

# Zpravodaj Československého sdružení uživatelů TeXu

---

Miroslava Krátká  
METAPOST a mfpic - druhá část

*Zpravodaj Československého sdružení uživatelů TeXu*, Vol. 11 (2001), No. 1-3, 66–135

Persistent URL: <http://dml.cz/dmlcz/150210>

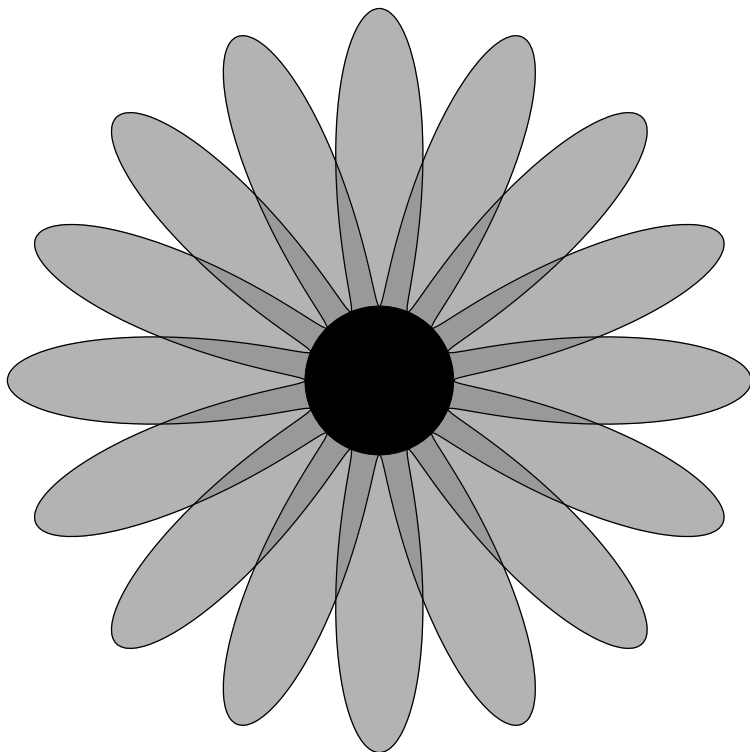
## Terms of use:

© Československé sdružení uživatelů TeXu, 2001

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* <http://dml.cz>



# Obsah

---

<b>1</b>	<b>Balíky maker pro METAPOST</b>	<b>68</b>
<b>2</b>	<b>Historie mfpic</b>	<b>68</b>
<b>3</b>	<b>Zdrojový soubor</b>	<b>68</b>
<b>4</b>	<b>Volby mfpic</b>	<b>71</b>
<b>5</b>	<b>Parametry mfpic</b>	<b>72</b>
<b>6</b>	<b>Barvy mfpic</b>	<b>73</b>
<b>7</b>	<b>Jednoduché útvary</b>	<b>74</b>
7.1	Souřadné osy	78
<b>8</b>	<b>Polární souřadnice</b>	<b>78</b>
<b>9</b>	<b>Uložení objektu</b>	<b>79</b>
<b>10</b>	<b>Prefixová makra</b>	<b>80</b>
10.1	Kreslení	80
10.2	Výplně	83
10.3	Šipky	86
10.4	Změna orientace křivky	87
10.5	Uzavírání křivek	87
10.6	Napojení křivek	89
10.7	Afinní transformace	89
10.8	Používání afinních transformací	90
10.9	Pořadí prefixových maker	94
<b>11</b>	<b>Rendrování</b>	<b>95</b>
<b>12</b>	<b>Funkce</b>	<b>96</b>
<b>13</b>	<b>Kreslení dat z externího souboru</b>	<b>100</b>
<b>14</b>	<b>Popisy obrázků</b>	<b>103</b>
<b>15</b>	<b>Pole křivek</b>	<b>107</b>
<b>16</b>	<b>Definice příkazů</b>	<b>107</b>
<b>17</b>	<b>Složitější obrázky</b>	<b>107</b>
<b>18</b>	<b>Použití mfpic při tvorbě obrázků</b>	<b>113</b>
18.1	Grafické znázornění množin	114
18.2	Grafy funkcí	116
18.3	Určitý integrál a jeho geometrické aplikace	120
18.4	Množiny bodů dané vlastnosti	130
<b>Summary: METAPOST and mfpic—the second part</b>		<b>135</b>

---

## 1. Balíky maker pro METAPOST

První část práce ukazuje nemalé možnosti METAPOSTu při kreslení obrázků. Chceme-li ovšem nakreslit náročnější obrázek, nestačí jen prolistovat manuálem; musíme se naučit pracovat s novým programovacím jazykem. To mnohé, zejména ty méně zvědavé, odradí. A právě pro ně se objevují balíky maker pro METAPOST.

Takovými balíky jsou `mfpic`, kterým se v dalším textu budeme podrobněji zabývat (<http://comp.uark.edu/~luecking/tex/mfpic.html>), a `feynmf`, jehož autorem je Thorsten Ohl (<ftp://ftp.cstug.cz/pub/CTAN/macros/latex/contrib/supported/feynmf>). V roce 1989 byl vytvořen soubor maker `feynman.mf` pro kreslení Feynmanových diagramů. O pět let později, tedy v roce 1994, se autor inspiroval balíkem `mfpic` (zdrojový kód se zapisoval do zdrojového souboru  $\LaTeX$ u). Vzniklý `feynmf` využíval pro kreslení obrázků buď METAFONT, nebo METAPOST.

Zcela odlišná je grafická nadstavba pro METAPOST nazvaná `metagraf`. Jde o aplikaci napsanou v jazyce Java (je třeba Java2). Kreslí se pomocí menu a myši (<http://w3.mecanica.upm.es/metapost/metagraf.php>).

## 2. Historie mfpic

Během roku 1992 začal Tom Leathrum pracovat na tvorbě `mfpic`. Tento balík maker využívá pro kreslení METAFONT, přičemž zdrojový kód se zapisuje do vstupního souboru  $\TeX$ u a užití příkazů samotného METAFONTu je pro uživatele skryto. V roce 1996 vývoj převzal Daniel H. Luecking. Od verze 0.3.0 z roku 1998 již můžeme využívat i METAPOST. Aktuální verzi k dubnu 2001 je verze 0.4.05. První zmínka o `mfpic` v češtině se objevila v roce 1994 v druhém čísle Zpravodaje  $\zeta$ TUGu zmíněného roku ([5, str. 87]). Autor se věnoval verzi `mfpic` 0.2, kterou upravil, a verzi 0.2.5.

## 3. Zdrojový soubor

Makra `mfpic` lze použít ve formátech plain,  $\LaTeX$  i  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\TeX$ . Než začneme tvořit naše obrázky, potřebujeme mít nainstalované následující soubory a balíky (prvním předpokladem je samozřejmě  $\TeX$  a METAPOST, respektive METAFONT):

- `grafbase.mp` a `dvipsnam.mp` v adresáři prohledávaném METAPOSTem (například aktuální adresář), respektive `grafbase.mf` v adresáři prohledávaném METAFONTem,
- `mfpic.tex` a `mfpic.sty` v adresáři, ve kterém hledá  $\TeX$  (například aktuální adresář),

- `epsf.tex` pro plain  $\text{T}_\text{E}\text{X}$ ,  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_\text{E}\text{X}$  a  $\text{L}^\text{A}\text{T}_\text{E}\text{X}2.09$ ,
- `supp-pdf.tex` a `supp-mis.tex` pro `pdfTEX`,
- `graphics` nebo `graphicx` pro  $\text{L}^\text{A}\text{T}_\text{E}\text{X}2_\epsilon$  a pro `pdfLATEX` s výstupem do pdf; tento případ vyžaduje tři soubory, a to `pdftex.def`, `supp-pdf.tex`, `supp-mis.tex`.

Jestliže žádný z uvedených souborů a balíčků nenačteme (nebo tak učiníme až po načtení balíku `mfpic`), `mfpic` to provede sám. Například pro  $\text{L}^\text{A}\text{T}_\text{E}\text{X}$  načítá balík `graphics`. Toto vše je nezbytné pro vkládání námi vytvořených obrázků, jež je prováděno takto:

- `\epsfbox {jméno_souboru}` v plain  $\text{T}_\text{E}\text{X}$ u,  $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_\text{E}\text{X}$ u a  $\text{L}^\text{A}\text{T}_\text{E}\text{X}$ u 2.09,
- `\convertMPtoPDF {jméno_souboru}{1}{1}` v `pdfTEX`u,
- `\includegraphics {jméno_souboru}` v  $\text{L}^\text{A}\text{T}_\text{E}\text{X}2_\epsilon$  a `pdfLATEX`u.

V dalším výkladu budeme pojmem  $\text{L}^\text{A}\text{T}_\text{E}\text{X}$  myslet  $\text{L}^\text{A}\text{T}_\text{E}\text{X}2_\epsilon$ . Zaměříme se na `mfpic` s využitím `METAPOST`u.

Jak víme z předchozích odstavců, `METAPOST` vytváří soubory s příponou `.číslo`, a proto je v `pdfLATEX`u dále třeba přidat příkaz

```
\DeclareGraphicsRule {*} {mps} {*} {*} .
```

Každý soubor využívající `mfpic` s `METAPOST`em má v  $\text{L}^\text{A}\text{T}_\text{E}\text{X}$ u následující strukturu:

```
\documentclass{article}
\usepackage[metapost]{mfpic} %% pokud neuvedeme nepovinný para-
                             %% metr, bude použit pro tvorbu
                             %% obrázků Metafont
%\usepackage{mfpic} %% tyto dva řádky jsou ekvivalentní s řád-
%\usemetapost %% kem předcházejícím

\begin{document}

\opengraphsfile{obrazek}

%libovolný zdrojový kód

\begin{mfpic}[a][b]{c}{d}{e}{f}
%příkazy popisující obrázek - výsledkem je obrazek.1
\end{mfpic}

%okolí mfpic lze zapisovat také takto:

\mfpic[g][h]{i}{j}{k}{l}
%příkazy popisující obrázek - výsledkem je obrazek.2
\end{mfpic}
```

```

\closegraphsfile

\opengraphsfile{picture}

\mfpic[m][n]{o}{p}{q}{r}
%příkazy popisující obrázek - výsledkem je picture.1
\endmfpic

```

```

\closegraphsfile

```

```

\end{document}

```

Obdobný zdrojový soubor v plain T<sub>E</sub>Xu:

```

\input mfpic.tex
\usemetapost

```

```

\opengraphsfile{obrazek}

```

```

\mfpic[a][b]{c}{d}{e}{f}
%příkazy popisující obrázek - výsledkem je obrazek.1
\endmfpic

```

```

\closegraphsfile

```

```

\bye

```

Příkazem

```

\opengraphsfile{název_souboru}

```

otevíráme soubor, do kterého se zapisují příkazy METAPOSTu, příkaz

```

\closegraphsfile

```

tento soubor zavírá (tyto příkazy by měly být používány pouze v udaném tvaru, to jest ne jako okolí `\begin{okoli} ... \end{okoli}` v L<sup>A</sup>T<sub>E</sub>Xu). Příkazy

```

\mfpic [měřítkoosa_x] [měřítkoosa_y] {xmin} {xmax} {ymin} {ymax}

```

```

...

```

```

\endmfpic

```

```

\begin{mfpic}[měřítkoosa_x][měřítkoosa_y]{xmin}{xmax}{ymin}{ymax}

```

```

...

```

```

\end{mfpic} (pouze v LATEXu)

```

uzavírají popis jednoho obrázku. Měřítka musí být zadáno alespoň na jedné ose; je-li zadáno pouze jedno, je měřítko shodné pro osu  $x$  i  $y$ . Čísla  $x_{\min}$ ,  $x_{\max}$  určují rozsah osy  $x$ , čísla  $y_{\min}$ ,  $y_{\max}$  určují rozsah osy  $y$ .

Nyní jsou dvě možnosti způsobu zpracování vstupního souboru:

Po překladu vstupního souboru  $\LaTeX$ em (respektive plain  $\TeX$ em) vzniknou soubory `obrazek.mp` a `picture.mp`. Tyto soubory přeložíme programem `mpost` (viz první část práce) a poté znovu spustíme  $\LaTeX$  (respektive  $\TeX$ ). Výsledkem je soubor s příponou `.dvi`. Nejsou-li při jeho prohlížení viditelné obrázky (například `xdvi` na novějších verzích operačního systému Linux obrázky zobrazuje), vytvoříme příslušným programem (`dvips`) postscriptový soubor a prohlédneme si jej například pomocí `Ghostview`.

Použijeme-li `pdf $\LaTeX$`  (respektive `pdf $\TeX$` ), vznikne soubor s příponou `.pdf`, který si prohlédneme například prohlížečem `Acrobat Reader`.

## 4. Volby `mfpic`

Již v předchozím odstavci jsme se seznámili s jednou volbou balíku `mfpic`, a to `metapost`. Jak bylo vidět z ukázky, tyto volby se zapisují buďto jako nepovinné parametry u  $\LaTeX$ ovského příkazu `\usepackage`, nebo ekvivalentními příkazy.

Můžeme použít následující volby:

- `metapost`, `\usemetapost` – popisy obrázků vytvářených pomocí `mfpic` jsou zapisovány do souboru s příponou `.mp` a tento soubor je zpracován `METAPOST`em; příkaz musí být zapsán před otevřením souboru `.mp`, to jest před příkazem `\opengraphsfile`,
- `metafont`, `\usemetafont` – popisy obrázků jsou zadávány do souboru s příponou `.mf`, tento soubor je zpracován `METAFONT`em; příkaz musí být zapsán před otevřením souboru `.mf`, to jest před příkazem `\opengraphsfile`; tato volba je přednastavena (chceme-li tedy používat `METAFONT`, není nutno volbu udávat),
- `mplabels`, `\usemplabels`, `\nomplabels` – tato volba ovlivňuje zpracování popisek u příkazu `\tlabel`, popisky jsou zpracovány `METAPOST`em; příkaz musí být uveden až za příkazem `\usemetapost` ; používáme pouze s následující volbou,
- `truebbox`, `\usettruebbox`, `\nottruebbox` – zajistí, aby `METAPOST` určil bounding box obrázku včetně textu; tuto volbu využíváme zároveň s volbou `mplabels`,
- `clip`, `\clipmfpic`, `\noclipmfpic` – odstraní části obrázku přesahující obdélník udaný rozměry u `\mfpic`; lze použít jak u `METAFONT`u, tak u `METAPOST`u,
- `centeredcaptions`, `\usecenteredcaptions`, `\nocenteredcaptions` – zkrácené řádky textu pod obrázkem vytvořeného příkazem `\tcaption` jsou umístěny do středu, neboli vycentrovány podle šířky obrázku,
- `debug`, `\mfpicdebugtrue`, `\mfpicdebugfalse` – sdělí nám více informací o průběhu zpracování vstupního souboru; tyto informace jsou zapisovány do souboru s příponou `.log` a v některých případech na obrazovku.

Pro naši práci jsou ideální následující dvě kombinace:

```
\usepackage [metapost, mplabels, truebbox] {mfpic} ,  
\usepackage [metapost, mplabels, truebbox, clip] {mfpic} .
```

Volba `mplabels` zajistí, aby popisky zpracovával METAPOST; volba `truebbox` zajistí, aby METAPOST určil přesný bounding box obrázku a tento předal T<sub>E</sub>Xu (a tedy nevzniknou problémy například při obtékání obrázku); volba `clip` způsobí ořezání výsledného obrázku na rozměr námi zadaný (tudíž zamezíme zasahování okolního textu do obrázku). Všechny další kombinace různých voleb `mfpic` mohou dát zcela nesprávné výsledky.

## 5. Parametry `mfpic`

V makrech `mfpic` se samozřejmě objevuje mnoho parametrů, jejichž změnami můžeme dosáhnout různých, pro nás příjemných, efektů. U většiny maker, o kterých se zmiňuji v dalších odstavcích, je řečeno, jaké parametry ovlivňují jejich výsledky.

