` jest `final`; parametr `draft` może być użyty w wierszu `\usepackage`:

```
\usepackage[draft]{graphicx}
```

i wówczas dotyczy wszystkich włączanych grafik, lub w linii `\documentclass`:

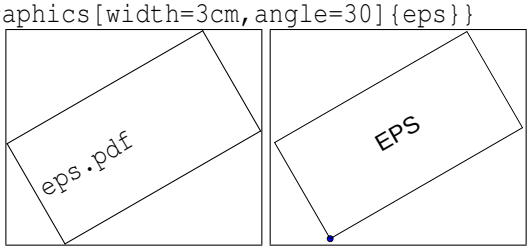
```
\documentclass[draft]{}
```

[23] Normalnie ten parametr nie jest potrzebny, jednak w przypadku pewnych obiektów graficznych sam rysunek jest nieco większy niż **zadeklarowana** jego wielkość. W takich sytuacjach pominięcie parametru `clip` powoduje, że grafika będzie wykraczała poza przydzielone jej miejsce.

[24] Ale wówczas zazwyczaj nie jest prawidłowym plikiem EPS!

i wówczas jest to parametr obowiązujący globalnie (i modyfikujący zachowanie również innych elementów systemu LaTeX 2ε).

```
\fbox{\includegraphics[draft,width=3cm,angle=30]{eps}}
\fbox{\includegraphics[width=3cm,angle=30]{eps}}
```



Jeżeli po parametrze ma być podana jakaś wartość (`width`, `height`, `angle`...) wówczas między parametrem a wartością powinien być znak = (równość):

```
\includegraphics[width=\textwidth]{obrazek}
```

w tym przypadku grafika zajmie całą szerokość strony: `\textwidth`.

Zamiast `\usepackage{graphicx}` można również użyć `\usepackage{graphics}`. Oba pakiety udostępniają zbliżony zestaw możliwości. Zaletą pakietu graphicx jest możliwość podawania wszystkich parametrów w postaci *słowo_kluczowe=wartość*.

7. Włączanie grafik rastrowych

Temu tematowi (zupełnie poza zakresem naszych zainteresowań) poświęcimy jednak kilka zdań. Czasami może zdarzyć się że mamy plik graficzny w postaci rastrowej i musimy włączyć go do przygotowywanego tekstu. W takim wypadku możemy:

— Przekształcić plik do postaci EPS i postępować dalej jak to opisaliśmy powyżej.

— Dokonać przekształcenia pliku do postaci „fontu" (to znaczy plików `pk` i `tfm`) i postąpić z tymi plikami jak z każdym fontem. Wadą takiego rozwiązania jest to, że konwersji musimy dokonywać osobno dla każdej rozdzielczości urządzenia drukującego.

— Jeżeli używamy programu dvips do przekształcania plików `dvi` możemy skorzystać[25] z obecnej ciągle w tym programie możliwości włączania czarno-białych map bitowych poleceniem `\special{em:graph` ⟨*nazwa_pliku*⟩`}`.

Kilka uwag na temat powyższych metod.

Konwersja do postaci EPS ma tę wadę, że powstające pliki są duże. (Stosunkowo małe pliki generuje program convert z pakietu ImageMagic; można również dokonywać kompresji za pomocą programu cep.) Alternatywnym rozwiązaniem, jest konwersja "w locie" z wykorzystaniem jakiegoś programu pomocniczego. Pakiet graphicx pozwala czynność tę zautomatyzować.

Załóżmy, że mamy dużo rysunków w postaci plików PNG i chcemy włączyć je do tekstu przygotowywanego LaTeXem. W preambule dokumentu umieścimy dwa polecenia [21]:

```
\DeclareGraphicsExtensions{.png,.eps}
\DeclareGraphicsRule{.png}{eps}{.png.bb}{`convert #1 eps:-}
```

Pierwsze z nich deklaruje, że LaTeX, jeżeli nie określimy rozszerzenia nazwy pliku, będzie poszukiwał plików graficznych o rozszerzeniach `.png` i `.eps`. Drugie określa regułę konwersji z formatu PNG do postaci EPS (użyjemy programu convert) oraz miejsce gdzie zapisane zostaną wymiary pliku (będzie to plik o rozszerzeniu `.png.bb`).

Ilustracje włączamy w sposób „klasyczny", wydając polecenie:

[25] Wymaga to aby program dvips został skompilowany z opcją `-Demtex`.

```
\includegraphics[ ]{face1}
```

Przygotować tylko musimy plik o nazwie `face1.png.bb` w tym celu musimy znać wymiary grafiki. Możemy w tym celu albo użyć programu convert: utworzyć plik EPS poleceniem `convert face1.png face1.eps` i obejrzeć jakimś dobrym edytorem tekstowym (powstały plik może być dosyć duży!) jakie parametry zostały wpisane w powstałym pliku. Początek pliku wyglądał będzie tak:

```
%!PS-Adobe-3.0 EPSF-3.0
%%Creator: (ImageMagick)
%%Title: (face1.eps)
%%CreationDate: (Wed Oct 27 21:38:25 1999)
%%BoundingBox: 0 0 128 128
%%DocumentData: Clean7Bit
```

Interesuje nas linia:

```
%%BoundingBox: 0 0 128 128
```

i ją wpisujemy do pliku `face1.png.bb`. Program convert ma tę właściwość, że tak konwertuje rysunek, iż wygenerowany `BoundingBox` równy jest wymiarom grafiki w pixelach. (Po wykonaniu tej operacji powstały plik EPS możemy skasować.) Można się też posłużyć programem identify z tego samego pakietu:

```
D:\Program Files\ImageMagick-4.1.8>identify face1.png
face1.png 129x129 PseudoClass 9c 5571b PNG 2s
```

Oprócz różnych innych informacji podane są tam wymiary rysunku `129x129` i jest to informacja wystarczająca do ręcznego wygenerowania pliku `.png.bb` (pamiętając o liczeniu od zera).

Do przekształcania plików rastrowych do postaci fontu używać można programu bm2font. Daje on wiele możliwości wpływania na proces konwersji (cieniowania czy ditheringu) tworzonych obrazów czarno białych. Więcej na temat algorytmów cieniowania można poczytać w pracach [14], [15], [13]. Generalnie jest dosyć sprawny, ale otrzymujemy w wyniku tylko czarno białe obrazki. (Ma on jeszcze jedną ciekawą własność — pozwala przygotowywać „wyciągi” kolorowe.)

Poniżej przykład tej samej grafiki przekształconej trzema różnymi metodami: a — bm2font, b – „zwykłe” przekształcenie do EPS, c — przekształcenie uzyskane za pomocą programu kvec. Aby najlepiej porównać uzyskane wyniki należy stronę wydrukować na drukarce.

a b c

Postać polecenia `\special` używanego przez emTeX jest następująca:
`\special{em:graph` ⟨*filename*⟩`[,`⟨*width*⟩`,`⟨*height*⟩`]}`
znaczenie parametrów jest następujące:

⟨***filename***⟩ nazwa pliku graficznego; plik powinien być czarno-biały w formacie BMP, PCX, MSP,
⟨***width***⟩ parametr nieobowiązkowy; szerokość grafiki (podana w jednostkach długości),
⟨***height***⟩ parametr nieobowiązkowy; wysokość grafiki.

Jeżeli nie podamy wymiarów grafiki będzie ona drukowana tak, aby każdemu punktowi mapy bitowej odpowiadał jeden punkt na wydruku (w rozdzielczości używanej drukarki!). Zatem plik

rastrowy o wymiarach 300 × 300 pikseli będzie miał wymiary 1" × 1" na drukarce o rozdzielczości 300 dpi i odpowiednio mniej na drukarce o wyższej rozdzielczości.

