

METAPOST je silným nástrojem pro výrobu jednodušších postscriptových obrázků. Jazyk a principy METAPOSTu jsou takřka shodné s jeho předchůdcem METAFONTEM (viz Šedivý, P. a kol. 1997) liší se pouze v detailech a přidává několik užitečných funkcí, jež jsou předmětem tohoto článku.

• Instalace pod DOSem a Linuxem

!!! Instalace pod DOSem, EMT_EX, gjkt.zip

!!! Instalace pod Linuxem, tetex

• Sazba PostScriptu v T_EXu

Protože METAPOST produkuje postscriptové obrázky, měli bychom vědět, jak se tyto obrázky sázejí v T_EXu. Mějme tedy dokument `foo.tex` a obrázek `foo.eps` — přípona tohoto souboru může být jiná, důležité však je, aby byl ve formátu EPS (Encapsulated PostScript). Jedná se o podmnožinu formátu PS, má pouze jednu stranu a hlavně jsou na začátku souboru (nebo na konci) uvedeny rozměry obrázku (viz řádek začínající `%%BoundingBox:`).

Pro vkládání EPS obrázků slouží makra nazvaná `epsf`. Někde na začátku dokumentu je tedy třeba napsat `\input epsf` a v místě, kde chceme umístit obrázek pak `\epsfbox{foo.eps}`. Obrázek se v T_EXu chová jako obyčejný box s rozměry bounding boxu (viz výše).

Pokud potřebujeme obrázek zvětšit nebo zmenšit na určitý rozměr, můžeme *před* makrem `\epsfbox` ještě uvést např. `\epsfxsize=\hsize` (šířka obrázku je rovna šířce stránky) nebo `\epsfysize=7.5cm` (výška obrázku rovná 7,5 cm). Poměr stran obrázku přitom zůstane zachován, přiřazení `\epsfxsize` má přitom přednost před nastavením výšky obrázku. S obrázkem se *změní i velikost popisků*, které obsahuje. Takovéto nastavení rozměru funguje *jen pro jeden obrázek*, pro další obrázky musíme nastavení `\epsfxsize` a `\epsfysize` provést znovu.

• Schéma práce METAPOSTu

Nejprve musíme vytvořit vlastní zdrojový text obrázku, obvykle má tento soubor příponu `.mp`, tedy např. `foo.mp`. V jednom souboru může být více obrázků najednou, každý začíná `\beginpic` (číslo)); a končí `\endpic`. METAPOST nám se souboru `foo.mp` vyrobí postscriptové obrázky `foo.1`, `foo.2`, atd. podle toho, jak jsme očíslovali obrázky v makru `\beginpic`.

Pokud chceme výsledné obrázky včetně popisků (viz níže) ihned prohlížet (např. programem GhostView, `gv`), musíme splnit dvě podmínky. Za prvé je nutné na začátek souboru `foo.mp` napsat řádek `\prologues:=1`, který říká METAPOSTu, že má do výsledných postscriptových obrázků vložit definice fontů použitých v popiscích. Za druhé musíme mít tyto *postscriptové fonty* nainstalované a program `ghostscript` o nich musí vědět (dostupné jsou i PS verze standardních Computer Modern fontů i jejich počestěných variant, tzv. CS fonty – viz výše instalace).

Existuje však ještě druhá možnost využívající možnosti programu `dvips`, která tyto fonty nepotřebuje a stačí jí běžné T_EXové fonty ve formátu `pk`. Nejprve vysázíme obrázky v dokumentu `foo.tex` pomocí maker `epsf` (viz výše), pak jej přeložíme T_EXem a výsledný `dvi` soubor převedeme do PostScriptu příkazem

```
dvips -o foo.ps foo.dvi
```

Nesmíte přitom použít přepínač `-a`. Program `dvips` doplní potřebné znaky z fontů do popisků, rozdíl proti předchozímu způsobu je v tom, že fonty tentokrát nejsou vektorové (postscriptové), ale jsou uloženy na začátku souboru `foo.ps` v podobě bitmap, standardně s rozlišením 600 dpi. Postscriptový soubor tím neztrácí nezávislost na rozlišení, stále jej můžeme libovolně škálovat a tisknout na zařízeních s libovolným rozlišením. Pokud však budeme tisknout s rozlišením např. 2400 dpi, popisky budou správně velké, ale jejich kvalita bude odpovídat pouhým 600 dpi.