Některé parametry jsou uloženy jako dimenze T<sub>E</sub>Xu; jiné jsou uloženy METAPOSTem, tedy jejich případné změny uvnitř okolí `\mfpic ... \endmfpic` mají pouze lokální význam.

Významné pro nás mohou být následující parametry (veškerá zde zmíněná makra jsou podrobněji popisována v dalším textu).

Parametry T<sub>E</sub>Xu:

- `\mfpicunit` – představuje základní jednotku délky; přednastavena je na hodnotu 1 pt; všechny *x*-ové a *y*-ové souřadnice jsou jí v makrech `mfpic` násobeny (proměnná typu dimenze),
- `\pointsize` – uchovává průměr kružnice používané v makru `\point`, dále průměr symbolů v makrech `\plotsymbol` a `\plot`; přednastavena je hodnota 2 pt (proměnná typu dimenze),
- `\pointfilltrue`, `\pointfillfalse` – logická proměnná určující, bude-li kružnice kreslená příkazem `\point` vyplněna, či nikoli,
- `\headlen` – dimenze udávající délku šipky při použití makra `\arrow`; přednastavena je délka 3 pt,
- `\axisheadlen` – dimenze, která uchovává délku šipky při využití makra `\axes`, `\xaxis`, `\yaxis`; přednastavena je délka 5 pt,
- `\dashlen` – dimenze určující délku čárek v makru `\dashed`; přednastavena je délka 4 pt,
- `\dashspace` – dimenze udávající velikost mezer používaných v makru `\dashed`; přednastavena je mezera 4 pt,
- `\dashlineset` – makro, které nastaví standardní hodnoty pro `\dashlen` a `\dashspace`, to znamená obě hodnoty na 4 pt,
- `\dotlineset` – makro nastavující hodnotu pro `\dashlen` na 1 pt a hodnotu `\dashspace` na 2 pt,



- `\symbolspace` – dimenze určující velikost mezery mezi symboly v makru `\plot`; přednastavena je mezera 5 pt,
- `\hashlen` – dimenze uchováající délku čárek používaných v makrech `\xmarks` a `\ymarks`; přednastaveny jsou délky 4 pt,
- `\dotsize` – dimenze určující velikost kruhů v makru, `\dotted`; přednastaven je průměr 0,5 pt,
- `\dotsspace` – dimenze udávající mezery mezi středy kruhů vykreslujících se v makru `\dotted`; přednastavena je mezera o velikosti 3 pt,
- `\polkadotsspace` – dimenze nastavující velikost mezer mezi středy kruhů používaných v makru `\polkadot`; přednastavena je mezera 10 pt,
- `\hatchspace` – dimenze uchováající vzdálenost rovnoběžek používaných v makru `\hatch`; přednastavena je vzdálenost 3 pt,
- `\mfpicheight` – dimenze, v níž je uložena výška obrázku vytvořeného v okolí `\mfpic ... \endmfpic`,
- `\mfpicwidth` – dimenze, v níž je uložena šířka obrázku vytvořeného okolím `\mfpic ... \endmfpic`.

Poslední dvě dimenze (výška a šířka obrázku) jsou pro nás praktické v případě, že bychom chtěli „šidit“ při umísťování obrázku. Standardně jsou vypočítávány z údajů zadaných v příkazu `\mfpic` v úvodu každého obrázku.

Chceme-li některou z dimenzí  $\TeX$ u změnit, provedeme to obvyklým způsobem, to znamená například `\mfpicunit=3pt` (v  $\LaTeX$ u můžeme také příkazem `\setlength{\mfpicunit}{3pt}`).

Parametry `METAPOST`u (neboli  $\TeX$ ovská makra nastavující interní parametry `METAPOST`u, jejichž přesné názvy pro nás nejsou podstatné):

- `\pen {tloušťka}`, `\drawpen {tloušťka}` – nastaví tloušťku pera pro kreslení; přednastavena je hodnota 0,5 pt,
- `\hatchwd {tloušťka}` – určuje tloušťku čar používaných při šrafování; přednastavena je hodnota 0,5 pt,
- `\polkadotwd {průměr}` – přiřazuje průměr kruhům využívaným v makru `\polkadot`; přednastavena je hodnota 5 pt,
- `\headshape {poměr} {napětí} {vybarvení}` – toto makro určuje tvar šipky používané v makrech `\arrow`, `\axes`, `\xaxis`, `\yaxis`; *poměr* je poměr šířky šipky k její délce, *napětí* představuje napětí Bézierovy křivky, kterou je šipka vykreslena, a *vybarvení* značí, či je šipka otevřená (hodnota `false`) nebo uzavřená a vybarvená; přednastaveny jsou hodnoty 1, 1, `false`.

## 6. Barvy `mfpic`

V `METAPOST`ových makrech pro `mfpic`, to jest v souboru `grafbase.mp`, se setkáváme se čtyřmi barvami – `drawcolor`, `fillcolor`, `hatchcolor`, `headcolor`.

Barvou `drawcolor` jsou vykreslovány křivky, barva `fillcolor` je využívána pro výplně, `hatchcolor` pro šrafování a `headcolor` pro kreslení šipek. Všechny tyto barvy jsou přednastaveny na černou barvu.

Nejjednodušší způsob, jak některou z výše uvedených barev změnit, je použití odpovídajícího příkazu:

```
\drawcolor {barva} ,  
\fillcolor {barva} ,  
\hatchcolor {barva} ,  
\headcolor {barva} .
```

Parametrem *barva* může být:

- předdefinovaná barva – `black`, `white`, `red`, `green`, `blue`, `cyan`, `magenta`, `yellow` (tzn. černá, bílá, červená, zelená, modrá, tyrkysová, fialová, žlutá),
- barva vyjádřená modelem `rgb` – `rgb (a, b, c)`, kde *a*, *b*, *c* jsou čísla z intervalu  $\langle 0, 1 \rangle$ ,
- barva vyjádřená modelem `cmyp` – `cmyp (a, b, c, d)`, kde *a*, *b*, *c*, *d* jsou čísla z intervalu  $\langle 0, 1 \rangle$ , převádí barvu udanou modelem `cmyp` na model `rgb`,
- barva vyjádřená modelem `RGB` – `RGB (a, b, c)`, kde *a*, *b*, *c* jsou z intervalu  $\langle 0, 255 \rangle$ , převádí barvu v modelu `RGB` na model `rgb`,
- barva vyjádřená modelem `gray` – `gray (a)`, kde *a* je z intervalu  $\langle 0, 1 \rangle$ , odpovídá barvě  $a * \text{white}$ ,
- barva námi předdefinovaná (model `named`) – můžeme využít barvy definované v souboru `dvipsnam.mp` nebo barvy jiné, námi nadefinované pomocí příkazu

```
\mfpdefinecolor {jméno_barvy} {model} {barva} ,
```

kde *jméno\_barvy* je libovolný námi zvolený název pro barvu, *model* je jeden z modelů `rgb`, `RGB`, `cmyp`, `gray`, `named` a *barva* je vyjádření požadované barvy v použitém modelu. Použijeme-li model `named`, pouze přejmenujeme danou *barvu*.

Druhou možností změny barvy jsou příkazy:

```
\drawcolor [model] {barva} ,  
\fillcolor [model] {barva} ,  
\hatchcolor [model] {barva} ,  
\headcolor [model] {barva} ,
```

kde *model* může být `rgb`, `cmyp`, `RGB`, `named`, nebo `gray` a *barva* je „hodnota“ barvy v příslušném modelu.

## 7. Jednoduché útvary

Příkaz

```
\pointdef {název} (x, y)
```

umožňuje definovat název pro bod a jeho souřadnice, kde *název* je námi zvolený název (neobsahující zpětné lomítko) pro bod o souřadnicích  $(x, y)$ . Nadefinujeme-li například `\pointdef{M}(3,7)`, `mfpic` bude příkaz `\M` chápat jako zadání souřadnic  $(3,7)$  a `\Mx`, popř. `\My`, bude expandovat na 3, popř. 7.

Makro

```
\point [průměr] {( $x_1, y_1$ ), ( $x_2, y_2$ ), ... }
```

vykreslí kruhy se středy  $(x_1, y_1), (x_2, y_2), \dots$  a průměrem odpovídajícím hodnotě proměnné `\pointsize` (přednastavené na velikost 2 pt). Nepovinný parametr *průměr* nám umožňuje přednastavenou hodnotu průměru kruhů jednoduše upravovat. Chceme-li body označovat pouze kružnicemi, využijeme příkazu `\pointfillfalse`; standardní nastavení, `\pointfilltrue`, vyplňuje kruhy barvou `fillcolor`.

Příkaz

```
\grid {mezera_vertikálních_rovnoběžek,  
      mezera_horizontálních_rovnoběžek}
```

vytvoří systém kruhů, které umístí na průsečíky horizontálních a vertikálních rovnoběžek. Vzdálenost těchto rovnoběžek udáme dvěma povinnými parametry, a to *mezera\_vertikálních\_rovnoběžek* a *mezera\_horizontálních\_rovnoběžek*; tyto parametry zadáváme bezrozměrné, tedy pouze čísla bez jednotky. Kruhy odpovídají interní proměnné `onedot` typu `picture`, což je vnitřek kruhu o průměru 0,5 pt.

Nejsme-li zastánci označování bodů kruhy či kružnicemi, zalíbí se nám makro

```
\plotsymbol [průměr] {symbol} {( $x_1, y_1$ ), ( $x_2, y_2$ ), ... }
```

vykreslující *symbols* se středy o souřadnicích  $(x_1, y_1), (x_2, y_2), \dots$ . K dispozici jsou tyto *symbols*:

- `Circle` – kružnice,
- `Diamond` – kosočtverec,
- `Square` – čtverec,
- `Triangle` – trojúhelník,
- `SolidCircle` – kruh,
- `SolidDiamond` – vyplněný kosočtverec,
- `SolidSquare` – vyplněný čtverec,
- `SolidTriangle` – vyplněný trojúhelník,
- `Cross` – křížek,
- `Plus` – plus,
- `Star` – hvězda.

Velikost symbolů koresponduje s hodnotou `\pointsize` (=2 pt) a lze ji upravit nepovinným parametrem *průměr*. Vyplněné symboly mají barvu odpovídající `fillcolor`.

Lomenou čáru s vrcholy v bodech  $(x_1, y_1), (x_2, y_2), \dots$  nakreslíme příkazy

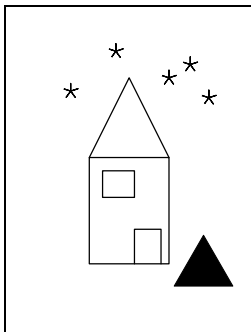
```
\polyline {( $x_1, y_1$ ), ( $x_2, y_2$ ), ... } ,  
\lines {( $x_1, y_1$ ), ( $x_2, y_2$ ), ... } .
```

Makro

`\polygon {(x_1, y_1), (x_2, y_2), \dots}`  
 vykreslí uzavřený polygon s vrcholy v bodech  $(x_1, y_1), (x_2, y_2), \dots$

Příklad

`\rect {(x_1, y_1), (x_2, y_2)}`  
 vytvoří obdélník, kde  $(x_1, y_1), (x_2, y_2)$  jsou protilehlé vrcholy obdélníku.



```
\mfpic[10]{0}{7}{0}{10}
\rect{(2,1.5), (5,5.5)}
\polyline{(2,5.5), (3.5,8.5), (5,5.5)}
\lines{(3.7,1.5), (3.7,2.8),
        (4.7,2.8), (4.7,1.5)}
\polygon{(2.5,4), (2.5,5),
        (3.7,5), (3.7,4)}
\plotsymbol[17pt]{SolidTriangle}
        {(6.3,1.3)}
\plotsymbol[5pt]{Star}{{(1.3,8),
        (3,9.5), (5,8.5), (6.5,7.75), (5.8,9)}}
\endmfpic
```

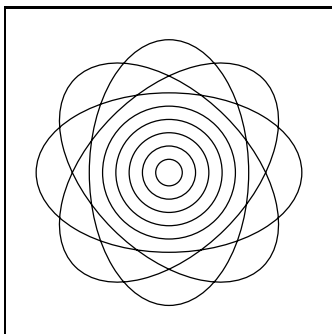
Kružnici se středem  $(x, y)$  a poloměrem  $r$  nakreslíme pomocí příkazu

`\circle {(x, y), r}` .

Elipsu popisuje makro

`\ellipse [ $\theta$ ] {(x, y),  $r_x, r_y$ }` ,

kde  $r_x$  je délka hlavní poloosy,  $r_y$  délka vedlejší poloosy,  $(x, y)$  je střed elipsy a nepovinný parametr  $\theta$  umožňuje otočení elipsy o  $\theta$  stupňů proti směru hodinových ručiček kolem jejího středu.



```
\mfpic[10]{-5}{5}{-5}{5}
\circle{(0,0), 2.5}
\circle{(0,0), 2}
\circle{(0,0), 1.5}
\circle{(0,0), 1}
\circle{(0,0), 0.5}
\ellipse{(0,0), 5, 3}
\ellipse[45]{(0,0), 5, 3}
\ellipse[90]{(0,0), 5, 3}
\ellipse[135]{(0,0), 5, 3}
\endmfpic
```

Bézierovu křivku procházející body  $(x_1, y_1), (x_2, y_2), \dots$  vykreslíme příkazem

`\curve [napětí] {(x_1, y_1), (x_2, y_2), \dots}` .

Nepovinný argument *napětí* upravuje napětí křivky, o němž jsme se zmiňovali již v první části této práce. Jeho předdefinovaná hodnota je 1.

Uzavřenou Bézierovu křivku procházející body  $(x_1, y_1), (x_2, y_2), \dots$  získáme příkazem

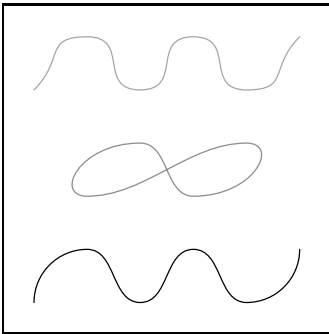
`\cyclic [napětí] {(x1, y1), (x2, y2), ...}` .

Nepovinný argument *napětí* využijeme stejně jako u předchozího příkazu v případě, že chceme upravit vzhled křivky.

Makro

`\fncurve [napětí] {(x1, y1), (x2, y2), (x3, y3)}`

vytvoří křivku procházející danými body. Výsledná křivka je více symetrická než při použití `\curve`, příkaz je proto vhodný například pro kreslení grafů funkcí. Nepovinný argument opět ovlivňuje napětí křivky a je přednastaven na hodnotu 1,2.



```
\mfpic[20]{0}{5}{0}{5}
\curve{(0,0),(1,1),(2,0),
      (3,1),(4,0),(5,1)}
\draw[.55white]\cyclic{
  (1,2),(2,3),(3,2),
  (4,3)}
\draw[.65white]\fncurve{
  (0,4),(1,5),(2,4),
  (3,5),(4,4),(5,5)}
\endmfpic
```

Chceme-li nakreslit kruhový oblouk, máme k dispozici několik možností pro jeho zadání:

`\arc [s] {(x1, y1), (x2, y2), θ}`

vykresluje oblouk z bodu  $(x_1, y_1)$  do bodu  $(x_2, y_2)$  tak, aby oblouk pokrýval úhel  $\theta$ , který je měřen proti směru hodinových ručiček.

Oblouk procházející třemi body  $(x_1, y_1), (x_2, y_2), (x_3, y_3)$  je zadán příkazem

`\arc [t] {(x1, y1), (x2, y2), (x3, y3)}` .