8. Kolor w tekście

Oprócz włączania kolorowych ilustracji w tekstach przygotowywanych systemem LaTeX zmieniać można tak kolor liter jak i kolor tła, na którym się one pojawiają. W preambule dokumentu trzeba zadeklarować wykorzystanie pakietu color:

```
\usepackage{color}
```

Kolory mogą być definiowane w jednym z czterech trybów (ang. *model*):

rgb kolory definiowane są za pomocą trzech składowych (czerwonej, zielonej, niebieskiej),
cmyk kolory definiowane są za pomocą czterech składowych: turkusowej,[26] karmazynowej, żółtej i czarnej.
gray tak na prawdę nie mamy do czynienia z kolorami tylko z odcieniami szarości,
named można używać tylko wcześniej zdefiniowanych kolorów (ten tryb nie zawsze jest dostępny).[27]

Składowe przyjmują wartości pomiędzy 0 a 1.

Generalnie (poza sytuacją gdy możemy skorzystać z trybu „named") każdy używany kolor (poza white, red, green, blue, cyan, magenta, yellow, black) powinien być zdefiniowany. Służy do tego polecenie:
`\definecolor{`⟨*name*⟩`}{`⟨*model*⟩`}{`⟨*color specification*⟩`}` gdzie:

⟨***name***⟩ nazwa koloru,
⟨***model***⟩ jeden z dostępnych trybów: gray, rgb, cmyk, named,
⟨***color specification***⟩ 1, 3, 4 liczby z zakresu od 0 do 1 lub nazwa koloru (w zależności od używanego trybu).

Nawet jeżeli korzystamy z trybu named używane kolory musimy definiować! Nie musimy jedynie podawać liczbowych wartości poszczególnych składowych. Jako nazw kolorów możemy użyć nazw pod którymi występują:

```
\definecolor{Dandelion}{named}{Dandelion}
```

Poleceniem definiującym obowiązujący kolor tekstu jest: `\color{`⟨*name*⟩`}`. Działa ono analogicznie jak na przykład `\bf`.

Można też użyć polecenia `\textcolor{`⟨*name*⟩`}{`⟨*text*⟩`}` — zmienia ono tylko kolor wskazanego tekstu.

Kolejnym poleceniem jest: `\colorbox{`⟨*name*⟩`}{`⟨*text*⟩`}` — tworzy pudełko, którego tło przyjmuje zadany kolor oraz `\fcolorbox{`⟨*name1*⟩`}{`⟨*name2*⟩`}{`⟨*text*⟩`}` (pierwszy parametr określa kolor ramki a drugi kolor tła).

Aby zmienić kolor całej strony, używamy polecenia `\pagecolor{`⟨*name*⟩`}` lub `\pagecolor[`⟨*model*⟩`]{`⟨*color specification*⟩`}` (znaczenie parametrów takie jak powyżej).

Poniżej kilka przykładów użycia powyższych poleceń.

[26] Jak na polski przetłumaczyć nazwę koloru *cyan*? Że to nie jest turkusowy — wynika jasno z tabeli na następnej stronie...

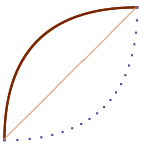
[27] Gdy korzystamy z programu dvips można sprawdzić zawartość pliku `dvipsnam.def` lub `color.pro` — zawarte są tam definicje wszystkich kolorów, które możemy bezpiecznie wykorzystywać.

```
\definecolor{lososiowy}{cmyk}{0,0.53,0.38,0}
\textcolor{lososiowy}{To jest kolor {\l}ososiowy.}\\
\definecolor{szary}{gray}{0.7}
{\color{szary} To jest kolor szary.}\\
\definecolor{zielony}{rgb}{0,.5,0}
\colorbox{zielony}{To jest czarny tekst na zielonym tle.}\\
\fcolorbox{zielony}{szary}{\textcolor{lososiowy}{{\L}ososiowy
tekst na szarym tle w pude{\l}ku o zielonym brzegu}}
```

To jest kolor łososiowy.
To jest kolor szary.
To jest czarny tekst na zielonym tle.
Łososiowy tekst na szarym tle w pudełku o zielonym brzegu

Poleceń można użyć również do nadawania koloru pozornie nietekstowym obiektom (na przykład liniom czy punktom):

```
\begin{picture}(50,50)
   \thicklines
   \textcolor{Brown}{\qbezier(0,0)(0,50)(50,50)}%
   \textcolor{Violet}{\qbezier[20](0,0)(50,0)(50,50)}%
   \thinlines
   \put(0,0){\textcolor{Tan}{\line(1,1){50}}}%
\end{picture}
```



Poniżej przedstawiam tabelę ze wszystkimi nazwanymi w pliku `dvipsnam.def` kolorami (na wzór tej umieszczonej w [3]).

GreenYellow
Yellow
Goldenrod
Dandelion
Apricot
Peach
Melon
YellowOrange
Orange
BurntOrange
Bittersweet
RedOrange
Mahogany
Maroon
BrickRed
Red
OrangeRed
RubineRed
WildStrawberry
Salmon
CarnationPink
Magenta
VioletRed
Rhodamine
Mulberry
RedViolet
Fuchsia
Lavender
Thistle
Orchid
DarkOrchid
Purple
Plum
Violet
RoyalPurple
BlueViolet
Periwinkle
CadetBlue
CornflowerBlue
MidnightBlue
NavyBlue
RoyalBlue
Blue
Cerulean
Cyan
ProcessBlue
SkyBlue
Turquoise
TealBlue
Aquamarine
BlueGreen
Emerald
JungleGreen
SeaGreen
Green
ForestGreen
PineGreen
LimeGreen
YellowGreen
SpringGreen
OliveGreen
RawSienna
Sepia
Brown
Tan
Gray
Black
White

9. Inne efekty specjalne

9.1. Skalowanie obiektów

Polecenie `\scalebox{`⟨*h-scale*⟩`}[`⟨*v-scale*⟩`]{`⟨*argument*⟩`}` pozwala[28] przeskalować dowolny obiekt występujący jako *argument*. Gdy ⟨*v-scale*⟩ zostanie pominięty — obiekt będzie skalowany z zachowaniem proporcji.

```
\scalebox{2}[3]{Ala ma kota}
```

Ala ma kota

⟨*h-scale*⟩ i ⟨*v-scale*⟩ to bezwymiarowe współczynniki skalowania obiektu.

Zamiast `\scalebox` można użyć polecenia `\resizebox{`⟨*width*⟩`}{`⟨*height*⟩`}{`⟨*argument*⟩`}`

[28] Wymaga wykorzystania pakietu graphicx.

```
\resizebox{1cm}{1cm}{Ala ma kota}
\resizebox{1cm}{!}{Ala ma kota}
\resizebox{!}{1cm}{Ala ma kota}
```

Ala ma kota Ala ma kota Ala ma kota

powodującego przeskalowanie obiektu do zadanych wymiarów. ⟨*width*⟩ i ⟨*height*⟩ to mianowane liczby określające wymiary do jakich zostanie przeskalowany obiekt. Użycie jako jednego z argumentów wykrzyknika spowoduje przeskalowanie obiektu z zachowaniem proporcji.

