Zpravodaj Československého sdružení uživatelů TeXu

Jaromír Kuben

Kreslení obrázků v TeXu pomocí mujmfpic

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 4 (1994), No. 2, 87–95

Persistent URL: http://dml.cz/dmlcz/149707

byly na sobě závislé, nedopadlo by to dobře. Příklad, který jsme zde uvedli, pamatuje skoro na vše. Vybereme-li souvislý úsek, vytiskne se vše dobře. Program, který stránky vybírá, musí totiž vždy zkopírovat prolog i trailer. Tím se vše, co jsme napáchali, zase napraví. Pokud ale vybereme několik náhodných stránek, nebude podmíněný příkaz na řádcích 41–53 fungovat správně a výsledkem bude zmatek.

V tomto příspěvku nebylo možné podat vysvětlení PostScriptu. Není zde dostatek prostoru. Jeho cílem bylo pouze upozornění na některé nové možnosti a záludnosti. Pro toho, kdo má zájem dozvědět se o PostScriptu více, lze doporučit např. knížku Davida A. Holzganga: Understanding PostScript (SYBEX, San Francisco), která poslouží jako vhodná základní učebnice pro začátečníky.

Zdeněk Wagner
<wagner@icpf.cas.cz>

Kreslení obrázků v TEXu pomocí `mujmfpic`

Jaromír Kuben

V několika posledních číslech TEX bulletinu se objevila řada příspěvků týkajících se otázky, jak vyrobit v TEXovském dokumentu obrázky nejrůznějšího druhu. Chtěl bych se se zájemci o tuto problematiku podělit o své zkušenosti s balíkem `mfpic`, který využívá pro tyto účely program METAFONT, aniž je třeba znát jeho jazyk (což skalní zastánci tohoto skvělého programu odsuzují).

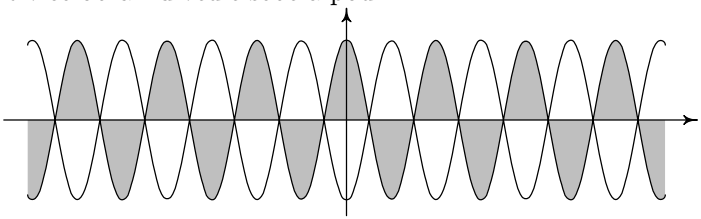
S tímto balíkem ve verzi 0.2 jsem se poprvé setkal před prázdninami 1993. Vzorové obrázky vypadaly pěkně, dobře dopadly i první experimenty, ale současně se objevily i jisté potíže. Protože jsem tehdy měl za sebou určité, sice nevelké ale úspěšné, pokusy s METAFONTem (namaloval jsem obrázky do dvojích skript), pokusil jsem se problémy odstranit. Stálo mne to značné úsilí, ale podařilo se. Přitom mne napadlo, že když už jsem věnoval tolik času na proniknutí do zdrojového kódu, nebylo by od věci rozšířit škálu obrázků, které lze pomocí `mfpic` namalovat. Zejména jsem postrádal křivky dané parametricky nebo v polárních sou-

řadnicích, vadilo mi, že uzavřené křivky, které se měly nějak „vyplnit“, se nesměly protínat, nerespektoval se přesah popisů vlevo, nahoru a dolů mimo plochu obrázku a pod.

Když jsem vše dokončil, dozvěděl jsem se, že autoři uveřejnili další verzi 0.25, která je opravená. „Naštěstí“ šlo skutečně jen o opravu některých chyb a vylepšení průběžných hlášení při překladu, takže moje práce s rozšířením nebyla zbytečná. Svůj „výtvor“ (vycházející z verze 0.2) jsem nazval `mujmfpic` a zaslal ho autorům. Ti se o něm vyjádřili pochvalně a přislíbili, že většinu mých doplňků zahrnou do nové vylepšené verze `mfpic`, která by se podle poslední informace snad mohla v dohledné době objevit (pracuje se na ní).

A nyní konečně něco k praktickému použití. Jde o makra, která lze použít ve formátech `plain`, $\mathcal{AMS}$-TEX i LATEX. Přitom obrázky nakreslí METAFONT a případný popis provede TEX. Kompletní zdrojový kód pro obrázky se napíše do zdrojového TEXovského souboru. Při překladu TEXem je vygenerován pomocný soubor, který obsahuje zdrojový kód pro METAFONT. Tento soubor je třeba přeložit METAFONTem a vzniklý generický font převést na `pk` font. Po dalším překladu TEXem a zavolání prohlížečky se objeví obrázky na obrazovce.