Makro

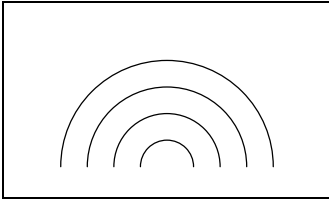
`\arc [p] {(xs, ys), θ1, θ2, r}`

nakreslí část kružnice se středem  $(x_s, y_s)$  a poloměrem  $r$  začínající úhlem  $\theta_1$  a končící úhlem  $\theta_2$ , přičemž oba úhly jsou měřeny proti směru hodinových ručiček od kladné části souřadné osy  $x$ .

Posledním možným zadáním kruhového oblouku je příkaz

`\arc [c] {(xs, ys), (x1, y1), θ}` ,

který vykreslí oblouk se středem  $(x_s, y_s)$  začínající v bodu  $(x_1, y_1)$  a pokrývající úhel  $\theta$ . Všechna předchozí zadání oblouku (s parametry [s], [t], [p]) jsou interně převáděna a vykreslena tímto makrem.



```
\mfpic[10]{-5}{5}{0}{5}
\arc[s]{(1,0),(-1,0),180}
\arc[t]{(-2,0),(0,2),(2,0)}
\arc[p]{(0,0),0,180,3}
\arc[c]{(0,0),(4,0),180}
\endmfpic
```

## 7.1. Souřadné osy

Pro znázornění souřadných os můžeme použít jedno z následujících maker:

```
\axes [délka_šipky] ,
\xaxis [délka_šipky] ,
\yaxis [délka_šipky] .
```

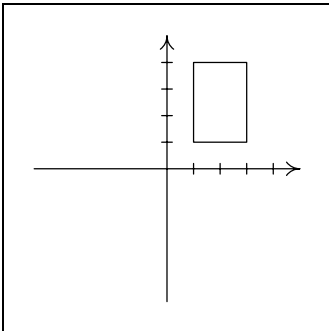
První z maker vyznačuje obě souřadné osy, zbývající dvě makra pouze souřadnou osu  $x$ , respektive  $y$ . Nevyužijeme-li nepovinný parametr, budou na osách vykresleny šipky o délce přiřazené proměnné `\axisheadlen` (přednastavena je hodnota 5 pt), tvaru určeného makrem `\arrow` a barvě odpovídající `headcolor`. Značky na osách získáme příkazem

```
\xmarks {seznam_bodů_na_ose_x}
```

pro osu  $x$  a příkazem

```
\ymarks {seznam_bodů_na_ose_y}
```

pro osu  $y$ . Délka čárečky odpovídá hodnotě `\hashlen`, nastavena je na 4 pt.



```
\mfpic[10]{-5}{5}{-5}{5}
\axes
\xmarks{1,2,3,4}
\ymarks{1,2,3,4}
\rect{(1,1),(3,4)}
\endmfpic
```

## 8. Polární souřadnice

Příkaz

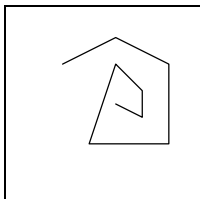
```
\plr {(r1, θ1), (r2, θ2), ... }
```

nahradí seznam bodů zadaných polárními souřadnicemi odpovídajícími souřadnicemi pravoúhlými.

Makro

```
\turtle {(x, y), (u1, v1), (u2, v2), ...}
```

nakreslí lomenou čáru, která vychází z bodu  $(x, y)$  a dále postupuje vždy o daný vektor.

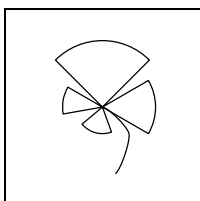


```
\mfpic[10]{0}{5}{0}{5}
\turtle{(1,4), (2,1), (2,-1), (0,-3),
(-3,0), (1,3), (1,-1), (0,-1), (-1,0.5)}
\endmfpic
```

Kruhovou výseč o středu  $(x, y)$  a poloměru  $r$  zadáváme dvěma ohraničujícími úhly, a to pomocí příkazu

```
\sector {(x, y), r, θ1, θ2}
```

Úhly  $\theta_1$  a  $\theta_2$  jsou měřeny proti směru hodinových ručiček od rovnoběžky se souřadnou osou  $x$ .



```
\mfpic[10]{0}{5}{0}{5}
\sector{(2.5,2.5), 2, 30, -30}
\sector{(2.5,2.5), 2.5, 45, 135}
\sector{(2.5,2.5), 1.5, 150, 190}
\sector{(2.5,2.5), 1, 220, 290}
\curve[2]{(2.5,2.5), (3.5,1.5), (3,0)}
\endmfpic
```

## 9. Uložení objektu

Používáme-li ve zdrojovém textu obrázku několikrát jeden *objekt*, který je dvojicí nebo cestou, je výhodné si jej uložit příkazem

```
\store {jméno} {objekt}
```

a na potřebném místě vložit pomocí makra

```
\mfobj {jméno}
```

Pomocí `\mfobj` lze vložit libovolnou proměnnou typu `path` (nejen uloženou pomocí `\store`), která je využívána v METAPOSTovém kódu našeho obrázku.

Praktické využití těchto příkazů bude ukázáno u vhodných obrázků následujících odstavců.

## 10. Prefixová makra

Některá z maker balíku `mfpic` se chovají jako tzv. *prefixová* makra, což v praxi znamená, že se tato makra aplikují na příkazy stojící ihned za nimi.

### 10.1. Kreslení

Makro

```
\draw [barva] ...
```

vykreslí námi požadovaný objekt plnou čarou.

Příkaz

```
\dashed [délka, mezera] ...
```

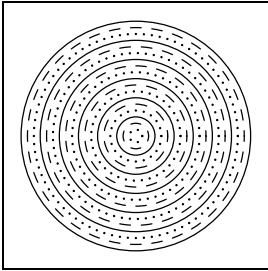
použije pro vykreslení objektu čáru přerušovanou. Přednastavená délka jedné čárky je dána hodnotou proměnné `\dashlen` (je rovna 4 pt); mezery mezi jednotlivými čárkami jsou určeny rozměrem `\dashspace` (je roven rovněž 4 pt). Čárky na začátku a konci křivky jsou dlouhé polovinu z hodnoty `\dashlen`. Aby byl zachován námi požadovaný poměr čárek a mezer, mohou být námi zadané délky čárek a rozměry mezer v nepovinných parametrech mírně upraveny, tedy mohou být prodlouženy či zkráceny.

Křivku vykreslíme tečkovanou čarou, použijeme-li makro

```
\dotted [velikost, mezera] ...
```

Předdefinovanou velikost teček udává hodnota proměnné `\dotsize` (je rovna 0,5 pt), předdefinovaný rozměr mezer je dán hodnotou `\dotsspace` (je roven 3 pt). Na začátku a konci křivky je vykreslena tečka. Velikost teček a mezer může být ovlivněna stejně jako u `\dashed`.





```

\mfpic[8]{-5.5}{5.5}{-5.5}{5.5}
\dotted\circle{(0,0),0.3}
\dashed\circle{(0,0),0.6}
\draw\circle{(0,0),0.9}
\dotted\circle{(0,0),1.2}
\dashed\circle{(0,0),1.5}
\draw\circle{(0,0),1.8}
\dotted\circle{(0,0),2.1}
\dashed\circle{(0,0),2.4}
\draw\circle{(0,0),2.7}
\dotted\circle{(0,0),3}
\dashed\circle{(0,0),3.3}
\draw\circle{(0,0),3.6}
\dotted\circle{(0,0),3.9}
\dashed\circle{(0,0),4.2}
\draw\circle{(0,0),4.5}
\dotted\circle{(0,0),4.8}
\dashed\circle{(0,0),5.1}
\draw\circle{(0,0),5.4}
\endmfpic

```

Zdá-li se nám tečkovaná nebo čárkovaná čára příliš obyčejná, máme možnost vykreslení křivky například pomocí symbolů trojúhelníku, kosočtverce, křížku či hvězdičky

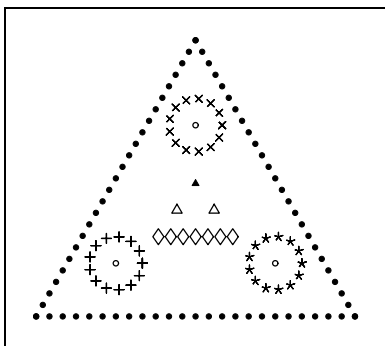
```
\plot [velikost, mezera] {symbol} ... .
```

Na místě povinného parametru *symbol* smí být jakýkoli symbol používaný u makra `\plotsymbol` (viz strana 75). Přednastavená *velikost* odpovídá `\pointsize (=2pt)` a *mezera* odpovídá `\symbolspace (=5pt)`.

Makro

```
\plotnodes [velikost] {symbol} ...
```

také vykresluje *symboly* stejné jako v makru `\plotsymbol` (viz strana 75), ale nakreslí je pouze v bodech, jimiž jsme zadali křivku, na kterou toto makro aplikujeme. Parametr *velikost* je nastaven na hodnotu rovnu `\pointsize (=2pt)`.



```

\mpic[10]{0}{12}{0}{10.4}
\plot{SolidCircle}
\lines{(0,0),(12,0)}
\plot{SolidCircle}
\lines{(0,0),(6,sqrt108)}
\plot{SolidCircle}
\lines{(12,0),(6,sqrt108)}
\plot[3pt,5pt]{Plus}
\circle{(3,2),1}
\plot[3pt,5pt]{Star}
\circle{(9,2),1}
\plot[3pt,5pt]{Cross}
\circle{(6,(2+sqrt27)),1}
\plotnodes{Circle}\polygon{
(3,2),(9,2),(6,(2+sqrt27))}
\plot[4pt,5pt]{Diamond}
\lines{(4.6,3),(7.4,3)}
\plotnodes[3pt]{Triangle}
\lines{(5.3,4),(6.7,4)}
\plotsymbol[2pt]
{SolidTriangle}{(6,5)}
\endmpic

```

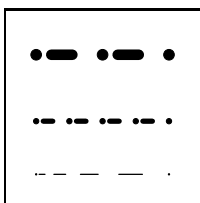
Pro nadefinování vlastní čárkované čáry použijeme příkaz

```
\dashpattern {jméno} {čárka1, mezero1, čárka2, mezero2, ... }
```

kde *jméno* je přiřazen vzorek tvořený postupně *čárkou<sub>1</sub>*, *mezerou<sub>1</sub>*, *čárkou<sub>2</sub>*, *mezerou<sub>2</sub>* atd. Čárka délky 0 pt odpovídá tečce. Popis vzoru musí končit *mezerou*.

Námi nadefinovanou čáru využijeme pro vykreslení křivky pomocí příkazu

```
\gendashed {jméno} ...
```



```

\mpic[10]{0}{5}{0}{5}
\dashpattern{moje}{0pt,1pt,2pt,3pt,4pt,
5pt,6pt,7pt,8pt,9pt}
\gendashed{moje}\lines{(0,0),(5,0)}
\pen{2pt}
\dashpattern{takemoje}{0pt,3pt,4pt,7pt}
\gendashed{takemoje}\lines{(0,2),(5,2)}
\pen{4pt}
\dashpattern{jestemoje}{0pt,5pt,7pt,
10pt}
\gendashed{jestemoje}\lines{(0,4.5),
(5,4.5)}
\endmpic

```

## 10.2. Výplně

Následující *prefixová* makra mohou být aplikována pouze na uzavřené křivky.

Makro

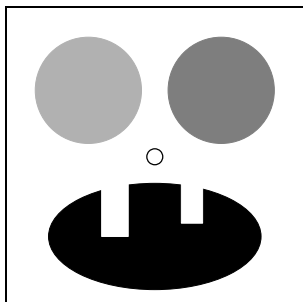
```
\gfill [barva] ...
```

vyplní uzavřenou křivku barvou uvedenou v nepovinném parametru *barva*. Není-li *barva* uvedena, je použita přednastavená barva odpovídající `fillcolor`. Inverzním příkazem ke `\gfill` je příkaz `\gclear`.

Příkaz

```
\shade [šed] ...
```

vyplní uzavřenou křivku. Nepovinným parametrem *šed* můžeme ovlivňovat stupeň šedi (přednastavena je hodnota `0.75white`).



```
\mpic[10]{-5}{5}{-5}{5}
\gfill\ellipse{(0,-3),4,2}
\gclear\rect{(-2,-3),(-1,-1)}
\gfill[white]
\rect{(1,-2.5),(1.8,-1)}
\shade[0.9]\circle{(-2.5,2.5),2}
\shade[0.7]\circle{(2.5,2.5),2}
\circle{(0,0),0.3}
\endmpic
```

Makro

```
\thatch [mezera,úhel] [barva] ...
```

vyšrafuje uzavřenou křivku. Mezery, standardně nastavené na hodnotu `\hatchspace` (=3 pt), mohou být změněny hodnotou nepovinného parametru *mezera*. Při nulové či záporné hodnotě je křivka vyplněna stejně jako příkazem `\gfill`. Tloušťka čáry, kterou jsou šrafy vykreslovány, je dána makrem `\hatchwd` (standardně =0,5 pt). Parametrem *úhel* ovlivníme sklon šrafů (přednastaven je úhel 0°) a parametrem *barva* jejich barvu (ta odpovídá hodnotě `hatchcolor`). Parametry *mezera* a *úhel* zadáváme pouze současně a také při využití parametru *barva* musí být zadány i *mezera* a *úhel*.

Pro práci se šrafováním jsou připravena makra:

```
\lhatch [mezera] [barva] ...
```

šrafuje z levého horního rohu k pravému dolnímu (to znamená pod úhlem  $-45^\circ$ );

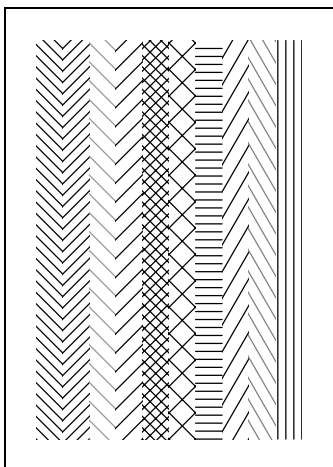
```
\rhatch [mezera] [barva] ...
```

šrafuje z pravého horního k levému dolnímu rohu (to znamená pod úhlem  $45^\circ$ );

```
\xhatch [mezera] [barva] ...
```

```
\hatch [mezera] [barva] ...
```

jsou kombinací `lhatch` a `rhatch`.



```

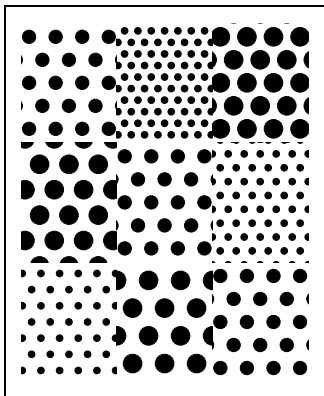
\mfpic[10]{0}{10}{0}{15}
\lhatch\rect{(0,0),(1,15)}
\rhatch\rect{(1,0),(2,15)}
\lhatch[5][.6white]
\rect{(2,0),(3,15)}
\rhatch[5]\rect{(3,0),(4,15)}
\lhatch\rect{(4,0),(5,15)}
\lhatch[7]\rect{(5,0),(6,15)}
\lhatch\rect{(6,0),(7,15)}
\lhatch[5,60 deg]
\rect{(7,0),(8,15)}
\lhatch[4,120 deg][.5white]
\rect{(8,0),(9,15)}
\lhatch[3,90 deg]
\rect{(9,0),(10,15)}
\endmfpic

```

Makro

`\polkadot [mezera] ...`

vyplní uzavřený útvar kruhy o velikosti udané hodnotou `\polkadotwd` (=5 pt) vyplněnými barvou `fillcolor` s mezerami odpovídající hodnotě `\polkadot-space` (=10 pt).



```

\mfpic[9]{0}{12}{0}{15}
\polkadot\rect{(0,10),(4,15)}
\polkadot\rect{(4,5),(8,10)}
\polkadot\rect{(8,0),(12,5)}
\polkadotwd{2.5pt}
\polkadot[5]
\rect{(4,10),(8,15)}
\polkadot[6]
\rect{(8,5),(12,10)}
\polkadot[7]\rect{(0,0),(4,5)}
\polkadotwd{7pt}
\polkadot\rect{(8,10),(12,15)}
\polkadot[11]
\rect{(0,5),(4,10)}
\polkadot[12]
\rect{(4,0),(8,5)}
\endmfpic

```

Další možností vyplňování uzavřené křivky je její „pokrytí“ námi nadefinovaným vzorem, jakési „kachličkování“. Vzor vytvoříme pomocí okolí

```

\tile {jméno_vzoru, jednotka, šířka, výška, ořezání}
...
\endtile ,
\begin{tile} ... \end{tile} (v LATEXu).

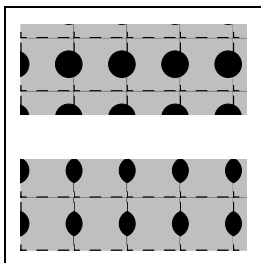
```

Vzor je tvořen v obdélníku, respektive čtverci, s protilehlými vrcholy (0, 0) a (šířka, výška) a definují ho veškeré příkazy v okolí `\tile ... \endtile`, respektive `\begin{tile} ... \end{tile}`. Povinný parametr *jednotka* určuje jednotku, ve které je daný vzor kreslen; parametr *ořezání* má hodnoty `true` (kachlička je oříznuta na námi zadaný obdélník nebo čtverec) a `false` (kachličky nejsou ořezány; při pokládání jsou použity takové, jaké jsme je vytvořili, nezávisle na možné přesahy přes dané rozměry).