Výhodou je, že tento postup bychom stejně nejspíše nakonec prodělali, neboť konečné T_EXové dokumenty včetně vysázených obrázků je zvykem šířit právě v postscriptovém formátu.

• Základní jednotka

Základní jednotkou vzdálenosti je postscriptový bod `bp=1in/72`. Samozřejmě jsou definované i ostatní obvyklé jednotky jako `cm`, `mm`, `pt`.

• Transformace obrázků

Díky vlastnostem jazyka PostScript je nyní možné provádět všechny afinní transformace i s celými obrázky, nejen s body, cestami nebo s tvary pera. Tuto vlastnost oceníme zejména při tvorbě popisků, které se jako obrázky chovají (viz níže).

• Barvy a stupně šedi

Novinkou oproti METAFONTu je možnost využití stupňů šedi a barev. Barva je trojice čísel (r, g, b) , kterou deklarujeme primitivem `color`. Předdefinované barvy jsou `black=(0,0,0)`, `white=(1,1,1)`, `red`, `green`, `blue`. S barvami zacházíme podobně jako s vektory (typ `pair`). Složky barvy zjistíme primitivy `redpart`, `greenpart`, `bluepart` (obdoba `xpart`, `ypart` pro vektory). Před výstupem se všechny složky barev zaokrouhlí do intervalu $\langle 0; 1 \rangle$, předchozí *výpočty* (např. sčítání, odčítání barev) však probíhají v celém rozsahu.

Chceme-li tedy kreslit jinou barvou, než standardní černou, musíme za `fill` nebo `draw` uvést `withcolor` \langle barva \rangle . Např. vyplnění provedeme pomocí

`fill` \langle uzavřená cesta \rangle `withcolor` \langle barva \rangle

V případě, že budeme kreslit více cest stejnou barvou (vyberme např. $(0.4, 0.5, 0.9)$), vyplatí se před vlastním kreslením použít makro `drawoptions(withcolor (0.4,0.5,0.9))`; Nastavení platí až do dalšího `drawoptions` nebo do konce obrázku určeného `endpic`. Tento trik nám také pomůže tehdy, pokud kreslíme pomocí vlastních maker, která nemají barvu mezi svými parametry.

• Cesty – průsečíky, tvary ukončení a navázání

METAPOST zavádí několik nových maker pro práci s cestami a jejich průsečíky: `buildcycle`, `cutbefore` a `cutafter`.

Makro `buildcycle(p1, p2, p3, ..., pk)` vybere průsečíky mezi cestami p_i a p_{i+1} a vytvoří uzavřenou cestu, kterou lze vyplnit.

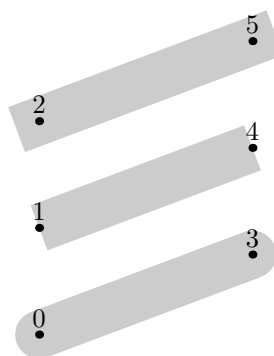
p_1 `cutbefore` p_2 uřízne cestu p_1 před průsečíkem s cestou p_2 . Je definováno jako

`subpath (xpart(p_1 intersectiontimes p_2), length p_1) of p_1`

a navíc do cesty pojmenované `cuttings` uloží uříznutou část. Makro p_1 `cutafter` p_2 dělá totéž s `reverse` p_1 .

Parametr `linecap` určuje tvar ukončení cesty. Může nabývat hodnot `rounded`, `butt`, `squared` — konkrétní vzhled viz následující příklad:

```
beginfig(4);
for i=0 upto 2:
z[i]=(0,40i); z[i+3]-z[i]=(80,30);
endfor
pickup pencircle scaled 18;
def gray = withcolor .8white enddef;
draw z0..z3 gray;
linecap:=butt; draw z1..z4 gray;
linecap:=squared; draw z2..z5 gray;
dotlabels.top(0,1,2,3,4,5);
endfig; linecap:=rounded;
```