Podczas skalowania fragmentów tekstu pamiętać trzeba, aby używać (w miarę możliwości) fontów skalowalnych: albo Type1 albo (jeżeli nasza implementacja (La)TEXa na to pozwala) fontów TrueType. W przeciwnym wypadku możemy spodziewać się takich efektów:

```
\newfont{\testowy}{ecrm0500}
...
\scalebox{10}{{\testowy Ala ma kota}}
```

Ala ma kota

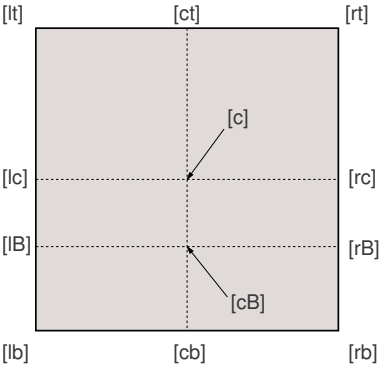
Podobnie zachowają się i inne grafiki rastrowe.

9.2. Obroty obiektów

Polecenie `\rotatebox[`⟨*options*⟩`]{`⟨*angle*⟩`}{`⟨*argument*⟩`}`[29] może być wykorzystane do obrotu dowolnego obiektu. W przypadku gdy chcemy obracać grafikę dostarczaną w postaci pliku EPS znacznie prościej (i efektywniej) będzie skorzystać z możliwości oferowanych przez polecenie `\includegraphics`.

Parametr ⟨*options*⟩ pozwala na określenie między innymi współrzędnych punktu wokół którego będzie obracany obiekt. Można je podać w postaci `x=`⟨*xdim*⟩`,y=`⟨*ydim*⟩ co wskaże bezwzględne współrzędne punktu obrotu (⟨*xdim*⟩ i ⟨*ydim*⟩ to, oczywiście, liczby mianowane.).

Alternatywnie można współrzędne podać w sposób „względny” korzystając ze schematu przedstawionego na rysunku 7; wówczas odpowiedni parametr będzie miał postać `origin=`⟨*współrzędne*⟩.



Rysunek 7: Sposób oznaczania charakterystycznych punktów obiektu

Obrócony obiekt ma właściwe wymiary i może być umieszczany w dowolnym miejscu tekstu bez żadnych dodatkowych zabiegów. Zwracam jednak uwagę, że większość (wszystkie??) programów do oglądania `.dvi` nie będzie w stanie pokazać efektu obrotu na ekranie. Obrót (podobnie jak i skalowanie opisane wcześniej czy efekt „landscape”) realizowane jest za pomocą specjalnych poleceń języka PostScript.

[29] Wymaga użycia pakietu graphicx.

```
\begin{tabular}{|c|c|}
\hline
\rotatebox{90}{Liczba porz\k{a}dkowa} & obiekt \\
\hline
      1         &  aaa   \\
      2         &  bbb   \\
\hline
\end{tabular}
```

Liczba porządkowa	obiekt
1	aaa
2	bbb

Jeżeli zachodzi potrzeba umieszczenia w tekście dużej tabeli "w poprzek" — można w tym celu użyć pakietu „lscape”:

```
\usepackage{lscape}
```

i środowiska `landscape`

10. Poprawianie plików EPS

Znam takich ludzi, którzy twierdzą, że lepiej jest nauczyć się „programowania” w języku PostScript niż korzystać z jakichś wymyślnych programów do przygotowania niezbyt skomplikowanych rysunków. Nie namawiam nikogo do takiego działania.

Istnieje jednak czasami potrzeba zmodyfikowania (lub takiego przygotowania) pliku EPS aby umieścić na rysunku symbole normalnie niedostępne w programie którego używamy — na przykład opis w języku polskim czy symbole matematyczne. Choć, oczywiście, najlepiej używać dobrego oprogramowania (to znaczy takiego, po którym nie trzeba nic poprawiać).

Do osiągnięcia takich efektów należy użyć pakietu psfrag. Pozwala on w sposób „automagiczny” podmienić wstawione „znaczniki” zadanymi ciągami znaków.

Dodatkowo pozwala on w opisach umieszczać polecenia LaTeXa, które zostaną odpowiednio „zinterpretowane” na etapie przetwarzania PostScriptu.

Procedura postępowania jest następująca:

1. W dokumencie musimy użyć pakietu graphicx (lub graphics).
2. Dodatkowo używamy pakietu psfrag.
3. W przygotowywanym rysunku umieścić musimy „znaczniki”; są to krótkie teksty, najlepiej **jednowyrazowe**; umieszczamy je w tych miejscach, gdzie chcemy umieścić „specjalne” teksty.
4. W dokumencie używamy polecenia `\psfrag` które spowoduje zastąpienie każdego znacznika zadanym przez nas tekstem.
5. Polecenie `\psfragscanon` „włącza” a `\psfragscanoff` „wyłącza” działanie procedur podmiany ciągów znaków.

Polecenie `psfrag` ma następującą postać:

`\psfrag{`⟨***tag***⟩`}[[`⟨***pos***⟩`]][`⟨***pspos***⟩`]][`⟨***scale***⟩`]][`⟨***rot***⟩`]{`⟨***replacement***⟩`}`

Znaczenie poszczególnych parametrów jest następujące:

⟨***tag***⟩ znacznik: tekst, którego **każde** wystąpienie w tekście pomiędzy poleceniem `\psfragscanon` a `\psfragscanoff` będzie zastępowane;

⟨***pos***⟩ punkt odniesienia wstawianego tekstu, dwie litery: jedna z zakresu {`t`, `b`, `B`, `c`} a druga {`l`, `r`, `c`} (patrz również rysunek 7); wartość domyślna `cc`;

⟨***pspos***⟩ punkt odniesienia tekstu PostScriptowego; wartość domyślna `cc`;

⟨***scale***⟩ współczynnik skali, domyślnie 1;

⟨***angle***⟩ kąt obrotu (w stopniach) wstawianego tekstu wokół punktu odniesienia; standardowo 0;

⟨***replacement***⟩ wstawiany tekst (lub ogólniej obiekt) LaTeXowy.

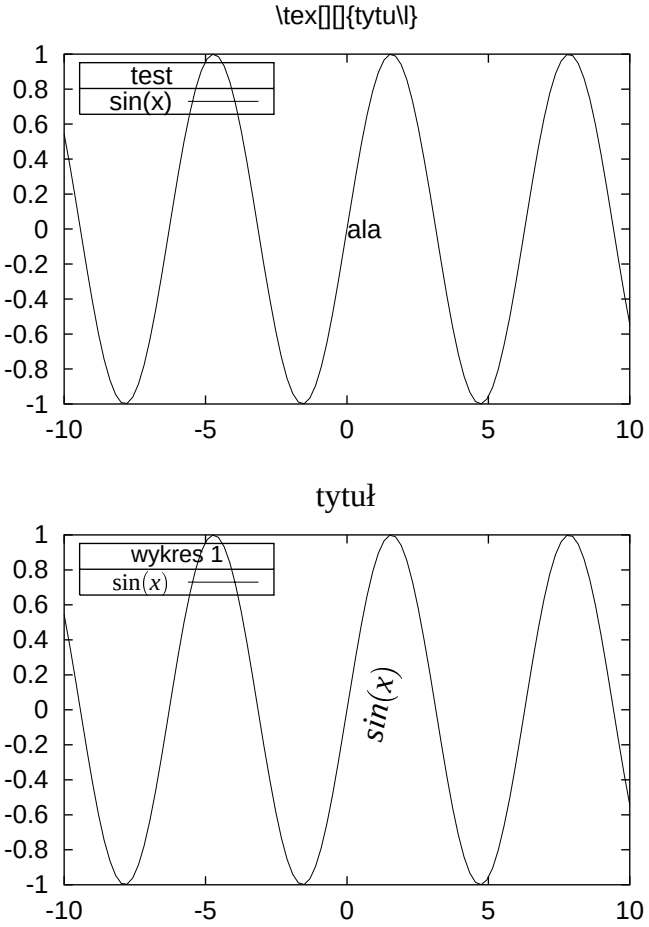
Jeżeli chcemy, aby pewne elementy tekstu w pliku graficznym były interpretowane zgodnie z regułami LaTeXa muszą mieć one postać:

`\tex[`⟨***pos***⟩`]][`⟨***scale***⟩`]][`⟨***rot***⟩`]{`⟨***LaTeX text***⟩`}`

Znaczenie parametrów jest identyczne jak parametrów polecenia `psfrag`.