Příkaz

```
\tess {jméno_vzoru} ...
```

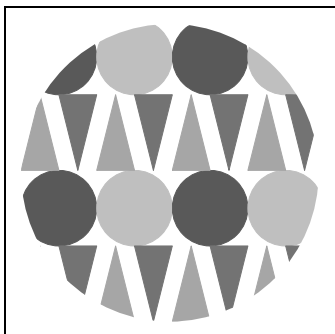
vyplní uzavřenou křivku vzorem *jméno\_vzoru*. Vždy musíme použít nějaký námi vytvořený vzor, neboť v `mfpic` není žádný předdefinovaný. Při pokusu o pokrytí otevřené křivky nedostaneme chybové hlášení, ale křivka bude pouze vykreslena. Kachličkování je prováděno tak, aby levý dolní roh některé z pokrývajících kachliček ležel v počátku, a dělá se postupně zdola nahoru a zleva doprava (viz efekt v případě `false` na následujícím obrázku).



```

\mfpic[17]{0}{5}{0}{5}
\begin{tile}{kachl_i,20pt,1,1,true}
\dashed\shade\rect{(0,0),(1,1)}
\gfill\circle{(-0.1,0.5),0.25}
\gfill\circle{(1.1,0.5),0.25}
\end{tile}
\tile{kachl_ii,20pt,1,1,false}
\dashed\shade\rect{(0,0),(1,1)}
\gfill\circle{(-0.1,0.5),0.25}
\gfill\circle{(1.1,0.5),0.25}
\end{tile}
\tess{kachl_i}\rect{(0,0),(5,2)}
\tess{kachl_ii}\rect{(0,3),(5,5)}
\endmfpic

```



```

\mfpic[8]{-7}{7}{-7}{7}
\tile{kachlicka,cm,2,2,true}
\gfill[.35white]
\circle{(.5,1.5),.5}
\gfill[.75white]
\circle{(1.5,1.5),.5}
\gfill[.65white]\polygon{
(0,0),(0.25,1),(0.5,0)}
\gfill[.45white]\polygon{
(.5,1),(0.75,0),(1,1)}
\gfill[.65white]\polygon{
(1,0),(1.25,1),(1.5,0)}
\gfill[.45white]\polygon{
(1.5,1),(1.75,0),(2,1)}
\endtile
\tess{kachlicka}
\circle{(0,0),7}
\endmfpic

```

### 10.3. Šipky

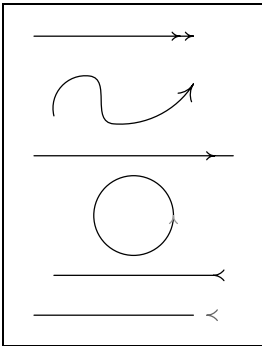
Makro

```
\arrow [l délka_šipky] [r otočení] [b posunutí_zpět] [c barva] .
```

Jednotlivé nepovinné parametry mají následující význam:

- **b** – posun šipky směrem k počátku křivky, a to po tečně křivky v jejím posledním bodu; přednastavená hodnota pro tento parametr je 0 pt,
- **c** – barva šipky; předdefinovaná barva odpovídá barvě `headcolor`,
- **l** – délka šipky; předdefinovaná délka koresponduje s hodnotou `\headlen`, což je 3 pt,
- **r** – otočení šipky proti směru hodinových ručiček; přednastaven je úhel 0°.

Parametry mohou být zapsány v jakémkoli pořadí. Při použití otočení současně s posunem je třeba myslet na to, že je nejdříve provedeno otočení a poté posun, ovšem ten už probíhá na otočené tečně. Při zápisu nepovinných parametrů tohoto makra může (a nemusí) být mezi *l* a *délkou\_šipky*, respektive mezi *r* a *otočením* a podobně, mezera.



```

\mfpic[15]{0}{5}{0}{7}
\arrow\arrow[b 5pt]\lines{
(0,7),(4,7)}
\arrow[l 6pt]\curve{(0.5,5),(1.4,6),
(2,4.8),(4,5.8)}
\arrow[b 7pt]\lines{(0,4),(5,4)}
\arrow[c .6white]\circle{
(2.5,2.5),1}
\arrow[r180][14pt]
\lines{(0.5,1),(4.5,1)}
\arrow[r180][14pt][b5pt][c.4white]
\lines{(0,0),(4,0)}
\endmfpic

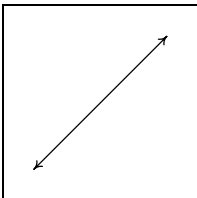
```

#### 10.4. Změna orientace křivky

Orientaci křivky můžeme jednoduše změnit příkazem

```
\reverse ...
```

Tohoto makra využijeme například při vykreslování šipek v počátečním i koncovém bodu křivky:



```

\mfpic[10]{0}{5}{0}{5}
\arrow\reverse\arrow\lines{(0,0),(5,5)}
\endmfpic

```

První šipka zleva ve zdrojovém zápisu je kreslena v počátečním bodu dané křivky.

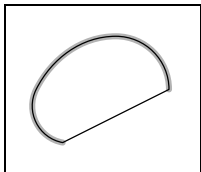
#### 10.5. Uzavírání křivek

V `mfpic` jsou nadefinované tyto uzavřené křivky: `\rect`, `\circle`, `\ellipse`, `\sector`, `\cyclic`, `\polygon`, `\plrregion` a `\btwnfcn`. Libovolné jiné křivky, které se nám na pohled mohou jevit uzavřenými, `mfpic` chápe jako otevřené.

Pokud jsme vytvořili otevřenou křivku, můžeme využít některý z následujících příkazů pro její uzavření. Všechny uvedené příkazy způsobí uzavření i z pohledu `mfpic`.

- `\lclosed` – uzavře křivku úsečkou z počátečního do koncového bodu křivky
- `\bclosed` – uzavře křivku Bézierovou křivkou vedenou z prvního k poslednímu bodu dané křivky

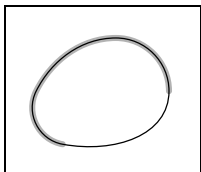
- `\cbclosed` – uzavře křivku doplněním kubického B-splinu mezi prvním a posledním bodem zadané křivky
- `\sclosed` – uzavře křivku tak, aby se jevila co nejhladší, přičemž v tomto smyslu může být změněn tvar křivky, kterou uzavíráme
- `\uclosed` – uzavře křivku tak, aby se jevila co nejhladší, ale tvar zadané křivky nezmění



```
\mpic[20]{0.5}{3}{1}{3}
\pen{2.3pt}
\draw[.7white]\curve{(1,1),(.5,2),
                    (2,3),(3,2)}

\pen{.5pt}
\lclosed\curve{(1,1),(.5,2),(2,3),
              (3,2)}

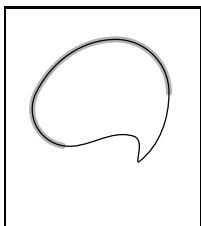
\endmpic
```



```
\mpic[20]{0.5}{3}{1}{3}
\pen{2.3pt}
\draw[.7white]\curve{(1,1),(.5,2),
                    (2,3),(3,2)}

\pen{.5pt}
\bclosed\curve{(1,1),(.5,2),(2,3),
              (3,2)}

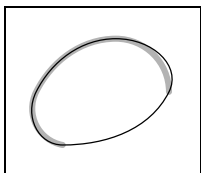
\endmpic
```



```
\mpic[20]{0.5}{3}{0}{3}
\pen{2.3pt}
\draw[.7white]\curve{(1,1),(.5,2),
                    (2,3),(3,2)}

\pen{.5pt}
\cbclosed\curve{(1,1),(.5,2),(2,3),
              (3,2)}

\endmpic
```

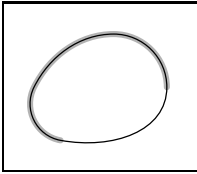


```
\mpic[20]{0.5}{3}{1}{3}
\pen{2.3pt}
\draw[.7white]\curve{(1,1),(.5,2),
                    (2,3),(3,2)}

\pen{.5pt}
\sclosed\curve{(1,1),(.5,2),(2,3),
              (3,2)}

\endmpic
```





```

\mfpic[20]{0.5}{3}{1}{3}
\pen{2.3pt}
\draw[.7white]\curve{(1,1),(.5,2),
                    (2,3),(3,2)}

\pen{.5pt}
\uclosed\curve{(1,1),(.5,2),(2,3),
              (3,2)}

\endmfpic

```

## 10.6. Napojení křivek

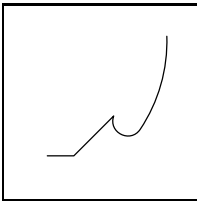
Okolí

```

\connect ... \endconnect ,
\begin{connect} ... \end{connect} (pro LATEX)

```

sestrojí úsečku z koncového bodu výše zapsané otevřené křivky k počátečnímu bodu další otevřené křivky. Výsledkem je otevřená křivka.



```

\mfpic[10]{0}{5}{0}{5}
\begin{connect}
\lines{(0.5,0.5),(1.5,0.5)}
\curve{(3,2),(4,1.5),(5,5)}
\end{connect}
\endmfpic

```

## 10.7. Afinní transformace

V `mfpic` můžeme použít následující afinní transformace:

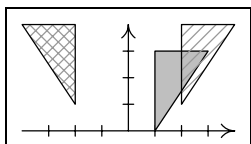
- `\rotate {θ}` – otočení o  $\theta$  stupňů kolem počátku proti směru hodinových ručiček,
- `\rotatearound {(x,y)} {θ}` – otočení o  $\theta$  stupňů kolem bodu  $(x,y)$  proti směru hodinových ručiček,
- `\turn [(x,y)] {θ}` – otočení o  $\theta$  stupňů kolem bodu  $(x,y)$  proti směru hodinových ručiček; není-li uveden nepovinný parametr, otáčení je provedeno kolem počátku,
- `\mirror {(x1,y1)} {(x2,y2)}` – osová souměrnost zadaná přímkou procházející body  $(x_1, y_1)$  a  $(x_2, y_2)$ ,
- `\reflectabout {(x1,y1)} {(x2,y2)}` – osová souměrnost daná přímkou procházející body  $(x_1, y_1)$  a  $(x_2, y_2)$ ,
- `\shift {(u,v)}` – posunutí o vektor  $(u, v)$ ,
- `\scale {k}` – stejnolehlost se středem v počátku a koeficientem  $k$ ,

- `\xscale {k}` – změna měřítka na ose  $x$  s koeficientem  $k$ ,
- `\yscale {k}` – změna měřítka na ose  $y$  s koeficientem  $k$ ,
- `\zscale {(u,v)}` – stejnolehlost s koeficientem o velikosti vektoru  $(u,v)$  a otočení proti směru hodinových ručiček o úhel, který svírá vektor  $(u,v)$  s kladnou částí osy  $x$  (bod  $(x,y)$  odpovídá bod  $(xu - yv, xv + yu)$ ),
- `\xslant {k}` – zkosení ve směru osy  $x$  s koeficientem  $k$  (obrazem bodu  $(x,y)$  je bod  $(x + ky, y)$ ),
- `\yslant {k}` – zkosení ve směru osy  $y$  s koeficientem  $k$  (bod  $(x,y)$  odpovídá bod  $(x, ky + y)$ ),
- `\zslant {(u,v)}` – transformace, která bod  $(x,y)$  zobrazí na bod  $(xu + yv, xv + yu)$ ,
- `\boost {\chi}` – má stejný efekt jako `\zslant {(cosh \chi, sinh \chi)}`,
- `\xyswap` – osová souměrnost daná osou  $y = x$ .

## 10.8. Používání afinních transformací

Použijeme-li libovolnou afinní transformaci, budou se jí podrobovat všechny objekty v aktuálním okolí `\mpic ... \endmpic` zapsané níže od této transformace. Při aplikaci další transformace dojde ke složení s předcházející.

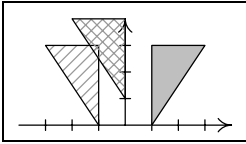
Dvěma nepřiliš složitými obrázky můžeme ukázat, že grupa afinních transformací s operací skládání není komutativní. (U obou obrázků využíváme posunutí o vektor  $(1,1)$  a osovou souměrnost podle osy  $y$ , přičemž v prvním obrázku musí být zadaná osa  $y$  posunuta o odpovídající vektor, neboť `\mpic` ji před provedením osové souměrnosti posune o vektor předcházejícího posunutí; u druhého obrázku zase upravíme příslušným způsobem posunutí tak, abychom neprováděli posunutí o vektor  $(1,1)$  zobrazený v předchozí osové souměrnosti.)



```

\mpic[10]{-4}{4}{0}{4}
\axes
\xmarks{-3,-2,-1,1,2,3}
\ymarks{1,2,3}
\store{trojuhelnik}
      {\polygon{(1,0),(1,3),(3,3)}}
\draw\shade\mfobj{trojuhelnik}
\shift{(1,1)}\draw
      \rhatch[\the\hatchspace]
      [.6white]\mfobj{trojuhelnik}
\reflectabout{(-1,-1)}{(-1,3)}\draw
      \hatch[3][.6white]
      \mfobj{trojuhelnik}
\endmpic

```



```

\mfpic[10]{-4}{4}{0}{4}
\axes
\xmarks{-3,-2,-1,1,2,3}
\ymarks{1,2,3}
\store{trojuhelnik}
  {\polygon{(1,0),(1,3),(3,3)}}
\draw\shade\mfobj{trojuhelnik}
\reflectabout{(0,0)}{(0,3)}\draw
  \rhatch[\the\hatchspace][.6white]
  \mfobj{trojuhelnik}
\shift{(-1,1)}\draw
  \hatch[3][.6white]
  \mfobj{trojuhelnik}
\endmpic

```

Pro případ, kdy nám skládání transformací není moc vhod, lze použít okolí

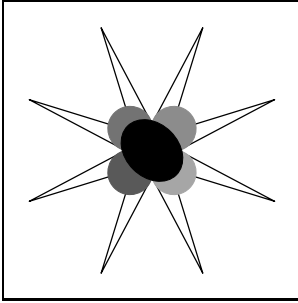
```

\coords ... \endcoords ,
\begin{coords} ... \end{coords} (pro LATEX).