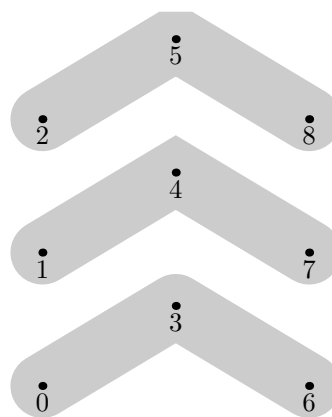


Parametr `linejoin` určuje způsob navázání cest. Tři možné hodnoty jsou `rounded`, `beveled`, `mitered`. Pokud cesty svírají úhel blízký 360° , může u posledního typu dojít k nežádoucímu prodloužení. Toto je omezeno parametrem `miterlimit`, který určuje maximální poměr prodloužení ku šířce čáry. Standardní hodnota je `miterlimit:=10`;

```

beginfig(5);
for i=0 upto 2:
z[i]=(0,50i); z[i+3]-z[i]=(50,40);
z[i+6]-z[i]=(100,0);
endfor
pickup pencircle scaled 24;
def gray = withcolor .8white enddef;
draw z0--z3--z6 gray;
linejoin:=mitered; draw z1..z4--z7 gray;
linejoin:=beveled; draw z2..z5--z8 gray;
dotlabels.bot(0,1,2,3,4,5,6,7,8);
endfig; linejoin:=rounded;

```



• Čárkované čáry, operátory arclength, arctime

Čárkované čáry je možné kreslit makrem

```
draw <cesta> dashed <vzor>
```

<vzor> se chová jako obrázek. Existují jednoduché předdefinované vzory `evenly`, `withdots`, ale lze nadefinovat i složitější vlastní pomocí makra

```
dashpattern(on 3mm off 6mm on 6mm)
```

Čárky mají vždy stejnou délku, i když je cesta extrémně zakřivená. Nevýhodou těchto čárkovaných čar však je, že nekončí čárkou v místě, kde končí cesta.

V tomto směru nám mohou být velmi užitečné jsou operátory `arclength p` a `arctime a of p`, které se vztahují k délce oblouku, nikoli k času, kterým je cesta `p` parametrizována. Jednoduché makro na kreslení čárkované čáry pak může vypadat např. takto

```

def drawdash(expr p,n)=
for t=0 upto n:
draw subpath(arctime(arclength(p)*3t/(3n+2)) of p,
arctime(arclength(p)*(3t+2)/(3n+2)) of p) of p
endfor
enddef

```

• Primitivy `cull` a `clip`

V METAPOSTu zcela chybí primitiv `cull`, který v METAFONTu sloužil k mazání bodů, jež byly několikrát překresleny. Velmi častou aplikací bylo vyšrafování určité oblasti. METAPOST však nemůže počítat, kolikrát byl každý bod překreslen, neboť vždy pracuje s vektorovým, nikoli rastrovým obrázkem. Navíc obecnou vlastností PostScriptu je, že později kreslené a vyplňované tvary zcela překrývají dřívější.

Jako náhradu zavádí METAPOST primitiv `clip`, který z obrázku nechá pouze část uvnitř cesty.

```
clip <obrázek> to <cesta>
```

Vhodnou manipulací s obrázkem a tímto primitivem (připomínáme, že aktuální obrázek se označuje `currentpicture` a že funguje primitiv `addto`), však můžeme dosáhnout stejných výsledků jako v METAFONTu.

• Začlenění textu do grafiky

Asi největší výhodou MetaPostu je možnost začlenění textu vysázeného T_EXem do obrázku, bez toho, že bychom museli několikrát zkoušet přesné umístění popisku. K těmto účelům slouží makro zvané `label`.

```
label<umístění>(<řetězec nebo obrázek>, <bod>)
```

Umístění popisku vzhledem k bodu buď nemusíme specifikovat vůbec, potom bod znamená střed popisku, anebo můžeme za *tečkou* uvést jednu z následujících možností: `rt`, `lft`, `top`, `bot`, `urt`, `ulft`, `llft`, `lrt` (vpravo, vlevo, nahoře, dole, vpravo nahoře, ...). Celá konstrukce pak může vypadat např. takto:

```
label.lft("popisek bodu z1", z1);
```