Nejprve je třeba načíst příslušné definice příkazem `\input mujmfpic` pro `plain` resp. `\input mlamfpic` pro LATEX (Pozor! Pokud používáte instalaci $\mathcal{CS}$TEXu a styl `czech`, nesmí být v tomto okamžiku aktivní příkaz `\csprimeson`). V nové verzi už nebude speciální soubor pro LATEX a nebude tudíž speciální LATEXovské okolí pro obrázky — viz níže. Dále příkazem `\opengraphsfile{jmeno}` určíme, jak se bude jmenovat pomocný soubor pro METAFONT. Kód pro jednotlivé obrázky je obklopen dvojicí příkazů `\picture` a `\endpicture` pro `plain` resp. `\begin{mfpic}` a `\end{mfpic}` pro LATEX. Někam za poslední obrázek je třeba umístit příkaz `\closegraphsfile`. Celý obrázek bude vysázen jako `\hbox` na místě odpovídajícím umístění zdrojového kódu. Chová se tedy jako jeden velký symbol a je proto možné jej obtékat, dát do plovoucího materiálu, umístit více obrázků vedle sebe a pod.



Předchozí obrázek byl vytvořen příkazy

```
\opengraphsfile{obr}
\begin{mfpic}[3]{-43}{44}{-12}{14}
\axes
\shadefcn{-40,40}{10*cosd(32*x)}{0}
\function{-40,40,1,10*cosd(32*x)}
\function{-40,40,1,-10*cosd(32*x)}
\end{mfpic}
\closegraphsfile
```

Všechny rozměry jsou zadávány v relativních jednotkách. Parametr `[4]` udává, kolik bodů (pt) má tato jednotka. Ve verzi 0.2 a v `mujmfpic` toto číslo musí být celé, ve verzi 0.25 už tomu tak není. Dále mohou být odlišné jednotky v horizontálním a vertikálním směru, pak se zadá např. `[3][2]`. Další parametry udávají horizontální a vertikální rozměry kreslící plochy jako rozsah souřadnic na osách x a y. Skutečná velikost je tedy v našem případě 261 pt × 78 pt. Další kód je celkem názorný. U příkazu `\shadefcn` jsou zadány dvě funkce (zde $y = 10\cos 32x$ a $y = 0$), mezi nimiž se plocha „vyšedí“. Funkce `cosd` je kosinus s argumentem ve stupních.

Po překladu TEXem vznikne mimo jiné soubor `obr.mf`. Aby byl překlad úspěšný, je třeba umístit soubory `mlamfpic.tex` a `mujmfpic.tex` do některého adresáře, kde TEX hledá vstup. Dále je třeba umístit soubor `mgraphba.mf` do adresáře, kde hledá vstup METAFONT. Pak se přeloží soubor `obr.mf` METAFONTem, což se provede např. příkazem `mf obr`. Výstup bude určen pro zařízení odpovídající volbě `mode=localfont`. Pokud má být pro jiné zařízení, např. tiskárnu Epson, bude vstup např. `\mode=epsonfx;input obr`. Je-li konkrétně `localfont=hplaser`, vzniknou soubory `obr.tfm` a `obr.300`. Číslo 300 znamená, že rozlišitelnost, pro niž byl tento font vygenerován, je 300 dpi. Nyní se zavolá program `gftopk`, který převede tzv. generický font `obr.300` na `pk` font `obr.pk`. Tyto soubory budou v aktuálním adresáři. Proto je třeba zajistit, aby TEX hledal metriky i zde. Např. pro `tex386` je nutné nastavit proměnnou `TEXTFM`. Konečně je třeba zajistit, aby příslušné ovladače (pro obrazovku, tiskárnu a pod.) hledaly bitové mapy fontů také aktuálním v adresáři. Při dalším překladu TEXem bude obrázek zařazen na své místo.

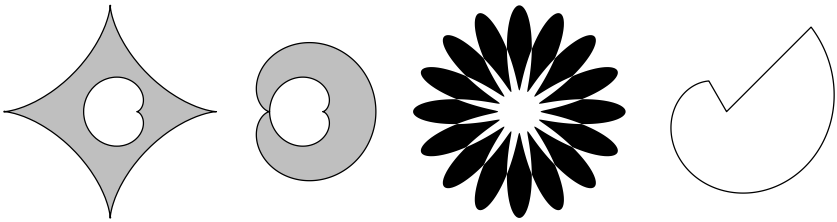
Nyní je na čase přiznat se k malému podvodu. Předchozí obrázek je vycentrován. Ukázalo se ale, že LATEXovské okolí `center` dělá jisté problémy (související zřejmě s přechodem z vertikálního do horizontálního