```

Při otevření okolí dojde k „uschování“ právě aktuální transformace a v okamžiku uzavření je transformace navržena.

Následující tři obrázky přibližují chování tohoto okolí. První z nich provede postupné otáčení trojúhelníku o  $45^\circ$ , otevřením okolí `coords` je uloženo otočení o  $7 \cdot 45^\circ$  a poté je vykreslen (nezapomeňme, že je nejprve otočen o  $7 \cdot 45^\circ$ ) a otočen další útvar.

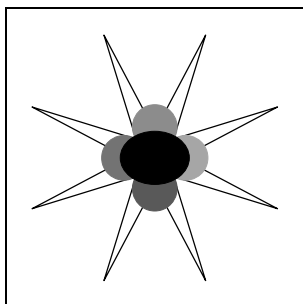


```

\mpic[10]{-5}{5}{-5}{5}
\store{listik}{\polyline{(1,0),
    5*dir(22.5).dir(45)}}
\store{listecek}{\lclosed\curve{
    dir(-45),(2,0),dir(45)}}
\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\begin{coords}
\gfill[.65white]\mfobj{listecek}
\rotate{90}\gfill[.55white]
    \mfobj{listecek}
\rotate{90}\gfill[.45white]
    \mfobj{listecek}
\rotate{90}\gfill[.35white]
    \mfobj{listecek}
\end{coords}
\gfill\ellipse{(0,0),1.3,1}
\endmpic

```

Ve druhém obrázku je na počátku uložena identita, neboť žádná jiná transformace nebyla použita, potom je aplikováno otáčení vždy o  $45^\circ$ , to jest celkem o  $7 \cdot 45^\circ$ . Uzavřením prvního okolí `coords` je zapomenuto otočení o  $7 \cdot 45^\circ$  a je navracena identita, což způsobí, že útvar vytvořený v dalším okolí `coords` se nijak netransformuje.

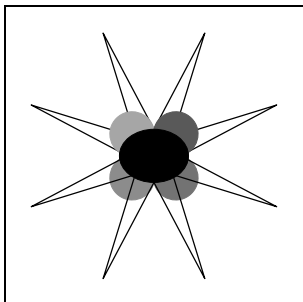


```

\mfpic[10]{-5}{5}{-5}{5}
\store{listik}{\polyline{(1,0),
    5*dir(22.5),dir(45)}}
\store{listecek}{\lclosed\curve{
    dir(-45),(2,0),dir(45)}}
\mfobj{listik}
\begin{coords}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\end{coords}
\begin{coords}
\gfill[.65white]\mfobj{listecek}
\rotate{90}\gfill[.55white]
    \mfobj{listecek}
\rotate{90}\gfill[.45white]
    \mfobj{listecek}
\rotate{90}\gfill[.35white]
    \mfobj{listecek}
\end{coords}
\gfill\ellipse{(0,0),1.3,1}
\endmfpic

```

Ve třetím obrázku jsou dvě okolí `coords` vnořena do sebe, což je praktické v situacích, kdy využíváme některé z transformací a na určité objekty potřebujeme aplikovat ještě nějaké transformace další. V naší ukázce je tedy trojúhelník nejprve otočen třikrát, ve vnořeném okolí `coords` je první vykreslovaný útvar otočen o  $3 \cdot 45^\circ$ , druhý útvar je otočen o  $3 \cdot 45^\circ + 90^\circ$  atd.; při uzavírání vnořeného okolí `coords` je vrácena hodnota otočení na  $3 \cdot 45^\circ$ .



```

\mfpic[10]{-5}{5}{-5}{5}
\store{listik}{\polyline{(1,0),
    5*dir(22.5),dir(45)}}
\store{listecek}{\lclosed\curve{
    dir(-45),(2,0),dir(45)}}
\mfobj{listik}
\begin{coords}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\begin{coords}
\gfill[.65white]\mfobj{listecek}
\rotate{90}\gfill[.55white]
    \mfobj{listecek}
\rotate{90}\gfill[.45white]
    \mfobj{listecek}
\rotate{90}\gfill[.35white]
    \mfobj{listecek}
\end{coords}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\rotate{45}\mfobj{listik}
\end{coords}
\gfill\ellipse{(0,0),1.3,1}
\endmfpic

```

Zdrojové texty předchozích příkladů bychom mohli učiniti trochu „elegantnějšími“ využitím cyklu  $\TeX$ u, kterým nahradíme sedmkrát prováděné otáčení:

```

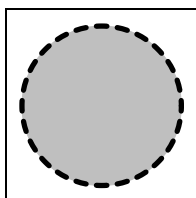
\newcount\pocet
\def\kresli#1{\pocet=#1
  \loop\ifnum\pocet>0
    \rotate{45}\mfobj{listik}
    \advance \pocet by -1
  \repeat}
\kresli{7}

```

### 10.9. Pořadí prefixových maker

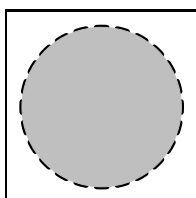
Při práci s tímto druhem maker je důležité si uvědomit, že příkaz vlevo je vždy aplikován na výsledek příkazu vpravo. Jinak řečeno, námi napsaný postup tvorby obrázku je prováděn od konce (zprava).

Podívejme se, jak se mění výsledek kreslení, použijeme-li různá pořadí příkazů `\gfill`, `\dashed`:



```
\mfpic[30]{-1}{1}{-1}{1}
\pen{2pt}
\dashed\gfill[0.75white]\circle{(0,0),1}
\endmfpic
```

Nejdříve je nadefinována kružnice se středem  $(0, 0)$  a poloměrem 1, poté je vyplněna. Hranice vytvořeného kruhu je vyznačen čárkovanou čarou, přičemž střed pera je veden po hraniční křivce (kružnici), což způsobí, že polovina tloušťky kružnice překrývá výplň kruhu.



```
\mfpic[30]{-1}{1}{-1}{1}
\pen{2pt}
\gfill[0.75white]\dashed\circle{(0,0),1}
\endmfpic
```

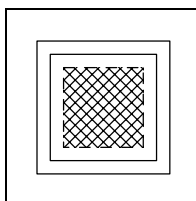
Této posloupnosti příkazů odpovídá překrytí poloviny tloušťky pera, vyznačujícího kružnici, výplní kruhu.

## 11. Rendrování

Redrováním rozumíme proces vlastního překreslení geometrického popisu obrázku. V METAPOSTu je to konkrétně tvorba postscriptového popisu křivek, výplní a podobně. Příkazem

```
\setrender {příkazy-TEXu}
```

změníme přednastavený způsob rendrování – pomocí `\draw`.



```
\mfpic[10]{0}{5}{0}{5}
\rect{(0,0),(5,5)} % \draw není nutné
\draw\rect{(.5,.5),(4.5,4.5)}
\setrender{\dashed\xhatch}
\rect{(1,1),(4,4)}
\endmfpic
```

## 12. Funkce

Makro

```
\fdef {název_funkce} {proměnná1, proměnná2, ... } {předpis}
```

nadefinuje funkci *název\_funkce* (*název\_funkce* smí být složen pouze z písmen a podtržítka) o proměnných *proměnná<sub>1</sub>*, *proměnná<sub>2</sub>*, ... a předpisem *předpis*, který je předán METAPOSTu ke zpracování. Můžeme tudíž používat všechny operace, které umožňuje METAPOST, tedy +, -, \*, /, \*\*; závorky pro upřesňování pořadí operací (, ); znaménka rovnosti, respektive nerovnosti =, <, >, <=, >= a další funkce uvedené v následující tabulce:

<b>round</b>	zaokrouhlení podle obvyklých pravidel,
<b>floor</b>	zaokrouhlení směrem dolů,
<b>ceiling</b>	zaokrouhlení směrem nahoru,
<b>min</b>	minimum (dva a více argumentů),
<b>max</b>	maximum (dva a více argumentů),
<b>abs</b>	absolutní hodnota,
<b>sqrt</b>	druhá odmocnina,
<b>sind</b>	funkce sinus (argument zadáváme ve stupních),
<b>cosd</b>	funkce kosinus (argument zadáváme ve stupních),
<b>mlog</b>	logaritmus se základem $e^{1/256}$ ( $mlog = 256 * \ln x$ ),
<b>mexp</b>	inversní funkce k funkci předchozí.

Dále můžeme využít funkce předdefinované v **grafbase.mp**, což jsou tyto:

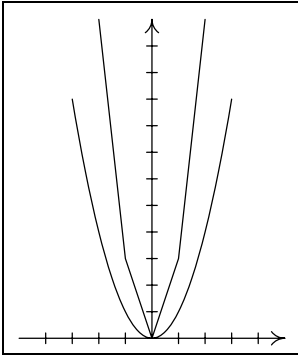
<b>tand</b>	funkce tangens (argument zadáváme ve stupních),
<b>cotd</b>	funkce kotangens (argument zadáváme ve stupních),
<b>secd</b>	funkce sekans (argument zadáváme ve stupních),
<b>cscd</b>	funkce kosekans (argument zadáváme ve stupních),
<b>asin</b>	funkce arkussinus (výsledek dostáváme ve stupních),
<b>acos</b>	funkce arkuskosinus (výsledek dostáváme ve stupních),
<b>atan</b>	funkce arkustangens (výsledek dostáváme ve stupních),
<b>sin</b>	funkce sinus (argument zadáváme v radiánech),
<b>cos</b>	funkce kosinus (argument zadáváme v radiánech),
<b>tan</b>	funkce tangens (argument zadáváme v radiánech),
<b>cot</b>	funkce kotangens (argument zadáváme v radiánech),
<b>sec</b>	funkce sekans (argument zadáváme v radiánech),
<b>csc</b>	funkce kosekans (argument zadáváme v radiánech),
<b>invsin</b>	funkce arkussinus (výsledek dostáváme v radiánech),
<b>invcos</b>	funkce arkuskosinus (výsledek dostáváme v radiánech),
<b>invtan</b>	funkce arkustangens (výsledek dostáváme v radiánech),
<b>exp</b>	exponenciální funkce $e^x$ ,
<b>sinh</b>	funkce hyperbolický sinus,
<b>cosh</b>	funkce hyperbolický kosinus,



<code>tanh</code>	funkce hyperbolický tangens,
<code>ln</code>	přirozený logaritmus,
<code>asinh</code>	funkce argument hyperbolického sinu,
<code>acosh</code>	funkce argument hyperbolického kosinu,
<code>atanh</code>	funkce argument hyperbolického tangens.

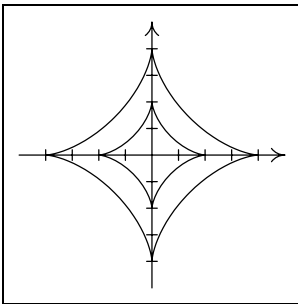
Následující čtyři makra mají společný nepovinný parametr *typ\_křivky*, který definuje, zda je funkce vykreslena jako Bézierova křivka (odpovídá hodnota parametru *s*) nebo jako lomená čára (odpovídá hodnota *p*). Dále je pro nás podstatné, že smíme využívat pouze proměnných *x* a *t*, jak je udáno u jednotlivých maker:

- `\function [typ_křivky] {x_min, x_max, krok} {f(x)}` – vykresluje funkci  $f(x)$  na intervalu  $\langle x_{\min}, x_{\max} \rangle$  (přičemž METAPOST postupuje od  $x_{\min}$  postupně po kroku *krok* až k  $x_{\max}$ ); nepovinný parametr je přednastaven na *s*,
- `\parafcn [typ_křivky] {t_min, t_max, krok} {f(t)}` – vytvoří parametricky zadanou křivku  $f(t) = (x(t), y(t))$  s parametrem *t* z intervalu  $\langle t_{\min}, t_{\max} \rangle$  (a krokem *krok*); nepovinný argument je přednastaven na *s*,
- `\plrfcn [typ_křivky] {t_min, t_max, krok} {f(t)}` – určuje křivku  $f(t)$  danou polárními souřadnicemi  $(t, r)$ , kde  $r = f(t)$ ; úhel *t* nabývá hodnot z intervalu  $\langle t_{\min}, t_{\max} \rangle$  a je měřen ve stupních; nepovinný parametr je nastaven jako *s*,
- `\btwnfcn [typ_křivky] {x_min, x_max, krok} {f(x)} {g(x)}` – vyznačí oblast ohraničenou funkcemi  $f(x)$ ,  $g(x)$  na intervalu  $\langle x_{\min}, x_{\max} \rangle$  a dvěma vertikálními přímkami v  $x_{\min}$ ,  $x_{\max}$ ; parametr *typ\_křivky* je nastaven jako *p*,
- `\plrregion [typ_křivky] {t_min, t_max, krok} {f(t)}` – stanoví oblast ohraničenou křivkou  $f(t)$  zadanou polárními souřadnicemi  $(t, r)$ , kde  $r = f(t)$ ; úhel *t* nabývá hodnot v intervalu  $\langle t_{\min}, t_{\max} \rangle$  a je měřen ve stupních; oblast je uzavřena úsečkou spojující počátek souřadné soustavy a bod odpovídající  $t_{\min}$  a spojnicí bodu odpovídajícího  $t_{\max}$  s počátkem; nepovinný parametr je přednastaven na *p*.



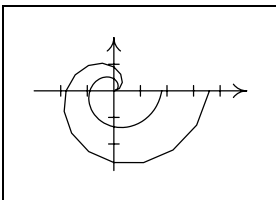
```
\mfpic[10]{5}{5}{0}{12}
\axes
\xmarks{-4,-3,-2,-1,1,2,3,4}
\ymarks{1,2,3,4,5,6,7,8,9,10,11}
\function{-3,3,.5}{x*x}
\function[p]{-2,2,1}{3*x*x}
\endmfpic
```

Asteroida:

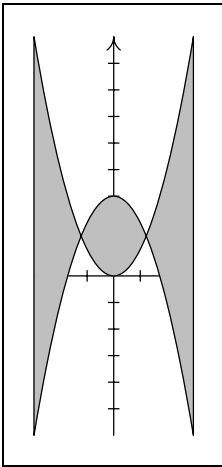


```
\mfpic[10]{-5}{5}{-5}{5}
\axes
\xmarks{-4,-3,-2,-1,1,2,3,4}
\ymarks{-4,-3,-2,-1,1,2,3,4}
\parafcn{0,1440,22.5}{
((3*cosd (t/4))+cosd ((3*t)/4),
(3*sind (t/4))-sind ((3*t)/4))}
\parafcn[p]{0,1440,22.5}{
(.5*((3*cosd (t/4))+
cosd ((3*t)/4)),
.5*((3*sind(t/4))-
sind ((3*t)/4)))}
\endmfpic
```

Archimédova spirála:



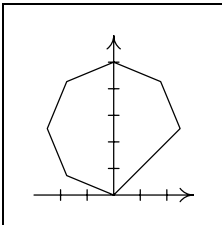
```
\mfpic[10]{-3}{5}{-3}{2}
\axes
\xmarks{-2,-1,1,2,3,4}
\ymarks{-2,-1,1}
\plrfcn{0,360,22.5}{.005*t}
\plrfcn[p]{0,360,22.5}{.01*t}
\endmfpic
```



```

\mfpic[10]{-3}{3}{-6}{9}
\axes
\xmarks{-2,-1,1,2}
\ymarks{-5,-4,-3,-2,-1,1,2,3,4,5,6,7,8}
\shade\btwnfcn[s]{-3,3,.5}{x*x}{-x*x+3}
\btwnfcn[s]{-3,3,.5}{x*x}{-x*x+3}
\endmfpic

```

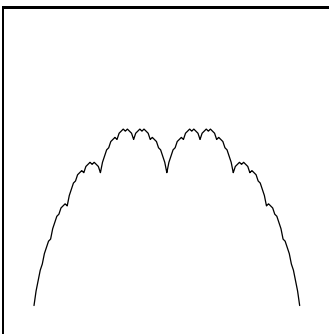


```

\mfpic[10]{-3}{3}{0}{6}
\axes
\xmarks{-2,-1,1,2}
\ymarks{1,2,3,4,5}
\plrregion{45,180,22.5}{5*sind(t)}
\endmfpic

```

Funkce na následující obrázku je sedmým částečným součtem řady, která konverguje ke spojitě funkci nemající nikde derivaci. V mfpic je znázornění takové funkce velmi jednoduché.



```

\mfpic[100]{0}{1}{0}{1}
\fdef{f}{x}{min(x-(floor x),
(ceiling x)-x)}
\function[p]{0,1,1/128}{f(x)+
(.5*f(2*x))+
(.25*f(4*x))+
(.125*f(8*x))+
(.0625*f(16*x))+
(.03125*f(32*x))+
(.015625*f(64*x))}
\endmfpic

```

### 13. Kreslení dat z externího souboru

Balík maker `mfpic` nám umožňuje vykreslovat útvary sestavené z dat externích souborů.