Poniższy przykład obrazuje niektóre możliwości pakietu.

```
\includegraphics[width=.6\textwidth]{psfrag}
%
\psfrag{ala}[t][t][1][79]{$sin(x)$}
\psfrag{test}[l][l][.8]{{\sf wykres 1}}
\psfrag{sin\(x\)}[r][r][.75]{$\sin(x)$}
%
\psfragscanon
\resizebox{.6\textwidth}{!}{\includegraphics{psfrag}}
\psfragscanoff
```



Bardzo często słyszę zadawane pytanie: „Czy jest jakieś narzędzie które przeczyta plik [E]PS, pozwoli na jego edycję a następnie zapisze w postaci EPS?”

Trudno znaleźć narzędzie które będzie bardzo dobre i tanie. Różne poszukiwania i wnikliwe przysłuchiwanie się dyskusjom na comp.lang.postscript[30] pozwoliło znaleźć następujące narzędzia:

— SGI IPE (patrz `ftp://ftp.cs.uu.nl/pub/SGI/IPE`) — ale nie przetestowałem! Program dostępny w źródłach; potrzebuje kompilatora C++, autor oprogramowania kompilował go na komputerach pod równymi wersjami SO Unix: Linux, IRIX, Solaris i HP–UX.

— pstoedit (patrz również na stronie 32). Konwertuje pliki (E)PS do kilku wektorowych formatów: HPGL, Windows MetaFile, Xfig (patrz również na stronę 32), DXF,[31] PDF, Adobe

[30] Problem ten dosyć często tam gości.

[31] Przenośny format używany przez wiele programów CAD.

Illustrator i kilku innych jeszcze formatów (których przydatności nie potrafię ocenić). Wymaga, oczywiście, dodatkowego narzędzia do przekształcania wynikowego pliku (ponoć działa świetnie w tandemie z programem Xfig). Ze względu na konwersję do formatu WMF może nadać się do przekształcania EPSów do postaci strawnej dla Worda...

— MayuraDraw — program dostarczany jest z prostym plikiem wsadowym, który korzystając z programu ghostscript konwertuje plik (E)PS do postaci Adobe Illustrator. Takie pliku MayuraDraw potrafi już importować. W programie można dokonać poprawek i wyeksportować do postaci EPS.

— CorelDRAW! — nie korzystałem.[32]

— ps_conv jest to program pomocniczy przygotowany przez P. Pianowskiego i B. Jackowskiego służący do konwersji jednostronicowych plików PS do postaci EPS. Dodatkowo wszystkie litery są zamieniane na krzywe. Tak przekształcone pliki mogą być bez problemów czytane przez CorelDRAW!, Adobe Illustrator czy Fontographer. Pewną wadą programu jest to, że wykorzystać go można w środowisku DOS (w każdym innym przypadku wymaga pewnych drobnych korekt). Patrz również na stronie 32.

11. Inne sposoby przygotowywania rysunków

11.1. Czcionki użyte do opisu rysunku[33]

Ilustracje są integralnym elementem dokumentu. Nie powinny się z nim „kłócić". W szczególności dotyczy to czcionki opisującej ilustracje, która powinna być albo taka sama jak czcionka dokumentu, albo powinna z nią współgrać i być konsekwentnie użyta we wszystkich rysunkach.

Jeśli rysunek jest po prostu „wstawionym pudełkiem" bez tekstu wewnątrz — nie ma problemu; polecenie `\caption` załatwia sprawę. Natomiast jeśli rysunek zawiera opisy które mają być złożone czcionką TEX-a — jesteśmy w trochę trudniejszej sytuacji. Możemy:

— jeśli wszystkie wstawiane rysunki tworzone są w jednym programie zewnętrznym, we wszystkich opisach stosować konsekwentnie tą samą czcionkę, dobraną tak, aby pasowała do czcionki dokumentu;

— użyć w programie graficznym, w którym robimy rysunek, tej samej czcionki, którą składamy tekst w LATEX-u, np. Computer Modern PS; działa to bez większych problemów np. w Adobe Illustratorze;

— użyć omówionego wcześniej pakietu psfrag;

— zrobić rysunek w METAPOST-ie, który pozwala dołączyć dowolny tekst (La)TEX-a;

— zrobić rysunek wewnątrz LATEX-a, używając jakiegoś pakietu graficznego lub standardowego otoczenia picture.

Możemy również w ogóle nie przejmować się kwestią czcionek użytych w opisach wstawianych rysunków. Czasami to nie przeszkadza.

11.2. Metapost

METAPOST to system o strukturze podobnej do systemu METAFONT: posiada on rozbudowany zestaw poleceń i kompilator zamieniający je — w złożonym nierzadko procesie — w plik EPS. Nie opisuję tu wszystkich możliwości systemu odsyłając do lektury dokumentacji [10, 12].

[32] Polecający go Marcus Gastreich <ghost@pcgate.thch.uni-bonn.de> napisał tak: „... corel draw (v.>6): afaik that does walk through pixel space. all files generated with corel were like 20times as big as before." Ale nie potrafię się odnieść do tej informacji. Są też doniesienia, że zapis w postaci EPS trwa strasznie długo, a powstające pliki są ogromne.

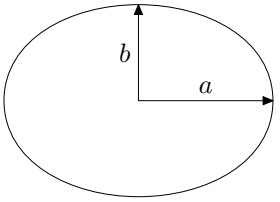
[33]

Do przygotowywania rysunków można również użyć programu METAFONT (różne ciekawe informacje na ten temat znaleźć można, na przykład, w [12]). Jednak obiekty, które on tworzy są znakami, które muszą być zamieniane do postaci mapy bitowej osobno dla każdej rozdzielczości. Opracowano nawet specjalizowany pakiet o nazwie mfpic[34] do łatwiejszego przygotowywania rysunków [12]. Inną alternatywą może być użycie pakietu mf-ps opracowanego przez Bogusława Jackowskiego.

METAPOST dający na wyjściu dosyć prosty plik EPS jest pod tym względem znacznie wygodniejszy. Za używaniem programów METAFONT/METAPOST przemawia fakt, że nie wychodzimy poza programy z pewnej rodziny... Dodatkową dyskusję różnych aspektów tego zagadnienia znajdziemy również w [4] czy [12].