modu). Ty se projeví tím, že obrázek není přesně uprostřed, popisy jsou zcela odtrženy mimo obrázek a pod. Nepomohl ani příkaz `\leavevmode`. Nakonec jsem problém vyřešil následujícím způsobem. Dodefinoval jsem příkaz `\putintobox`, který má jeden parametr, což musí být malá římská číslice od jedné do pěti, tj. `i` až `v`. Pak příkaz `\putintobox{i}` umístěný kdekoli mezi `\begin{mfpic}` a `\end{mfpic}` způsobí, že obrázek nebude vysazen, ale pouze umístěn do boxu se jménem `\boxi`. Kód pro předchozí obrázek obsahoval právě tento příkaz a vlastní obrázek byl pak umístěn takto:

```
\begin{center}
\leavevmode
\box\boxi
\end{center}
```

Elegantnější by bylo zjistit, co je skutečnou příčinou nežádoucích efektů, ale to se mi nepodařilo. Třeba mi některý zkušenější TEXpert poradí, a pak bude možné tento obrat vypustit.*

Předvedeme si nyní několik ukázek toho, co lze pomocí `mujmfpic` nakreslit. Výběr je volen víceméně náhodně.



Zdrojový kód vypadá takto:

```
\begin{mfpic}[2]{-20}{136}{-20}{20}
\paradiscshade[o]{0,360,6,0,360,6}%
{(20*((cosd(t))**3),20*((sind(t))**3))}%
{(-5*(1+cosd(t))*cosd(t)+5,-5*(1+cosd(t))*sind(t))}
\polardiscshade[o]{(30,0),0,360,(40,0),0,360}%
```

* V plainu funguje osvědčené `\noindent\hfil` (za předpokladu, že není změněna hodnota `\parfillskip`, nebo jiného důležitého parametru). (*Pozn. red.*)

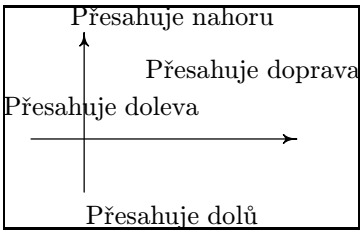
```
{10*(1+cosd(t))}{-5*(1+cosd(t))}
{\shadespace=0pt
\polardiscshade{(77,0),0,360,(77,0),0,360}%
{12+8*cosd(8t)}{12-8*cosd(8t)}}
\polarwedge{(116,0),120,405,t/18}
\end{mfpic}
```

V prvním obrázku jsou obě křivky (asteroida a kardioida) zadány parametricky. Ve druhém obrázku jsou obě kardioidy dány v polárních souřadnicích, Stejně tak je tomu ve třetím obrázku, z něhož je vidět, že křivky se mohou protínat a že místo „vyšedění“ je možné obrázek „vyčernit“ (příkaz `\shadespace=0pt`). Ve čtvrtém obrázku je Archimedova spirála dána v polárních souřadnicích. Tato křivka není uzavřená a příkaz `\polarwedge` spojí její konce se středem souřadnic.

Obrázky je rovněž možné popisovat. K tomu slouží příkaz `\label` (není to standardní příkaz LaTeXu, který platí mimo okolí `mfpic`). Následující obrázek byl vytvořen takto:

```
\begin{mfpic}[2]{-10}{40}{-10}{20}
\axes
\label[bl]{12}{12}{Přesahuje doprava}
\label[br]{22}{5}{Přesahuje doleva}
\label[bc]{17}{22}{Přesahuje nahoru}
\label[tc]{17}{-12}{Přesahuje dolů}
\putintobox{i}
\end{mfpic}
\fboxsep=0pt
\fbox{\box\boxi}
```

Text bude vysázen do boxu, jenž bude vzhledem k referenčnímu bodu umístěn horizontálně vlevo, centrovaně nebo vpravo a vertikálně nahoru, centrovaně nebo dolů. Přesah popisů je brán v úvahu při určování velikosti výsledného boxu (obsahujícího obrázek a popisy). Totéž platí pro případný popis pod obrázkem vytvořený příkazem `\caption` (i tento příkaz je odlišný od standardního příkazu LaTeXu,



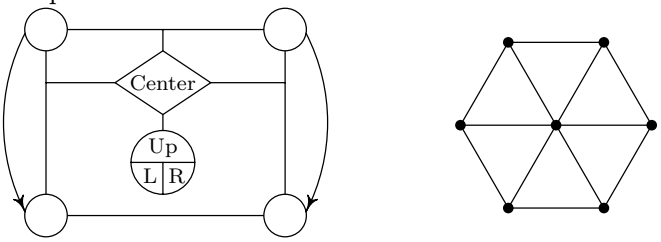
který platí mimo okolí `mfpic`). Současně je vidět, že není problém obrázek obtéct textem.