Externí soubory použité v tomto textu jsou nazvány `data`, `data1` a podobně. Při spouštění `TEX` z instalace `emTEX` byla tato jména nepostačující (pokud neměla příponu) a musela být doplněna o tečku na konci jména (`data.`).

Příkaz

```
\datafile {jméno_souboru}
```

vytvoří lomenou čáru procházející body zapsanými v souboru `jméno_souboru`. Předpokládáme, že každý neprázdný řádek obsahuje alespoň dvě čísla. První dvě čísla na řádku reprezentují souřadnice  $x$  a  $y$  jednoho bodu, ostatní čísla jsou ignorována. Prázdné řádky na začátku souboru jsou ignorovány, všechny další jsou chápány jako konec křivky; komentář je uvozen znakem procenta (%); viz příkaz `\mfpdatacomment`. Takto vytvořený objekt smíme uzavřít, vybarvit a podobně.

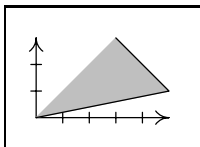
Příkaz

```
\smoothdata [napětí]
```

způsobí, že výsledkem makra `\datafile` je Bézierova křivka místo lomené čáry, a příkaz

```
\unsmoothdata
```

nastaví zpět použití lomené čáry.



```
\mfpic[10]{0}{5}{0}{3}  
\axes  
\xmarks{1,2,3,4}  
\ymarks{1,2}  
\shade\lclosed\datafile{data}  
\datafile{data}  
\endmfpic
```

Soubor `data` je pro tento příklad zadán:

```
prázdná řádka  
prázdná řádka  
prázdná řádka
```

```
0 0 7 2  
5 1 5  
3 3  
  
0 0  
3 4  
0 5
```

Chceme-li uvozovat komentáře jiným způsobem, než-li znakem procenta, nadefinujeme si k tomuto účelu námi zvolený znak příkazem

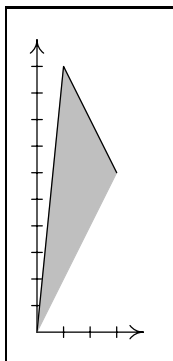
```
\mfdatacomment \znak
```

Zároveň bude při čtení daného souboru zrušeno zvláštní postavení znaku procenta.

Příkaz

```
\using {vstupní_vzor} {výstupní_vzor}
```

změní standardní způsob zpracování datového souboru. Použijeme-li například vzorek `\using{#1 #2 #3}{#3,#1}`, bude to znamenat následující: čísla na řádce až po první mezeru jsou přiřazena do parametru `#1 vstupního_vzoru`, další čísla až po druhou mezeru do parametru `#2` a zbývající čísla na řádce do parametru `#3 vstupního_vzoru`. Pro vykreslení křivky jsou použity souřadnice dle *výstupního\_vzoru* odpovídající jednotlivým parametrům *vstupního\_vzoru*. Námi nadefinovaný způsob zpracování datových souborů platí až do konce okolí `\mfpic`, tedy až do příkazu `\endmfpic`, a vztahuje se na zpracování souborů při použití příkazů `\datafile` i `\plotdata` (viz následující příkaz).



```
\mfpic[10]{0}{4}{0}{11}
\axes
\marks{1,2,3}
\ymarks{1,2,3,4,5,6,7,8,9,10}
\using{#1 #2 #3}{#2,#1*2}
\shade\lclosed\datafile{data}
\datafile{data}
\endmfpic
```

Pro nakreslení několika lomených čar (nebo Bézierových křivek po využití příkazu `\smoothdata`) zadaných v jednom datovém souboru do jednoho obrázku je připraveno makro

```
\plotdata {jméno_souboru}
```

Jednotlivé čáry jsou od sebe odděleny jednou volným řádkem, více prázdných řádků je chápáno jako konec souboru. Čáry jsou vykreslovány postupně, a to šesti různými typy čar. Příkazem `\coloredlines` změňme barvu čar (střídá se osm různých barev počínající od černé). Příkazy `\pointedlines` a `\datapointsonly` využívají makra `\plot` a `\plotnodes` a způsobují vykreslení čar nebo pouze zadaných bodů pomocí devíti různých symbolů známých z makra `\plotsymbol` (viz strana 75).

Nastavení původního způsobu (to jest různými typy čar) kreslení se provádí příkazem `\dashedlines`. Chceme-li ovlivnit první typ čáry (respektive barvu

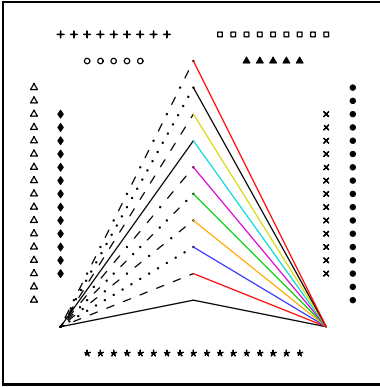
nebo druh symbolu), použijeme `\mfplinestyle {číslo}`, kde *číslo* je nezáporné. Typy čar (respektive barvy, druhy symbolu) jsou číslovány od 0; je-li v okolí `mfpic` více příkazů `\plotdata`, číslování vždy navazuje na číslování předcházející.

U tohoto příkazu se nesmí využívat žádná prefixová makra.

Pro datové soubory `data1`, `data2`, `data3`:

<code>data1</code>	<code>data2</code>	<code>data3</code>
0 0	10 0	1 -1
5 1	5 1	9 -1
0 0	10 0	-1 1
5 2	5 2	-1 9
0 0	10 0	11 1
5 3	5 3	11 9
0 0	10 0	0 11
5 4	5 4	4 11
0 0	10 0	6 11
5 5	5 5	10 11
0 0	10 0	0 2
5 6	5 6	0 8
0 0	10 0	10 2
5 7	5 7	10 8
0 0	10 0	1 10
5 8	5 8	3 10
0 0	10 0	7 10
5 9	5 9	9 10
0 0	10 0	
5 10	5 10	

dostaneme následující obrázek.

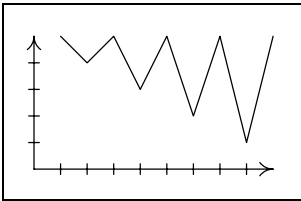


```

\mfpic[10]{-1}{11}{-1}{11}
\plotdata{data1}
\coloredlines
\plotdata{data2}
\pointedlines
\plotdata{data3}
\endmfpic

```

V dalším obrázku využijeme makro `\sequence`, které reprezentuje posloupnost přirozených čísel. V našem případě nabývají  $x$ -ové souřadnice bodů hodnot 1, 2, ... až počet řádků v souboru.



```

\mfpic[10]{0}{9}{0}{5}
\axes
\xmarks{1,2,3,4,5,6,7,8}
\ymarks{1,2,3,4}
\using{#1}{\sequence,#1}
\datafile{data4}
\endmfpic

```

Soubor data4:

```

5
4
5
3
5
2
5
1
5

```

## 14. Popisy obrázků

Příkazem

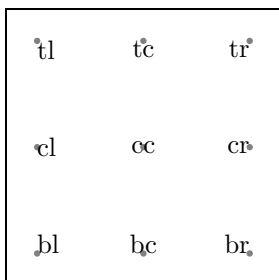
```

\tlabel [umístění otočení] (x,y) {popiska}

```

umístíme k bodu  $(x, y)$  text *popiska*. Parametr *umístění* je složen ze dvou písmen; první písmeno udává vertikální polohu (čtyři varianty: t, c, b, B odpovídající postupně anglickým výrazům top, center, bottom, Baseline), druhé polohu horizontální (tři varianty: l, c, r korespondující s anglickými left, center, right). Mezi písmeny nesmí být mezera. Použitím jednotlivých kombinací získáme popisky umístěné vzhledem k bodu  $(x, y)$  podle následujících obrázků; přednastavená hodnota tohoto argumentu je Bl.

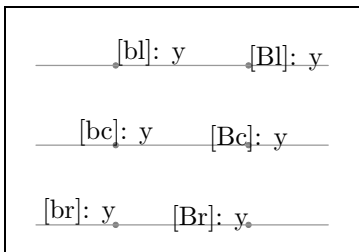
Všimněme si, že zadáváme polohu bodu vůči popisce (u METAPOSTu tomu bylo naopak).



```
\mpic[10]{-4}{4}{-4}{4}
\fillcolor{0.5white}
\point{(-4,4),(0,4),(4,4),
(-4,0),(0,0),(4,0),
(-4,-4),(0,-4),(4,-4)}
\tlabel[lt](-4,4){lt}
\tlabel[tc](0,4){tc}
\tlabel[tr](4,4){tr}
\tlabel[cl](-4,0){cl}
\tlabel[cc](0,0){cc}
\tlabel[cr](4,0){cr}
\tlabel[bl](-4,-4){bl}
\tlabel[bc](0,-4){bc}
\tlabel[br](4,-4){br}
\endmpic
```

Použijeme-li argument **b** je text sázen tak, aby bounding box písmene byl postaven na účaří, zatímco argument **B** ztotožní  $y$ -ovou souřadnici referenčního bodu bounding boxu písmene s  $y$ -ovou souřadnicí bodu, který zadáváme. Rozdíl mezi parametrem **b** a **B** je vidět na ukázce písmene  $y$ :





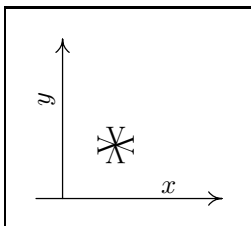
```

\mpic[10]{-3}{8}{-6}{1}
\fillcolor{.5white}
\point{(0,0),(5,0),(0,-3),
(5,-3),(0,-6),(5,-6)}
\draw[.6white]
\lines{(-3,0),(8,0)}
\draw[.6white]
\lines{(-3,-3),(8,-3)}
\draw[.6white]
\lines{(-3,-6),(8,-6)}
\tlabel[bl](0,0){[bl]: y}
\tlabel[Bl](5,0){[Bl]: y}
\tlabel[bc](0,-3){[bc]: y}
\tlabel[Bc](5,-3){[Bc]: y}
\tlabel[br](0,-6){[br]: y}
\tlabel[Br](5,-6){[Br]: y}
\endmpic

```

Parametrem *otočení* text otáčíme kolem bodu, který popisujeme; představeno je nula stupňů. Mezi parametry *umístění* a *otočení* může a nemusí být mezera.

O zpracování popisek rozhoduje volba `mplabels` (viz strana 71). Je-li využita, umísťuje a otáčí popisky METAPOST, v opačném případě je otáčení vyloučeno (nepovinný argument *otočení* je ignorován) a text sází T<sub>E</sub>X.



```

\mpic[20]{-.5}{3}{0}{3}
\axes
\tlabel[bc](2,0.1){$x$}
\tlabel[cr 90](-0.3,2){$y$}
\tlabel[bc](1,1){V}
\tlabel[bc 90](1,1){V}
\tlabel[bc 180](1,1){V}
\tlabel[bc 270](1,1){V}
\endmpic

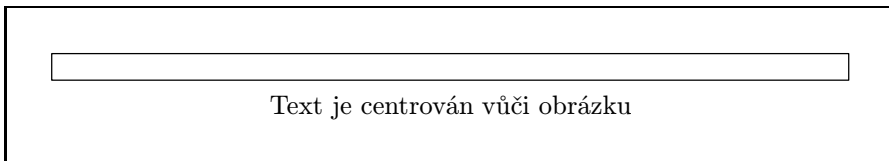
```

Příkaz

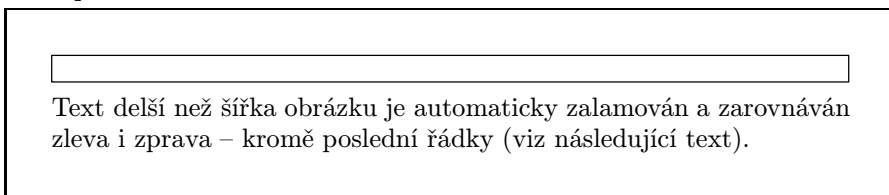
```
\tcaption [maximum, řádka] {text}
```

umísťí *text* pod obrázek.

Chování makra v různých situacích ukazují nejlépe následující obrázky.

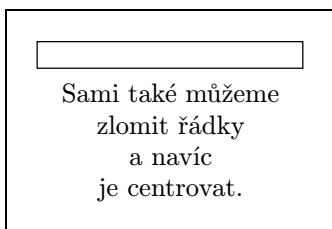


```
\mfpic[10]{0}{30}{0}{1}
\rect{(0,0),(30,1)}
\tcaption{Text je centrován vůči obrázku}
\endmfpic
```



```
\mfpic[10]{0}{30}{0}{1}
\rect{(0,0),(30,1)}
\tcaption{Text delší než šířka obrázku je automaticky zalamován
a~zarovnáván zleva i~zprava -- kromě poslední řádky
(viz následující text).}
\endmfpic
```

Překročí-li délka řádky *textu* hodnotu danou součinem *maxima* a šířky obrázku, jsou řádky lámány tak, aby jejich šířka odpovídala hodnotě součinu *řádky* a šířky obrázku. Přednastavená hodnota *maxima* je 1,2; *řádky* 1,0.



```
\mfpic[10]{0}{10}{0}{1}
\rect{(0,0),(10,1)}
\usecenteredcaptions
\tcaption{Sami také můžeme\\
zlomit řádky\\
a~navíc\\
je centrovat.}
\endmfpic
```

## 15. Pole křivek

Okolí

```
\patharr {jméno_pole} ... \endpatharr ,  
\begin{patharr} {jméno_pole} ... \end{patharr} (v LATEXu),
```

v němž je zapsáno několik křivek, vytvoří z těchto křivek pole nazvané *jméno\_pole*. Na jednotlivé položky tohoto pole se odkazujeme příkazy `\mfobj{jméno_pole1}`, `\mfobj{jméno_pole2}`, ...



```
\mpic[10]{0}{6}{0}{3}  
\patharr{pole}  
\lines{(1,0),(1,3)}  
\rect{(2,0),(3,3)}  
\polygon{(4,0),(5,3),(6,0)}  
\endpatharr  
\mfobj{pole1}  
\shade\mfobj{pole2}  
\rhatch\mfobj{pole3}  
\endmpic
```

## 16. Definice příkazů

Při otevření nového okolí `\mpic ... \endmpic` jsou vždy znovu nadefinovány všechny příkazy kreslení. Pokud bychom chtěli některý z těchto příkazů předefinovat, musíme to tedy provádět pouze uvnitř okolí `\mpic ... \endmpic`. Ovšem například změna velikosti pera mezi jednotlivými okolími `mpic` se v obrázcích projeví.