Poniżej dwa przykłady:

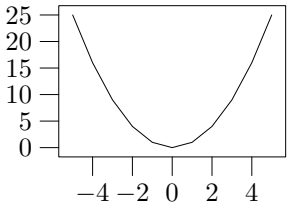
```
beginfig(1);
a=.7in; b=0.5in;
z0=(0,0); z1=(a,0); z2=(0,b);
z0=.5[z1,z3]=.5[z2,z4];
draw z1..z2..z3..z4..cycle;
drawarrow z0..z1;
drawarrow z0..z2;
label.top(btex $a$ etex, .5[z0,z1]);
label.lft(btex $b$ etex, .5[z0,z2]);
endfig;
end
```



Przykład ten można objaśnić następująco: `z0` *to współrzędne środka elipsy o półosiach* `a` *i* `b`. `z1` *i* `z2` *to dwa wierzchołki elipsy. Współrzędne punktów* `z3` *i* `z4` *dobierane są tak, aby* `z0` *leżał na środku odcinków* `[z1,z3]` *i* `[z2,z4]`. *Polecenie* `draw` *nakazuje połączyć wszystkie punkty linią zamkniętą (*`cycle`*).* `drawarrow` *rysuje strzałki,* `label` *wstawia napisy...*

W ten sposób można tworzyć całkiem skomplikowane rysunki. Można też tworzyć specjalne makra ułatwiające tworzenie powtarzalnych rysunków. Oto najprostszy przykład. Plik `dane.dat` zawiera dwie kolumny danych — współrzędne punktów pewnego wykresu:

```
input graph;
beginfig(1);
 draw begingraph(3cm,2cm);
 gdraw "dane.dat";
 endgraph;
endfig;
end
```



11.3. PSTricks

PSTricks to bardzo obszerny zestaw makr służących do rysowania z wykorzystaniem możliwości udostępnianych przez język PostScript. Makra mogą być wykorzystywane zarówno w LaTeX 2ε jak i w plain-TeXu.

Zaznam, że o ile do tej pory mówiłem o włączaniu do dokumentu ilustracji zrobionych *na zewnątrz* LaTeX-a, to używając pakietu PSTricks rysujemy *wewnątrz* dokumentu LaTeX-a.

Każde z makr generuje pewien zestaw poleceń w języku PostScript, które są włączane jako obiekty **special** do pliku DVI. Nie będą one zazwyczaj interpretowane (i widoczne) gdy używamy „zwykłej" przeglądarki DVI. Jednak po przetworzeniu DVI do PS, na przykład za pomocą programu dvips, uzyskujemy żądane rezultaty.

[34] `http://sunsite.icm.edu.pl/pub/CTAN/graphics/mfpic/`

Pakiet PSTricks powstał znacznie wcześniej niż LaTeX 2ε i z tego powodu wśród jego poleceń znaleźć można też funkcje dublujące się z tymi oferowanymi przez pakiety graphics, graphicx, color.

Główną chyba wadą systemu PSTricks jest to, że LaTeX nic nie wie na temat wielkości generowanego obiektu. Panować nad tym musi autor zamykając cały rysunek w otoczeniu `picture` lub `pspicture`.

Mały przykład rysunku przygotowanego tym pakietem znaleźć można na stronie 30.

11.4. Pakiet XY-pic[35]

Pakiet XY-pic, autorstwa Kristoffera Hogsbro Rose i Rossa Moore'a, służy do robienia rysunków wewnątrz (La)TeX-a. Rysunki składa ze specjalnych czcionek, przez co jest niezależny od DVI-procesora. Dodatkowo istnieje możliwość generowania rysunków jako PostScript — wykorzystuje wtedy polecenia `\special` DVIPS-a — co w niektórych przypadkach owocuje „gładszymi" rysunkami, lepiej nadającymi się do wstawiania do PDF-ów.

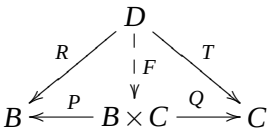
Pakiet XY-pic szczególnie dobrze nadaje się do rysowania diagramów przemiennych, grafów, wielościanów, węzłów, two-cell diagramów. Tutaj podamy przykłady rysowania diagramów przemiennych i grafów.

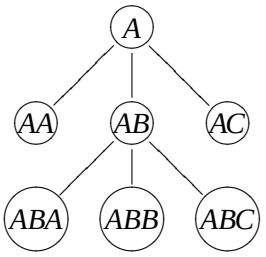
```
\usepackage[arrow,matrix,tips,curve]{xy}

\[
\xymatrix{                          % będziemy rysować diagram przemienny
 &D \ar[dl]_R \ar[dr]^T \ar@{-->}[d]^F \\
                                    % pierwszy wiersz tabeli wolne miejsce,
                                    % etykieta D strzałka do do-u w lewo
                                    % z etykietą R umieszczoną u góry
                                    % strzałka do dołu w prawo z etykietą
                                    % T umieszczoną u góry przerywana
                                    % strzałka do dołu z etykietą F
                                    % umieszczoną z lewej strony
B & B \times C \ar[l]_P \ar[r]^Q & C    % drugi wiersz tabeli: 3 pola
}                                   % i 2 strzałki; koniec rysunku
\]
```

```
\usepackage[graph,frame]{xy}

\[
\xygraph{                   % będziemy rysować graf []
*+[o]+[F]{A}                % ,A' w obwódce owalnej (+[o]) z narysowaną
                            % ramką (+[F])
  (-[dl] *+=[o]+[F]{AA}     % kreska (-) do dołu w lewo ([dl])
  ,-[d]  *+=[o]+[F]{AB}     % kolejna gałąź; do dołu ([d])
    (-[dl] *+=[o]+[F]{ABA} % podgraf --- działamy rekurencyjnie
    ,-[d]  *+=[o]+[F]{ABB} % kolejna gałąź podrafu
    ,-[dr] *+=[o]+[F]{ABC} % jeszcze jedna gałąź podgrafu; do dołu
                            % w prawo ([dr])
    )                       % koniec podgrafu
```

[35] © 1999 by Krzysztof Pszczoła.

```
  ,-[dr] *+=[o]+[F]{AC}     % kolejna gałąź górnego grafu
  )                         % koniec górnego grafu
}                           % koniec rysunku
\]
```





Po więcej informacji odsyłam do dokumentacji pakietu: [22], [23] oraz do 5. rozdziału podręcznika [6].

11.5. Specjalistyczne pakiety graficzne[36]

Istnieje szereg specjalistycznych pakietów do LaTeX-a, które pozwalają na uzyskiwanie specjalnych efektów graficznych. Są to na przykład:

XyMTeX zestaw pakietów do rysowania chemicznych wzorów strukturalnych; `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/other/xymtex/`;

FeynMF pakiet do rysowania fizycznych diagramów Feynmana; `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/supported/feynmf/`;

circ pakiet do rysowania schematów elektrycznych; `http://sunsite.icm.edu.pl/pub/CTAN/macros/generic/diagrams/circ/` oraz podobny: `circuit_macros` `http://sunsite.icm.edu.pl/pub/CTAN/graphics/circuit_macros/`;

MusiXTeX pakiet do rysowania nut muzycznych; istnieje do niego kilka preprocesorów; `http://sunsite.icm.edu.pl/pub/CTAN/macros/musixtex/`;

chess narzędzia do rysowania diagramów do gry w szachy; `http://sunsite.icm.edu.pl/pub/CTAN/fonts/chess/`.

Więcej informacji znaleźć można w rozdziałach 6–8 pracy [6].

12. Rysunki „oblane” tekstem

Czasami zachodzi konieczność (zwłaszcza, gdy rysunek jest niewielki) „oblania” go tekstem. Operacja ta nie zawsze przynosi dobre efekty, zwłaszcza, gdy mamy za mało tekstu do „oblania” rysunku. Inne niż tekst „obiekty” nie bardzo się do tego nadają. TeX posiada całkiem sprawne narzędzie (polecenie `\parshape`) do nadawania poszczególnym wierszom tekstu właściwej długości. Posiadaczy biuletynu GUST odsyłam do tekstu Janusza M. Nowackiego [18]. Problem polega jedynie na automatycznym wyliczeniem długości wierszy...