Vzhled popisu lze ještě ovlivnit nastavením parametrů `labeloffset` (vzdálenost popisu od bodu), `defaultfont` (písmo pro vysázení řetězce, pokud je parametrem `label` obrázek, nemá na výsledek vliv) a `defaultscale` (zvětšení/zmenšení popisu). Standardní hodnoty těchto parametrů jsou `labeloffset:=3bp`; `defaultfont:="cmr10"`; `defaultscale:=1`;

Místo řetězce můžeme v makru `label` přímo přistupovat ke znakům fontu `defaultfont` pomocí `char` ⟨číslo⟩ (v \TeX u je obdobou primitiv `\char`).

Makro `label` má ještě dvě varianty — `dotlabel` zobrazí popis i s tečkou v daném bodě. Pokud nechceme popisek zobrazit, ale raději bychom jej získali v podobě obrázku `v`, použijeme `v:=thelabel ...` (`v` musí být samozřejmě předtím deklarováno jako obrázek — `picture v`;). Do aktuálního obrázku jej můžeme kdykoliv přidat příkazem `addto currentpicture also v`;

Složitější popisky, ve kterých jsou potřeba různé fonty, matematické rovnice je možné vysázet \TeX em. Použijeme k tomu primitivy

`btex` ⟨příkazy \TeX u⟩ `etex`

Výsledkem této konstrukce je *obrázek*. Uveďme si jeden příklad — všimněte si přitom použití `\displaystyle` namísto dvojice `$$` (\TeX musí totiž zůstat v horizontálním módu) a také toho, že obrázek (vysázenou rovnici) můžeme libovolně transformovat.

```
label.lft(btex $\displaystyle y{2\over 1+\cos x}$ etex rotated 90, (10mm,5mm));
```

Pokud bychom chtěli v příkazech používat nějaká vlastní \TeX ová makra, uvedeme je na začátku souboru mezi `verbatimtex` a `etex`. Typickým příkazem tudíž bývá `\input makra.tex`.

- **Určení rozměru obrázku – makro `bbox`**

Aby mohl být popisek (obrázek) správně umístěn, potřebuje makro `label` znát jeho okraje. K tomu slouží makro `bbox`⟨obrázek⟩, které je ekvivalentní

`(llcorner v--lrcorner v--urcorner v--ulcorner v--cycle)`

Výsledkem je tedy uzavřená cesta těsně obepínající obrázek. Navíc makro přidává okraj, jehož šířku udává `bboxmargin` (standardní hodnota je `bboxmargin:=2bp`;). Makro lze využít např. pro sazbu orámečkových popisků — nejprve si popisek uložíme do obrázku `v:=btex x^2 etex`; , zobrazíme jej makrem `label(v,z2)`; a nakonec přidáme rámeček `draw bbox v shifted z2`;

- **Další nová makra**

V `METAPOSTu` přibyl nové makro kreslící šipku na konci cesty `drawarrow` ⟨cesta⟩. šipka je vyplněná a má poněkud složitější tvar, aby vypadala pěkně i na zakřivené cestě. Dvě proměnné `ahlength` a `ahangle` určují velikost a vrcholový úhel šipky, počáteční hodnoty definované v souboru `plain.mp` jsou `ahlength:=4bp`; `ahangle:=40`. Makro `drawdblarrow` ⟨cesta⟩ nakreslí cestu s šipkami na obou koncích.

O makru `drawoptions` jsme se již zmiňovali — slouží k uložení voleb pro kreslení pomocí `draw`, `fill`. Kromě `withcolor` funguje i s `dashed`, `withpen`.

- **Převod čísla na řetězec – makro `format`**

- **`boxes.mp` – kreslení diagramů**

- **Kreslení grafů `METAPOSTem`**

- **Literatura a odkazy**

Hobby, J. D.: A User's Manual for Metapost. postscriptový soubor `mpman.ps` lze nalézt v distribucích \TeX u anebo je možno jej stáhnout např. ze serveru CSTUGu <http://www.cstug.cz>

Šedivý, P. a kol.: Kreslíme `METAFontem`. Hradec Králové, 1997, postscriptovou verzi možno stáhnout z <http://sirrah.troja.mff.cuni.cz/~mira/kreslime/>