## 17. Složitější obrázky

Chceme-li tvořit složitější obrázky, budeme zřejmě potřebovat znát kromě příkazů `mpic` některé příkazy samotného METAPOSTu a navíc nám mohou posloužit i makra souboru `grafbase.mp`.

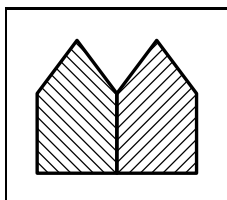
Příkazem

```
\mfsrc {přikazy_METAPOSTu}
```

zapisujeme `přikazy_METAPOSTu` přímo do výstupního souboru `.mp`.

Nadefinujeme-li si vlastní transformaci (přímo v METAPOSTu, musí být v uživatelských souřadnicích), aplikujeme ji pomocí příkazu

```
\applyT {transformed transformace} .
```

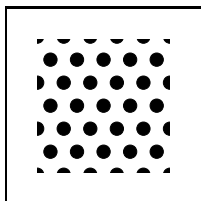


```

\mpic[10]{-3}{3}{0}{5}
\mfsrc{transform tr;
(1,0) transformed tr = (-1,0);
(2,0) transformed tr = (-2,0);
(1.5,1) transformed tr = (-1.5,1);}
\pen{1.3pt}
\draw\rhatch\polygon{(0,0),(3,0),
(3,3),(1.5,5),(0,3)}
\draw\lhatch\applyT{transformed tr}
\polygon{(0,0),(3,0),(3,3),
(1.5,5),(0,3)}
\endmpic

```

Využitím makra `image`, o kterém jsem se zmínila v první části, obrázek přímo nevykreslíme, jen ho uložíme do proměnné typu `picture`:



```

\def\uschovej#1#2{%
\mfsrc{picture #1; #1 = image{}%
#2%
\mfsrc{}};%
\mpic[10]{0}{5}{0}{5}
\uschovej{mujobr}{
\polkadot\rect{(0,0),(5,5)}}
\mfsrc{draw mujobr;}
\endmpic

```

Pokud bychom požadovali vykreslit jen část obrázku „oříznutou“ uzavřenou křivkou, lze využít makro ze souboru `grafbase.mp`

`clipto` (*proměnná\_typu\_picture*) *křivka* ,

kde *proměnná\_typu\_picture* je obrázek, který budeme „ořezávat“ křivkou *křivka*.  
Velmi podobný je příkaz

`clipped` (*proměnná\_typu\_picture*) *křivka* ,

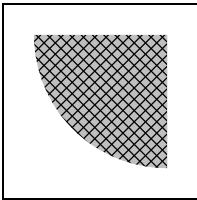
jen s tím rozdílem, že výstupem je oříznutý obrázek, jenž musí být přiřazen proměnné typu `picture`. *Křivku* je možné vytvořit příkazem `\store` (viz strana 79); je jen vhodné něco vědět o souřadných systémech, se kterými pracujeme.

Balík `mpic` pracuje interně v souřadném systému – device coordinates – odlišném od uživatelského souřadného systému – graph coordinates. Převod z uživatelského systému do interního udává afinní transformace nazvaná `zconv` (inverzní transformací je `invzconv`). Lineární zobrazení indukované touto afinní transformací se nazývá `vconv` (inverzní potom `invvconv`). Například

`\lines{(1,2),(3,-2)}`

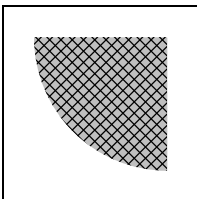
je ekvivalentní `\mfsrc{draw zconv((1,2)) -- zconv((3,-2))};`

Pro nás je tedy důležité, že křivka uložená pomocí makra `\store` je v uživatelských souřadnicích, avšak makra `clip`to a `clipped` požadují argument v interních souřadnicích.



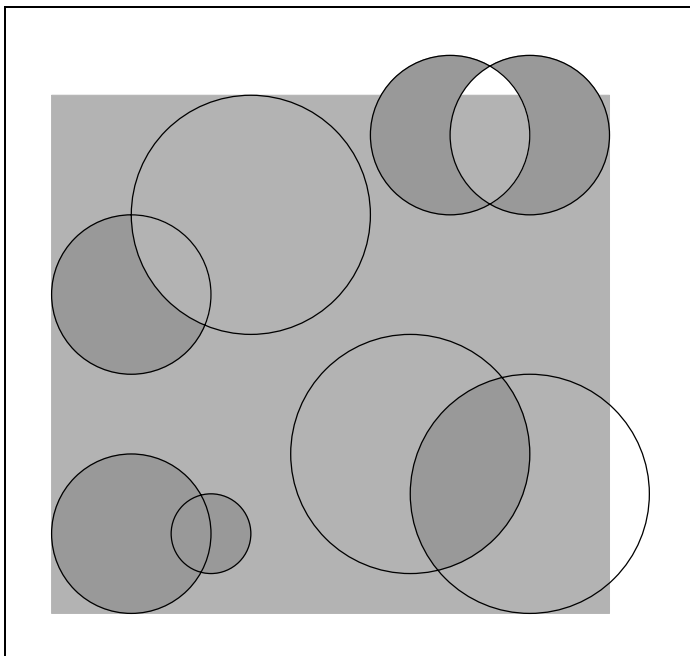
```
\def\uschovej#1#2{%
\mfsrc{picture #1; #1 = image(}%
#2%
\mfsrc{}};%
\mpic[10]{0}{5}{0}{5}
\uschovej{dalsiobr}{
\XHatch\SHade\rect{(0,0),(5,5)}
\store{krivka}{\circle{(5,5),5}}
\mfsrc{clip}to (dalsiobr)
zconv(krivka);
draw dalsiobr;}
\endmpic
```

Shodný výsledek dává i následující zdrojový text.



```
\def\uschovej#1#2{%
\mfsrc{picture #1; #1 = image(}%
#2%
\mfsrc{}};%
\mpic[10]{0}{5}{0}{5}
\uschovej{dalsiobr}{
\XHatch\SHade\rect{(0,0),(5,5)}
\mfsrc{picture} o;
o~ = clipped (dalsiobr)
fullcircle scaled 100
shifted (50,50);
draw o;}
\endmpic
```

Pro kreslení průniku a rozdílu dvou množin také použijeme zmíněná makra. Proměnná `active_plane` je proměnná METAPOSTu typu `picture` definovaná v `grafbase.mp` a ukládá se do ní postupně vše námi nakreslené. Má podobnou úlohu jako v METAPOSTu proměnná `currentpicture`. Dále si v definici příkazu `\rozdil` všimněte, že aby šedé pozadí nebylo překryto (barvy v PostScriptu jsou tzv. krycí, tedy pozdější barva překryje předchozí), musela se část pozadí (uloženého v `active_plane`) „uschovat“ do proměnné `pom_iii` a na závěr se zase nakreslila. Jinak by zde vznikl „otvor“ v barvě `background` (ta se používá v příkazu `undraw` pro mazání).



```

\def\uschovej#1#2{%
  \mfsrc{picture #1; #1 = image{}%
  #2%
  \mfsrc{}};%
\def\sjednoceni#1#2{%
\gfill#1\gfill#2}%
\def\prunik#1#2{%
  \uschovej{mnozina_i}{\gfill #1}
  \store{krivka_i}{#2}
  \mfsrc{clipto (mnozina_i) zconv(krivka_i);
  draw mnozina_i;}}%
\def\rozdil#1#2{%
  \uschovej{mnozina_i}{\gfill #1}
  \store{krivka_i}{#1}
  \store{krivka_ii}{#2}
  \mfsrc{picture mnozina_ii;
  mnozina_ii=clipped (mnozina_i) zconv(krivka_ii);
  mnozina_ii:=image(draw mnozina_i; undraw mnozina_ii);}
  \mfsrc{picture mnozina_iii; mnozina_iii=clipped (active_plane)
  zconv(krivka_i); clipto (mnozina_iii) zconv(krivka_ii);}
  \mfsrc{draw mnozina_ii; draw mnozina_iii;}}%

```

```

\def\symrozdil#1#2{%
  \rozdil{#1}{#2}
  \rozdil{#2}{#1}}%
\mpic[30]{0}{7.5}{0}{7}
\gfill[.7white]\rect{(0,0),(7,6.5)}
\fillcolor{.6white}
\sjednoceni{\circle{(1,1),1}}{\circle{(2,1),.5}}
\circle{(1,1),1}
\circle{(2,1),.5}
\prunik{\circle{(4.5,2),1.5}}{\circle{(6,1.5),1.5}}
\circle{(4.5,2),1.5}
\circle{(6,1.5),1.5}
\rozdil{\circle{(1,4),1}}{\circle{(2.5,5),1.5}}
\circle{(1,4),1}
\circle{(2.5,5),1.5}
\symrozdil{\circle{(6,6),1}}{\circle{(5,6),1}}
\circle{(6,6),1}
\circle{(5,6),1}
\endmpic

```

Kopretina na titulní straně této práce také využívá průniky množin. Autorem tohoto obrázku je doc. J. Kuben a zdrojový text vypadá takto (komentářem je barevná varianta):

```

\mpic[7]{-20}{20}{-20}{20}
\newcommand{\obraz}[2]{%
\mfsrc{picture #1; #1=image()}%
#2%
\mfsrc{}};}
\newcommand{\rozdil}[2]{%
\obraz{pom_i}{\gfill #1}%
\store{krivka_i}{#1} \store{krivka_ii}{#2}%
\mfsrc{picture pom_ii; pom_ii=clipped (pom_i) zconv(krivka_ii);
pom_ii:=image(draw pom_i; undraw pom_ii);}%
\mfsrc{picture pom_iii;
pom_iii=clipped (active_plane) zconv(krivka_i);
clipto (pom_iii) zconv(krivka_ii);}%
\mfsrc{draw pom_ii; draw pom_iii;}}
\newcommand{\symrozdil}[2]{\rozdil{#1}{#2}\rozdil{#2}{#1}}
\gfill[.6white]\circle{(0,0),12} \gfill[green]\circle{(0,0),12}
\gfill\circle{(0,0),4} \gfill[red]\circle{(0,0),4}
\store{ki}{\bclosed\plrfcn{0,360,1.25}{12+8*cosd(8t)}}
\store{kii}{\bclosed\plrfcn{0,360,1.25}{12-8*cosd(8t)}}

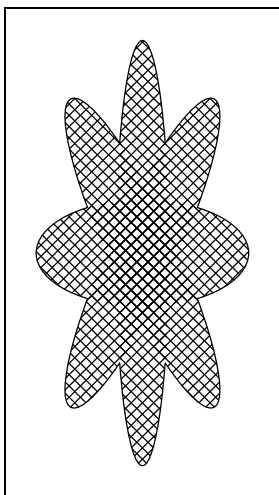
```

```

\fillcolor{.7white}           %\fillcolor{yellow}
\symrozdil{\mfobj{ki}}{\mfobj{kii}}
%\drawcolor[named]{OrangeRed}
\mfobj{ki}\mfobj{kii}
\endmpic

```

Křivky v interních souřadnicích jsou argumenty dalšího příkazu – clipsto: clipsto (*proměnná\_typu\_picture*) (*pole\_uzavřených\_křivek*) , který ořeže *proměnnou\_typu\_picture* na sjednocení vnitřků všech křivek, které jsou obsaženy v *poli\_uzavřených\_křivek*.



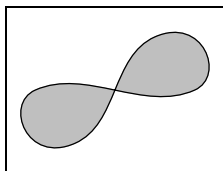
```

\def\uschovej#1#2{%
  \mfsrc{picture #1; #1 = image()}%
  #2%
  \mfsrc{)};}%
\mpic[10][20]{-4}{4}{-4}{4}
\uschovej{ctverec}{
  \xhatch\rect{(-4,-4),(4,4)}
\patharr{pole}
\ellipse{(0,0),4,1}
\ellipse[45]{(0,0),4,1}
\ellipse[90]{(0,0),4,1}
\ellipse[135]{(0,0),4,1}
\endpatharr
\mfsrc{for i=1 upto pole:
pole[i]:=zconv(pole[i]); endfor;
clipsto (ctverec,pole);
draw ctverec;}
\mfsrc{path pompole[];
for i=1 upto pole:
pompole[i]:=
(subpath (-.25*length(pole[i]),
.25*length(pole[i])) of pole[i]);
endfor;
for i=1 upto pole:
pompole[i+4]:=
(subpath (.25*length(pole[i]),
.75*length(pole[i])) of pole[i]);
endfor;
draw (buildcycle(pompole1 for i=2
upto 2pole: ,pompole[i] endfor));}
\endmpic

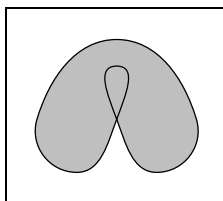
```



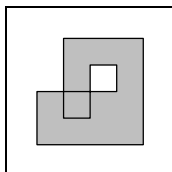
Na závěr se zmíníme o vybarvování uzavřených křivek. PostScript má dva způsoby, jak vyplňuje uzavřené křivky. METAPOST používá ten, který vybarví daný bod, pokud počet oběhů křivky kolem tohoto bodu (tzv. index bodu vzhledem ke křivce) je nenulový. Orientace po směru nebo proti směru hodinových ručiček (tedy zda je index kladný nebo záporný) nehraje roli. Situaci ilustrují následující obrázky.



```
\mpic[10]{-3.5}{3.5}{-2.5}{2.5}
\draw\gfill[.75white]
\cyclic{(-3,0),(-1.5,-2),(1.5,2),(3,0)}
\endmpic
```



```
\mpic[10]{-3}{3}{-2}{3}
\draw\gfill[.75white]
\cyclic{(-3,0),(-1.5,-2),(0,0),
(0,2),(0,0),(1.5,-2),(3,0),(0,3)}
\endmpic
```



```
\mpic[10]{0}{4}{0}{4}
\draw\gfill[.75white]
\polygon{(0,0),(4,0),(4,4),(1,4),
(1,1),(2,1),(2,3),(3,3),(3,2),(0,2)}
\endmpic
```

## 18. Použití `mpic` při tvorbě obrázků

V další části mé práce bych ráda na několika obrázcích ze středoškolské tematiky prakticky demonstrovala využití balíku `mpic`.

Obrázky jsou umístěny záměrně tak, aby je bylo možno porovnávat se zdrojovým kódem.

Ve většině obrázků je použito makro

```
\mfsrc {příkazy METAPOSTu} ,
```

pomocí něhož vpisujeme do výstupního souboru `.mp` libovolné příkazy METAPOSTu.

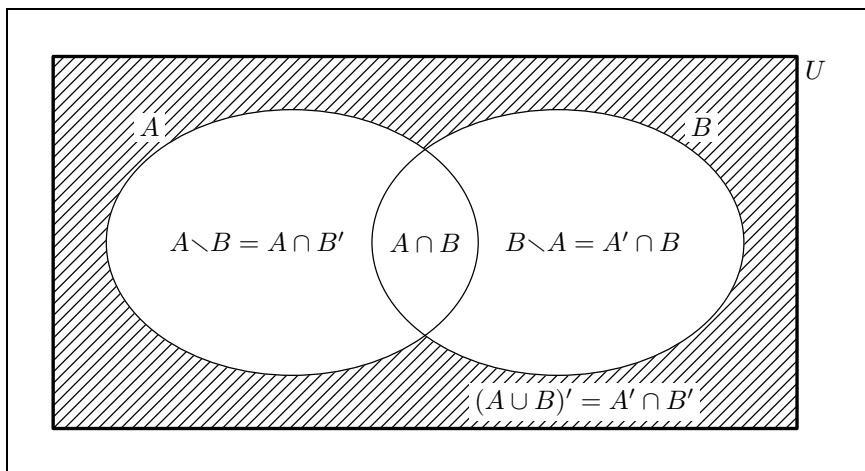
U dvou obrázků je ukázáno použití příkazu

`\mfpvrbtex {přikazy TEXu}` ,  
který zapíše *přikazy T<sub>E</sub>Xu* ohraničené příkazy `verbatimex ... etex` do výstupního souboru `.mp` (viz první část práce).