Aby „oblanie” rysunku tekstem udało się — potrzeba dosyć dużo tekstu. I najlepiej, żeby nie były to jakieś środowiska czy wzory — tylko **zwykły** tekst.

Najprostszym rozwiązanie tego problemu jest zamknięcie obrazka i tekstu w osobnych środowiskach „minipage” o odpowiedniej szerokości i umieszczenie ich obok siebie.

Wadą takiego rozwiązania, jest to, że jakakolwiek zmiana w ministronie z tekstem może popsuć cały efekt wizualny.

[36] © 1999 by Krzysztof Pszczoła.

Inne możliwości daje pakiet o nazwie `wrapfig`. Udostępnia on dwa środowiska o nazwach `wrapfigure` i `wraptable` pozwalające osiągnąć efekty, które (czasami) spełnią nasze wymagania...

Użycie ich jest następujące:

`\begin{wrapfigure}[`⟨*nl*⟩`]{`⟨*placement*⟩`}[`⟨*overhang*⟩`]{`⟨*width*⟩`}`
⟨*figure*⟩
`\end{wrapfigure}`

Znaczenie poszczególnych parametrów jest następujące:

⟨***nl***⟩ Nieobowiązkowy parametr, mówiący ile linii tekstu powinno być „krótszych” (w normalnych warunkach wartość ta zostanie wyznaczona automatycznie, czasami jednak zachodzi konieczność korekty).

⟨***placement***⟩ Obligatoryjny parametr mówiący gdzie ma być umieszczony rysunek:

r po prawej stronie,

l po lewej stronie,

i „wewnątrz” (przy druku dwustronnym — inaczej dla stron parzystych, inaczej dla nieparzystych),

o „na zewnątrz”.

Pojęcia „wewnątrz” i „zewnątrz” dotyczą tej strony kartki, która jest dalej lub bliżej zszycia. W przypadku druku dwustronnego — dla stron nieparzystych `o` znaczy tyle samo co `r`.

⟨***overhang***⟩ Określa jak bardzo rysunek będzie „wystawa-” na margines (normalnie nie będzie wystawa-).

⟨***width***⟩ Definiuje szerokość wstawianego rysunku.

⟨***figure***⟩ Wstawiany rysunek.

Zamiast pakietu wrapfig można użyć również: floatflt lub picins.

W każdej sytuacji musimy zdawać sobie sprawę z tego, że nawet bardzo mała zmiany w tekście mogą powodować poważne perturbacje z rozkładem ilustracji. Tak więc zawsze ilustracje rozmieszczamy w **ostatecznej** wersji tekstu.

13. Znaki wodne

Właściwie to wykracza poza zasadniczy temat (włączanie do tekstu ilustracji czy wykresów), ale przygotowywując jakiś tekst lub slajdy możemy zechcieć umieścić na każdej stronie logo...

Aby osiągnąć taki efekt musimy posłużyć się pewną „sztuczką”. Pakiet nazywa się `fancyhdr`. Pozwala on (i jest to jego zasadnicza funkcja) bardzo łatwo definiować i modyfikować wygląd paginy górnej (to co nazywane jest *header* w pakiecie fancyhdr i gdzie umieszczana jest żywa pagina) i paginy dolnej (ang. *footer* gdzie umieszczane są przypisy czy notki).[37]

Sztuczka polegać będzie na tym, że żywą paginę lub notkę będziemy definiować tak aby zawierała znak graficzny, który chcemy umieścić na każdej stronie. Pamiętać przy tym należy, że żywa pagina drukowana jest **przed** wydrukowaniem zawartości strony a notka **po**. Zatem grafika nakładana w notce może (o ile jest „nieprzezroczysta”) przysłonić tekst.

Pamiętać trzeba o odpowiednim pozycjonowaniu obiektu który chcemy umieścić na stronie. Dokonać tego można za pomocą polecenia `\put`.

Najprostsze rozwiązanie wyglądać może zatem tak:

```
\fancyhead[L]%
```

[37] Dziękuję Tomkowi Przechlewskiemu za zwrócenie uwagi na niepoprawne używanie określeń „nagłówek strony” i „stopka strony”. Próbowałem odnaleźć jakieś definicje w [17] i [1] ale bez większego rezultatu. Stąd konwencja „pagina górna (dolna)” na określenie **miejsca** i „żywa pagina” oraz „notka” na określenie zawartości.

```
{
 \unitlength 1cm
 \begin{picture}(0,0)
  \put(0,-20){\includegraphics[width=\textwidth]{obiekt}}
 \end{picture}
}
```

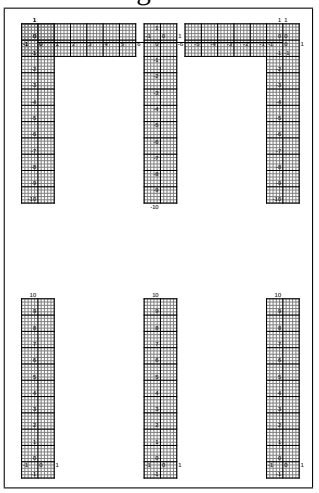
Niestety, nie jest to najlepsze rozwiązanie: podczas tworzenia każdej strony powyższe polecenia będą za każdym razem interpretowane. Lepszym rozwiązaniem może być utworzenie z obiektu graficznego „kontenera” (`\savebox`) i włączanie go do tekstu:

```
\newsavebox{\mygraphics}
\sbox{\mygraphics}{\includegraphics[width=\textwidth]{obiekt}}
...
\fancyhead[L]{\setlength{\unitlength}{1cm}
\begin{picture}(0,0)
\put(0,-20){\usebox{\mygraphics}}
\end{picture}}
```

Niestety, plik graficzny będzie włączony do wynikowego pliku PostScriptowego wielokrotnie. Osoby obeznane z PostScriptem mogą podjąć trud takiego przygotowania pliku EPS aby włączyć go tylko raz i wielokrotnie wykorzystać na każdej stronie. Wymaga to jednak pewnej pracy.

Aby ułatwić sobie określenie pozycji w której wstawiany ma być obrazek można posłużyć się następującymi poleceniami, które produkują coś w rodzaju papieru milimetrowego:

```
\fancyhead[L]{\psgrid(0,0)(-1,-10)(1,  1)%
 \psgrid(0,0)(-1,-1)( 6,1)}
\fancyhead[R]{\psgrid(0,0)(-1,-10)(1,  1)%
 \psgrid(0,0)(-1,-1)(-6,1)}
\fancyhead[C]{\psgrid(0,0)(-1,  1)(1,-10)}
\fancyfoot[L]{\psgrid(0,0)(-1, -1)(1, 10)}
\fancyfoot[R]{\psgrid(0,0)(-1, -1)(1, 10)}
\fancyfoot[C]{\psgrid(0,0)(-1, -1)(1, 10)}
```



14. Lektury dodatkowe

„Nie za krótkie wprowadzenie do systemu LaTeX 2ε” [19] może być dzisiaj uznane za podstawową lekturę każdego kto chce zacząć korzystać z LaTeX 2ε. Zawarte są tam również najbardziej elementarne informacje na temat dołączania rysunków w formacie EPS (rozdział 4.1).

Jako podstawową lekturę omawiającą wszelkie problemy związane z przygotowywaniem grafiki w LaTeX 2ε „w ciemno” (bo sam nie czytałem) polecam książkę Michela Goossensa, Sebastiana Rahtza i Franka Mittelbacha: The LaTeX Graphics Companion [6]. Jeżeli ktoś nie ma do niej dostępu — trudno. Będzie musiał zadowolić się materiałami dostępnymi w sieci (w tym i przykładami z książki: `ftp://sunsite.icm.edu.pl/pub/CTAN/info/lgc/` i `ftp://sunsite.icm.edu.pl/pub/CTAN/graphics/pstricks/doc/lgc/`). Na całe szczęście różnych pozycji jest dosyć dużo.