Poslední ukázka je především pro zájemce o teorii grafů. Celkem jednoduše lze nakreslit i komplikované grafy. Výhodou je, že METAFONT umí „přemazávat“ čáry, takže je možné nejprve nakreslit hrany, a pak teprve uzly (např. jako kroužky). Kdo umí trochu METAFONT, může souřadnice vrcholů zadávat pomocí funkcí, které jsou k dispozici. Proto je snadné vytvořit např. pravidelný šestiúhelník. Zdrojový kód vypadá takto:

```
\footnotesize
\begin{mfpic}[2]{-4}{119}{0}{43}
\polycurve{(4,4),(49,4),(49,39),(4,39),(4,4)}
\line{(26,14),(26,39)}\line{(4,29),(49,29)}
\dkruh{6,(26,14)}\kruhy{4,(4,4),(49,4),(4,39),(49,39)}
\rozhod{(26,29),9,6}
\setlength{\headlen}{6pt}
\arrowcorr{-3}
\curvedarrow{(4,39)+4(cosd(-170),sind(-170)),%
(-4,21.5),(4,4)+4(cosd(170),sind(170))}
\arrowcorr{3}
\curvedarrow{(49,39)+4(cosd(-10),sind(-10)),%
(57,21.5),(49,4)+4(cosd(10),sind(10))}
\label[bc]{26}{15.5}{Up}\label[tr]{25}{13}{L}
\label[tl]{27}{13}{R}\label[cc]{26}{29}{Center}
\polygon{18(cosd(0),sind(0))+(100,21),%
18(cosd(60),sind(60))+(100,21),%
18(cosd(120),sind(120))+(100,21),%
18(cosd(180),sind(180))+(100,21),%
18(cosd(240),sind(240))+(100,21),%
18(cosd(300),sind(300))+(100,21)}
\line{18(cosd(0),sind(0))+(100,21),%
18(cosd(180),sind(180))+(100,21)}
\line{18(cosd(60),sind(60))+(100,21),%
18(cosd(240),sind(240))+(100,21)}
\line{18(cosd(120),sind(120))+(100,21),%
18(cosd(300),sind(300))+(100,21)}
\disky{1,(100,21),18(cosd(0),sind(0))+(100,21),%
18(cosd(60),sind(60))+(100,21),%
```

```
18(cosd(120),sind(120))+(100,21),%
18(cosd(180),sind(180))+(100,21),%
18(cosd(240),sind(240))+(100,21),%
18(cosd(300),sind(300))+(100,21)}
\putintobox{i}
\end{mfpic}
\begin{center}
\leavevmode
\box\boxi
\end{center}
```

A takhle dopadne obrázek:



Nejprve se příkazy `\polycurve` a `\line` nakreslí hrany. Pak se příkazy `\kruhy`, `\dkruh` („dělený“ kruh) a `\rozhod` vytvoří uzly. Příkazy `\curvedarrow` udělají oblouky se šipkami. Implicitně jsou osy šipek tečnami k těmto obloukům. Protože se mi zdálo někdy hezčí (v případě velké křivosti oblouku) šipku pootočit, zavedl jsem příkaz `\arrowcorr`, který říká, o kolik se má šipka pootočit od tečné polohy. Podobně je vytvořen šestiúhelník. Kód je poněkud dlouhý, ale to je dáno způsobem zadání souřadnic vrcholů pomocí funkcí `cosd` a `sind`. Díky tomu lze snadno celý šestiúhelník posunout nebo změnit jeho poloměr.

Na závěr několik poznámek. Součástí balíku `mujmfpic.zip` je dokumentace, kde lze najít popis dalších příkazů, a soubor s ukázkami. Celý balík je k dispozici např. na anonymním ftp `ftp.muni.cz` v adresáři `pub/tex/teware`.