— Packages in the ‘graphics’ bundle [2],
— Using Imported Graphics in LaTeX 2ε [21],
— The psfrag system, version 3 [7],
— Graphics and Colour with LaTeX [3].

Bardzo wiele pakietów systemu LaTeX 2ε rozpowszechnianych jest w postaci plików `.dtx`. W jednym pliku zawarta jest wówczas sam pakiet wraz z jego dokumentacją. Przetwarzając plik `.dtx` za pomocą polecenia:
`latex` ⟨*filename*⟩`.dtx`
uzyskamy bardzo wiele informacji na temat używanego pakietu.

W wielu dystrybucjach systemu TeX dokumentacja ta dostępna jest w wersji już przetworzonej (w postaci plików `.ps` lub `.dvi`).

15. Źródła

Odsyłacze do większości wymienionych tu pakietów i programów znaleźć można w `http://sunsite.icm.edu.pl/pub/CTAN/help/Catalogue/catalogue.html` lub innym serwerze należącym do sieci CTAN[38] lub lusterek (mirrorów); oficjalny ich spis znajduje się w `http://sunsite.icm.edu.pl/pub/CTAN/CTAN.sites`. Poniżej podajemy bardziej dokładne odsyłacze do miejsc, gdzie programy można znaleźć (w miarę możliwości będą to miejsca w krajowe — co nie znaczy, że łatwo dostępne).

cep Program znajdziemy w: `http://sunsite.icm.edu.pl/pub/CTAN/support/pstools/cep/`. Wraz z programem cep służącym do kompresji rastrowych plików EPS znajdziemy tam program cop służący do kompresji dowolnych plików PostScriptowych. Podstawowa dokumentacja zawarta jest w pliku `cepcop_p.inf`[39]; dalsze szczegóły można znaleźć też w [24]. Oprogramowanie wykorzystuje własność języka PostScript Level 2 oferującą możliwość kompresji/dekompresji fragmentów programu.

color Pakiet wchodzi w skład pakietu graphics/graphicx (`http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/required/graphics/`).

egplot `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/supported/egplot/`.

fancyhdr `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/supported/fancyhdr/`

floatflt Pakiet o możliwościach zbliżonych do wrapfig; dostępny jako: `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/other/floatflt/`

gnuplot Po okresie dosyć długiego zastoju, wraz z rozwojem systemu Linux wrócił do łask gnuplot. Jest bardzo intensywnie rozwijany (ostatnio pojawiła się wersja 3.7 patchlevel 1), a strona poświęcona mu znajduje się pod adresem `http://www.geocities.com/SiliconValley/Foothills/6647/` lub `http://members.theglobe.com/gnuplot/`; FAQ na temat programu znaleźć można w `http://wwwfg.rz.uni-karlsruhe.de/~ig25/gnuplot-faq.html/`. Źródła są w `ftp://sunsite.icm.edu.pl/pub/CTAN/graphics/gnuplot/` Dostępne są wersje pracujące w środowisku Windows i w środowisku Unix.

graphics, graphicx Pakiety dostępne jako `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/required/graphics/`

hp2xx `http://sunsite.icm.edu.pl/pub/CTAN/support/hp2xx/`

ImageMagick Bardzo rozbudowany zestaw programów do konwersji pomiędzy różnymi formatami graficznymi (mapy bitowe) oraz przekształcania i modyfikacji obrazków. Pracuje w systemie Unix, został również przeniesiony do środowiska Windows 9x/NT. W tym ostatnim

[38] Comprehensive TeX Archive Network
[39] {`http://sunsite.icm.edu.pl/pub/CTAN/support/pstools/cep/cepcop_p.inf`}

do manipulacji grafikami na ekranie wymaga pracującego X–serwera; pozostałe programy mogą być wywoływane z linii komend („okno DOS”). Poszukiwania rozpocząć należy od: http://www.wizards.dupont.com/cristy/ImageMagick.html.

jpeg2ps ftp://sunsite.icm.edu.pl/pub/CTAN/support/jpeg2ps/

kvec Program którego autorem jest Karl-Heinz Kuhl dostępny jest pod adresem http://ourworld.compuserve.com/homepages/kkuhl. Pozwala na zamianę pliku graficznego o postaci rastrowej do postaci wektorowej. Podstawowa dokumentacja programu znajduje się w pliku http://ourworld.compuserve.com/homepages/kkuhl/ks_page1.htm

lscape Pakiet wchodzi w skład pakietu graphics/graphicx (http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/required/graphics/).

METAPOST Polecam odwiedzenie oficjalnej strony programu MetaPost. Znajduje się ona w: http://cm.bell-labs.com/who/hobby/MetaPost.html. Oprócz podręcznika programu MetaPost [10] polecić można (również dostępne w postaci elektronicznej) „Introduction to MetaPost” [9] oraz „Drawing graphs with MetaPost” [11] czy „Puzzling graphics in MetaPost” [8].

PageDraw, MayuraDraw Programy dostępne na stronach http://www.mayura.com/.

picins Odpowiednik pakietu wrapfig ale opracowany dla system LaTeX2.09. Można próbować użyć go i w środowisku LaTeX 2ε. Dostępny jako http://sunsite.icm.edu.pl/pub/CTAN/macros/latex209/contrib/picins/.

ps_conv Program PostScriptowy i prosty plik .bat który wykorzystuje ghostscript do konwersji pierwszej strony z pliku PS do postaci EPS. Fonty zamieniane są na krzywe. Dostępny w http://sunsite.icm.edu.pl/pub/CTAN/support/pstools/ps_conv/.

psfrag Pakiet dostępny jako http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/supported/psfrag/

pstoedit Zacząć można od adresu: http://www.geocities.com/SiliconValley/Network/1958/pstoedit/. Istnieje wersja pracująca zarówno w środowisku Unix jak i w środowisku Windows9x/NT. Można szukać również na serwerach CTAN, na przykład w dystrybucji pakietu GSview: ftp://sunsite.icm.edu.pl/pub/CTAN/support/ghostscript/ghostgum/

PSTricks Pakiet dostępny jako http://sunsite.icm.edu.pl/pub/CTAN/graphics/pstricks/.

sterowniki Adobe http://www.adobe.com/prodindex/printerdrivers/main.html (zwracam jednak uwagę że licencja nie pozwala na używanie tego sterownika z drukarką inną niż posiadającą licencjonowany przez Adobe interpreter języka PostScript).

tiff2ps http://sunsite.icm.edu.pl/pub/CTAN/support/pstools/tiff2ps/.

tkpaint http://www.netanya.ac.il/~samy/tkpaint.html

wmf2eps Shareware! http://www.lake.de/home/lake/p60/wmf2eps/. Znakomity program do konwersji „windowsowych” grafik wektorowych (WMF, EMF) do postaci EPS. Pozwala również na konwersję ze „schowka” (*clipboard*).

wrapfig http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/other/misc/.

Xfig http://sunsite.icm.edu.pl/pub/CTAN/graphics/xfig/

Podziękowania

Bardzo serdecznie chciałem podziękować wszystkim czytelnikom wstępnej wersji broszury, którzy zechcieli podzielić się ze mną swoimi uwagami, a w szczególności:

— Przemkowi Piotrowskiemu,
— Wiesławowi Palczewskiemu,
— Stanisławowi Olejnikowi,
— Tomaszowi Przechlewskiemu,
— Jerzemu Kucharczykowi.

Osobne podziękowania należą się Krzysztofowi Pszczole, który (oprócz szeregu cennych uwag) zechciał dostarczyć cztery (pod)rozdziały: 5, 11.4, 11.5 i 11.1.

I na koniec słowa podziękowania dla Staszka Wawrykiewicza — tylko dzięki jego męskim słowom doprowadziłem rzecz całą do końca. Dziękuję!

Bibliografia

[1] Aleksander Birkenmajer, Bronisław Kocowski, Jan Trzynadlowski, redaktorzy. *Encyklopedia wiedzy o książce*. Zakład Narodowy im. Ossolińskich, Wrocław, Warszawa, Kraków, 1971. Encyklopedię otrzymała w spadku moja żona Jadwiga po dziadku, wielkim miłośniku książek. Nie wiem, niestety, na ile informacje w niej zawarte są jeszcze aktualne.

[2] D. P. Carlisle. Packages in the graphics boundle. Dokumentacja pakietów graphics i graphicx. Dostępna w postaci elektronicznej jako: `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/required/graphics/grfguide.ps`, Styczeń 2000.

[3] Patrick W. Daly. Graphics and colour with LaTeX. Dokument dostępny jako: `http://www.loria.fr/services/tex/graph-pack/grf/grf.pdf` lub u autora: `http://www.mpae.gwdg.de/~daly/latex/grf.pdf`, `http://www.mpae.gwdg.de/~daly/latex/grf.htm`, `http://www.mpae.gwdg.de/~daly/latex/grf.ps`, Czerwiec 1998.

[4] Kees Van der Laan. Graphics and TeX — a reappraisal of METAFONT/METAPOST. *Biuletyn Polskiej Grupy Użytkowników Systemu TeX*, Zeszyt 6:51–57, 1996. Dokument dostępny w postaci elektronicznej jako: `ftp://ftp.gust.org.pl/TeX/GUST/bulletin/07/09-kvl.ps`.

[5] Michel Goossens, Frank Mittelbach, Alexander Samarin. *The LaTeX Companion*. Addison-Wesley Pub Co., Styczeń 1994. ISBN: 0201541998.

[6] Michel Goossens, Sebastian Rahtz, Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents With TeX and Postscript*. Addison-Wesley, Reading, Massachusetts, 1997. ISBN 0-201-54199-8 spis treści.

[7] Michael C. Grant, David Carlisle. The PSfrag system, version. Dostępny jako: `http://sunsite.icm.edu.pl/pub/CTAN/macros/latex/contrib/supported/psfrag/pfgguide.ps`, 1998.

[8] Hans Hagen. Puzzling graphics in METAPOST. `http://www.loria.fr/services/tex/prod-graph/metafun.pdf`.

[9] J. D. Hobby. Introduction to MetaPost. *EuroTeX '92 Proceedings*, strony 21–36, Wrzesień 1992. Dostępny w: `http://www.loria.fr/services/tex/prod-graph/mpintro.pdf`. Polskie tłumaczenie znaleźć można w `http://www.gust.org.pl/PDF/mpintrop.pdf`.

[10] J. D. Hobby. A user's manual for MetaPost. Computing Science Technical Report 162, AT&T Bell Laboratories, Murray Hill, New Jersey, 1992. Dostępny w: `http://www.loria.fr/services/tex/prod-graph/mpman.ps.gz` lub `http://www.loria.fr/services/tex/prod-graph/mpman.pdf`.

[11] J. D. Hobby. Drawing graphs with metapost. Computing Science Technical Report no. 164, AT&T Bell Laboratories, Murray Hill, New Jersey, 1993. Dostępny: `http://www.loria.fr/services/tex/prod-graph/mpgraph.pdf`.

[12] Alan Hoenig. *TeX Unbound. LaTeX & TeX Strategies for Fonts, Graphics & More*. Oxford University Press, Inc., 198 Madison Avenue, New York, NY 10016, 1998.

[13] Bogusław Jackowski. Szare jest piękne. *Biuletyn Polskiej Grupy Użytkowników Systemu TeX*, Zeszyt 6:45–50, 1995. Dokument dostępny jako plik PS: `ftp://ftp.gust.org.pl/TeX/GUST/bulletin/06/09-bj2.ps`.

[14] Donald E. Knuth. Digital halftones by dot diffusion. *ACM Transactions on Graphics*, 6:245–273, 1987. Nieco zmodyfikowana wersja tego artykułu została również opublikowana

jako rozdział 22 w pracy [16].

[15] Donald E. Knuth. Fonts for digital halftones. *TUGboat*, 8:135–160, 1987. Nieco zmodyfikowana wersja tego artykułu została opublikowana również jako rozdział 21 w pracy [16].

[16] Donald E. Knuth. *Digital Typography*. CSLI Lecture Notes Number 78. Center for the Study of Language and Information, Leland Stanford Junior University, 1999. ISBN 1–57586–010–4.

[17] Janusz Marian Nowacki. TEXnologia a typografia. *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, Zeszyt 6:1–15, 1995. Artykuł dostępny jest w wersji elektronicznej pod adresem: `ftp://ftp.gust.org.pl/TeX/GUST/bulletin/06/01-jmn.pdf`.

[18] Janusz Marian Nowacki. Przepis na wygodne używanie `\parshape`. *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, Zeszyt 8:64, 1997. Ze streszczenia: „Prezentowany zestaw skryptów AKW–owych oraz makrodefinicji TEX–owych umożliwia oblewanie tekstem ilustracji o dowolnych kształtach”.

[19] Tobias Oetiker, Hubert Partl, Irene Hyna, Elisabeth Schlegl. Nie za krótkie wprowadzenie do systemu LATEX 2ε. Polskie tłumaczenie dokonane przez Janusza Goldasza, Ryszarda Kubiaka i Tomasza Przechlewskiego. Dokument dostępny w postaci elektronicznej: wersja polska: `http://sunsite.icm.edu.pl/pub/CTAN/info/lshort/polish/`; inne tłumaczenia: `http://sunsite.icm.edu.pl/pub/CTAN/info/lshort/`, Wrzesień 1998.

[20] Ewaryst Rafajłowicz, Wojciech Myszka. *LATEX – zaawansowane narzędzia*. Problemy współczesnej nauki. Teoria i zastosowania. Informatyka. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1996. ISBN 83–7101–333–7.

[21] Keith Reckdahl. Using Imported Graphics in LATEX 2ε. Znakomity tekst poświęcony włączaniu grafik w postaci plików EPS do tekstów w LATEX 2ε. Dostępny w postaci elektronicznej jako: `http://sunsite.icm.edu.pl/pub/CTAN/info/epslatex.pdf`, Grudzień 1997.

[22] Kristoffer H. Rose. XY–pic user's guide. Dokument dostępny w postaci elektronicznej: `http://sunsite.icm.edu.pl/pub/CTAN/macros/generic/diagrams/xypic/xyguide.pdf.gz`, Luty 1999.

[23] Kristoffer H. Rose, Ross Moore. XY–pic reference manual. Podręcznik dostępny w postaci elektronicznej: `http://sunsite.icm.edu.pl/pub/CTAN/macros/generic/diagrams/xypic/xyrefer.pdf.gz`, Luty 1999.

[24] Piotr Strzelczyk. Z CEP–em na EPS–y. *Biuletyn Polskiej Grupy Użytkowników Systemu TEX*, Zeszyt 9:79, 1997.

Skorowidz