Zejména při tvorbě „vyšeděných“ obrázků je třeba počítat s tím, že překlad METAFONTem může trvat hodně dlouho (i s `mf386` až několik minut; např. zpracování obrázků k tomuto článku na PC486 trvalo asi 65s). Dále pokud je „vyšeděná“ plocha příliš velká (čtverec $10\,\mathrm{cm} \times 10\,\mathrm{cm}$ je už hrozně moc), METAFONT zhavaruje (bohužel i `mf386`, nemám zkušenosti s UNIXem). Podobně se můžeme setkat s potížemi při převodu

generického fontu na `pk` font pomocí programu `gftopk`. Pod DOSem lze ve WEBovském zdroji, který mám k dispozici, nastavit jakýsi parametr maximálně na 65 534 (vím, že pod UNIXem toto omezení neplatí). Poněkud větší verzi mi poskytl kolega Horák, ale i ta zhavaruje např. pro čtverec se svislým šrafováním 1 mm od sebe o velikosti asi 7 cm × 7 cm. Často ale pomůže rozdělit obrázek do několika, použít umístění do boxu (příkaz `\putintobox`) a boxy přes sebe překrýt, což TEX zcela přesně dokáže (viz též následující ukázka).

A na úplný konec „lahůdka“ pro ty, kteří mají k dispozici program `dvips` a `ghostview` (nebo aspoň `ghostscript`) a barevný monitor. Snad vás výsledek následujícího zdrojového textu potěší.

```
\documentclass{report}
\usepackage{colordvi}
\input mlamfpic
\textwidth400pt\pagestyle{empty}
\begin{document}
\opengraphsfile{kvet}
\begin{mfpic}[9]{-22}{22}{-22}{22}
\shadespace=0pt\cdisk{(0,0),12}\putintobox{i}
\end{mfpic}
\begin{mfpic}[9]{-22}{22}{-22}{22}
\shadespace=0pt\cdisk{(0,0),4}\putintobox{ii}
\end{mfpic}
\begin{mfpic}[9]{-22}{22}{-22}{22}
\shadespace=0pt
\polardiscshade{(0,0),0,360,(0,0),0,360}{12+8*cosd(8t)}%
{12-8*cosd(8t)}\putintobox{iii}
\end{mfpic}
\begin{mfpic}[9]{-22}{22}{-22}{22}%
\shadespace=0pt\polarcurve{(0,0),0,360,12+8*cosd(8t)}
\polarcurve{(0,0),0,360,12-8*cosd(8t)}\putintobox{iv}
\end{mfpic}
\wd\boxi=0pt\wd\boxii=0pt\wd\boxiii=0pt
\centerline{\Green{\box\boxi}\Red{\box\boxii}%
\Yellow{\box\boxiii}\OrangeRed{\box\boxiv}}
\closegraphsfile
\end{document}
```

Pokud by program `dvips` zhavaroval (verze `dvips32` pro PC386 vyžadující koprocesor to zvládne), zmenšete velikost měřítka 9 na 3. Současně je vidět, že `mujmfpic` se snáší i s LaTeXem 2_ε.

Rychlejší tisk na devítijehličkových tiskárnách

Jan Bryscejn

Tento krátký článek se zabývá možnostmi tisku `.dvi` souborů na devítijehličkových (či 24jehličkových) tiskárnách s použitím Mattesova ovladače a několika dalších utilit.

Nejprve se podívejme na situaci, kterou chceme řešit. Je jasné, že pro výstup z TeXu, který má vůbec někdo číst, se hodí laserová nebo alespoň inkoustová tiskárna s minimálně 300 dpi. Aby byl výstup opravdu kvalitní, je dokonce i 600 dpi někdy málo. Na druhé straně příznivci TeXu v tomto programu vyrábějí kdeco, takže vzniká několik situací, kdy je vhodný tisk na jehličkové tiskárně.

Zejména se může stát, že není kvalitní tiskárna dostupná. Podle dalších okolností může být výtisk na jehličkové tiskárně již konečný, nebo požadujeme pouze koncept kvůli nastavení tiskárny; často nás zajímá celkový layout vytištěné strany a nikoliv podrobnosti. Konečně je možné, že dokument je pro naši soukromou potřebu a obětujeme kvalitu tisku rychlosti; to může nastat zejména u různých `.doc` souborů dodávaných s TeXem.

Pokud máme k dispozici 24jehlovou tiskárnu, nenastává žádný problém, neboť při rozlišení 180×180 dpi tiskne celou řádku na jeden průchod. Proto se tímto případem nebudeme dále zabývat. Problémy však nastanou při použití devítijehlové tiskárny, kdy se každý vodorovný proužek dat tiskne na tři, nebo dokonce na šest průběhů s mikrořádkováním mezi jednotlivými průběhy. To je jednak časově zcela neúnosné a kromě toho kvalita bývá horší než při 180×180 dpi 24jehličkové tiskárny, přestože rozlišení je 240×216 dpi.

Jak tedy tento problém vyřešit? Pokud máme k dispozici Mattesův ovladač `dvidot` a utilitu `makedot`, můžeme ovladač přinutit, aby tiskl