### 18.1. Grafické znázornění množin

Obrázek zachycuje Vennův diagram znázorňující obecnou polohu dvou množin.

Je zde naznačeno využití příkazu `\mfpvrbtex`. Pokud by byla skutečně použita jeho zakomentovaná forma, znak rozdílů množin by byl vysázen L<sup>A</sup>T<sub>E</sub>Xem, což je v tomto případě vhodné. Samozřejmě je ještě třeba upravit soubor `makempx` (viz první část práce). Znak rozdílů množin lze také vysázet příkazem `\setminusminus` (nemusíme nic upravovat, protože plain T<sub>E</sub>X ho zná), ale výsledek není ideální. Třetí možností je nadefinování vlastního symbolu. Makro `\obd` vytvoří pod popiskou bílý obdélník.



```

%\mfpverbtex{\documentclass{article}
%           \usepackage{amsmath,amsfonts,amssymb}
%           \begin{document}}
\mfpverbtex%
\def\smallsetminus{{\font\amsb=msbm10
  \mathbin{\hbox{\amsb\char"72}}}}}%
\def\obd#1{%
\mfsrc{picture obr; obr = image()}%
\tlabel#1%
\mfsrc{}; unfill bbox (obr); draw obr; }%
}%
\mfpic[10]{-14}{15}{-7}{7}
\pen{1.3pt}
\rect{(-14,-7),(14,7)}
\pen{.5pt}
\rhatch\rect{(-14,-7),(14,7)}
\gfill[white]\ellipse{(5,0),7,5}
\draw\gfill[white]\ellipse{(-5,0),7,5}
\ellipse{(5,0),7,5}
\tlabel[cl](14.3,6.5){\mathbb{U}}
\obd[[br](-10,4){\mathbb{A}}}]
\obd[[bl](10,4){\mathbb{B}}}]
\obd[[cc](6,-6){\mathbb{A}\cup\mathbb{B}}}]
\tlabel[cc](0,0){\mathbb{A}\cap\mathbb{B}}
%\tlabel[cl](3,0){\mathbb{B}\setminus\mathbb{A}}
%\tlabel[cl](3,0){\mathbb{B}\setminus\mathbb{A}}
%\tlabel[cl](3,0){\mathbb{B}\setminus\mathbb{A}}
%\tlabel[cr](-3,0){\mathbb{A}\setminus\mathbb{B}}
%\tlabel[cr](-3,0){\mathbb{A}\setminus\mathbb{B}}
\end{mfpic

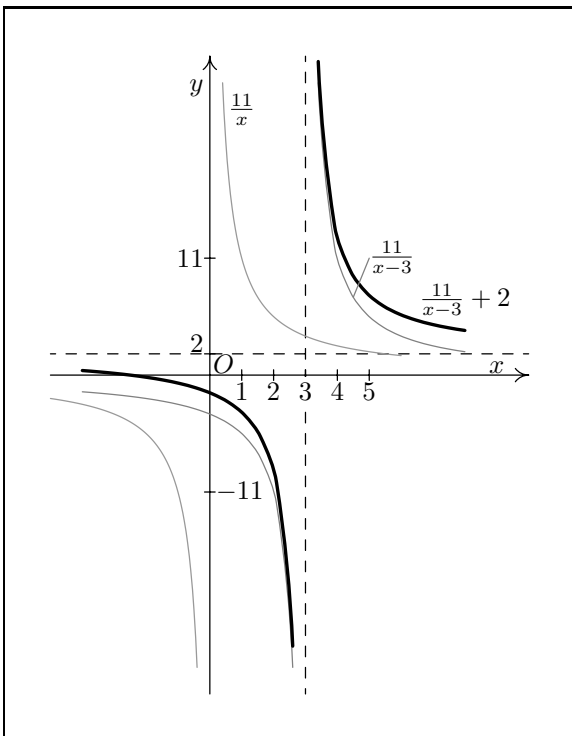
```

## 18.2. Grafy funkcí

Graf funkce  $g: y = \frac{2x+5}{x-3}$ .

U tohoto obrázku si znovu ukážeme praktické použití příkazu `\mfpverbtex`; pokud by byl skutečně využit, mohli bychom požadované popisky vysázet příkazem `\frac`.

Makro `\obd` nám zajistí, aby se popiska bodu na ose  $x$  nepřekrývala s přerušovanou čarou naznačující posun osy  $y$  při konstrukci funkce.



```
%\mfpverbtex{\documentclass{article}
%          \begin{document}}
%%% makro pro umístění textu na vybarvenou plochu
\def\obd#1{%
\mfsrc{picture obr; obr = image()}%
\tlabel#1%
\mfsrc{}; unfill bbox (obr); draw obr; }%
}%
```

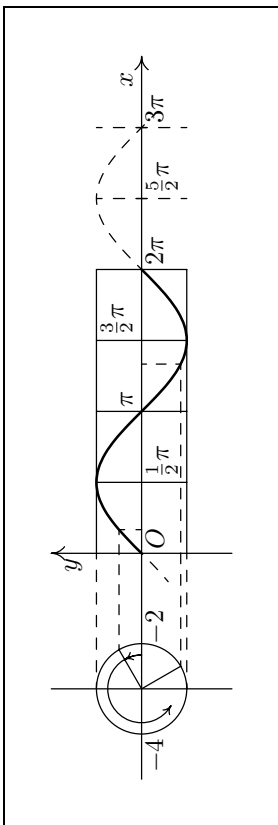
```

\mfpic[12][4]{-5}{10}{-30}{30}
\axes
\xmarks{1,2,3,4,5}
\ymarks{-11,2,11}
\tlabel[cr]{-.2,27}{y$}
\tlabel[bc]{9,.2}{x$}
\tlabel[bl]{.1,.2}{0$}
\dashed\lines{(-5,2),(10,2)}
\dashed\lines{(3,-30),(3,30)}
\draw[.6white]\function{-5,-.4,.1}{11/x}
\draw[.6white]\function{.4,6,.1}{11/x}
\draw[.5white]\function{-4,2.6,.5}{11/(x-3)}
\draw[.5white]\function{3.4,8,.5}{11/(x-3)}
\pen{1.3pt}
\function{-4,2.6,.5}{(11/(x-3))+2}
\function{3.4,8,.5}{(11/(x-3))+2}
\pen{.5pt}
\tlabel[cr]{-.2,11}{11}
\tlabel[cl]{.2,-11}{-$11$}
\tlabel[br]{-.2,2.2}{2}
\tlabel[bc]{1,-2.3}{1}
\tlabel[bc]{2,-2.3}{2}
\obd{[bc]{3,-2.3}{3}}
\tlabel[bc]{4,-2.3}{4}
\tlabel[bc]{5,-2.3}{5}
\draw[.5white]\lines{(4.5,11/1.5),(5,11)}
%\tlabel[cc]{1,25}{$\frac{11}{x}$}
%\tlabel[cl]{5,11}{$\frac{11}{x-3}$}
%\tlabel[cc]{8,7}{$\frac{11}{x-3}+2$}
\tlabel[cc]{1,25}{$11\over x$}
\tlabel[cl]{5,11}{$11\over {x-3}$}
\tlabel[cc]{8,7}{${11\over {x-3}}+2$}
\endmfpic

```

Graf funkce sinus.

Chceme-li obrázek otočit o devadesát stupňů, lze to provést až po nakreslení celého obrázku, nebo můžeme na úvod zapsat příkaz `\rotate{90}` a potom ještě otáčet všechny popisky (tento způsob je zakomentovaný a otočení popisek naznačeno u první z nich).



```

\mfpic[17]{-5}{11}{-2}{2}
%\rotate{90}
\axes
\xmarks{(5/2)*pi,3*pi}
%\tlabel[cr 90](-.1,1.5){$$}
\tlabel[cr](-.1,1.5){$$}
\tlabel[bc](10.5,.2){$x$}

```

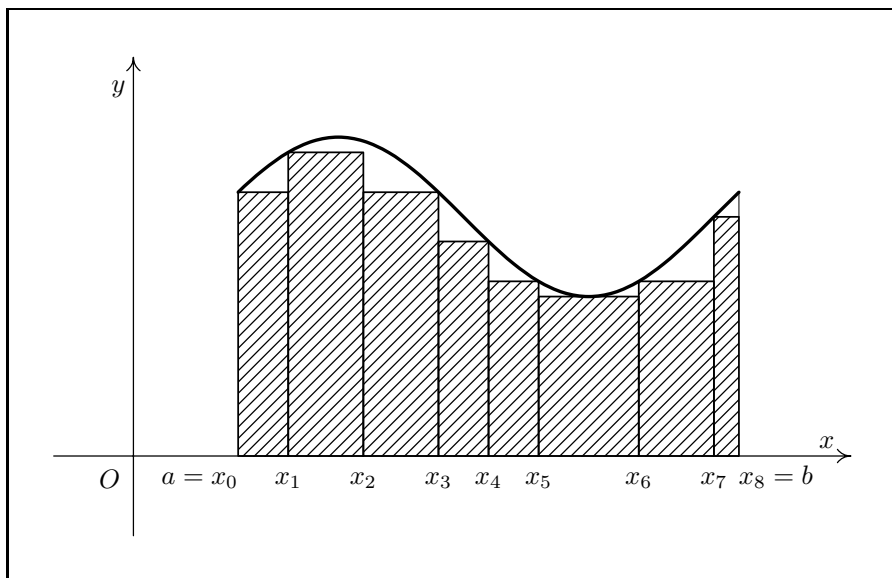
```

\lines{(-3,-2),(-3,2)}
\lines{(0,1),(2*pi,1)}
\lines{(0,-1),(2*pi,-1)}
\lines{(.5*pi,-1),(.5*pi,1)}
\lines{(pi,-1),(pi,1)}
\lines{((3/2)*pi,-1),((3/2)*pi,1)}
\lines{(2*pi,-1),(2*pi,1)}
\dashed\lines{((5/2)*pi,-1),((5/2)*pi,1)}
\dashed\lines{(3*pi,-1),(3*pi,1)}
\def{f}{x}{sin x}
\dashed\function{-pi/5,3.1*pi,.1*pi}{f(x)}
\pen{1pt}
\function{0,2*pi,.1*pi}{f(x)}
\pen{.5pt}
\circle{(-3,0),1}
\arrow[r-5]\arc[p]{(-3,0),0,30,.75}
\arrow[r -5]\arc[p]{(-3,0),0,240,.75}
\lines{(-3,0),(-3+cosd 30,sind 30)}
\lines{(-3,0),(-3-cosd 240,sind 240)}
\dashed\lines{(-3,1),(0,1)}
\dashed\lines{(-3,-1),(0,-1)}
\dashed\lines{(-3+cosd 30,sind 30),(pi/6,f(pi/6))}
\dashed\lines{(pi/6,0),(pi/6,f(pi/6))}
\dashed\lines{(-3-cosd 240,sind 240),((4/3)*pi,f((4/3)*pi))}
\dashed\lines{((4/3)*pi,0),((4/3)*pi,f((4/3)*pi))}
\tlabel[tr]{-4.1,-.1}{\$-4\$}
\tlabel[tl]{-2,-.1}{\$-2\$}
\tlabel[tl]{.1,-.1}{\$0\$}
\tlabel[tl]{(.5*pi)+.1,-.1}{\${1\over 2} \pi\$}
\tlabel[bl]{(1*pi+.1,0.2)}{\$\pi\$}
\tlabel[bl]{((3/2)*pi+.1,.2)}{\${3\over 2} \pi\$}
\tlabel[tl]{(2*pi+.1,-.1)}{\$2\pi\$}
\tlabel[tl]{((5/2)*pi+.1,-.1)}{\${5\over 2} \pi\$}
\tlabel[tl]{(3*pi+.1,-.1)}{\$3\pi\$}
\mfsrc{picture a; a~:= currentpicture;
      currentpicture:=nullpicture;
      draw a~rotated 90;}%
\endmfpic

```

### 18.3. Určitý integrál a jeho geometrické aplikace

Znázornění dolního integrálního součtu příslušného určitému dělení intervalu  $\langle a, b \rangle$ .



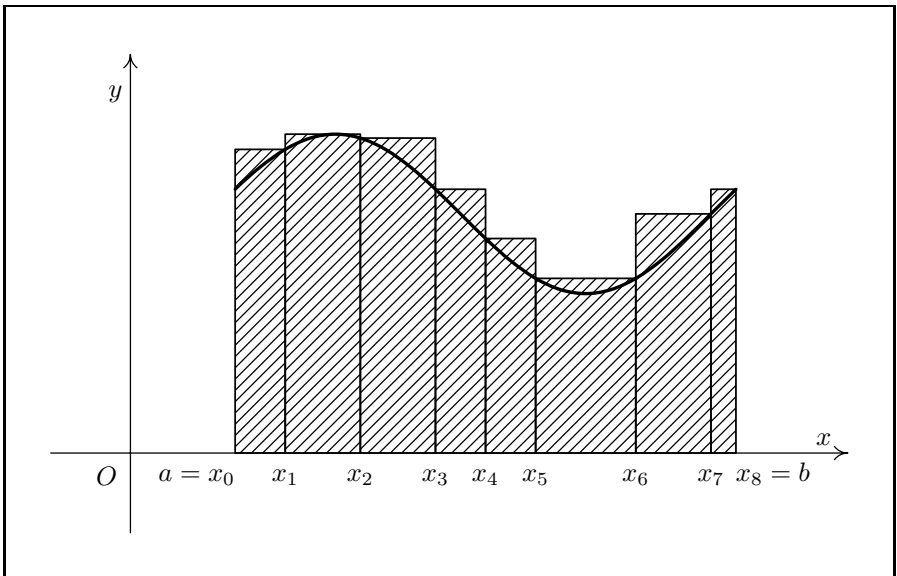


```

\mfpicunit=30pt
\mfpic[1]{-2}{8}{-1}{5}
\xaxis
\headlen=5pt          %% musíme změnit přednastavenou
\arrow\lines{(-1,-1),(-1,5)} %% délku, neboť je jiná než
\tlabel[cr](-1.1,4.6){$y$} %% u makra \xaxis; nebo lze
\tlabel[bc](7.7,0.1){$x$} %% \arrow[1 5pt]\lines{...}
\tlabel[bc](-1.3,-0.4){$0$}
\def{f}{x}{(sin x)+3}
\pen{1.3pt}
\function{.1*pi,2.1*pi,.1*pi}{f(x)}
\pen{.7pt}
\draw\rhatch\rect{(.1*pi,f(.1*pi)),(.3*pi,0)}
\draw\rhatch\rect{(.3*pi,f(.3*pi)),(.6*pi,0)}
\draw\rhatch\rect{(.6*pi,f(.9*pi)),(.9*pi,0)}
\draw\rhatch\rect{(.9*pi,f(1.1*pi)),(1.1*pi,0)}
\draw\rhatch\rect{(1.1*pi,f(1.3*pi)),(1.3*pi,0)}
\draw\rhatch\rect{(1.3*pi,f(1.5*pi)),(1.7*pi,0)}
\draw\rhatch\rect{(1.7*pi,f(1.7*pi)),(2*pi,0)}
\draw\rhatch\rect{(2*pi,f(2*pi)),(2.1*pi,0)}
\pen{.3pt}
\lines{(2.1*pi,f(2*pi)),(2.1*pi,f(2.1*pi))}
\tlabel[br](.1*pi,-0.4){$a=x_0$}
\tlabel[bc](.3*pi,-0.4){$x_1$}
\tlabel[bc](.6*pi,-0.4){$x_2$}
\tlabel[bc](.9*pi,-0.4){$x_3$}
\tlabel[bc](1.1*pi,-0.4){$x_4$}
\tlabel[bc](1.3*pi,-0.4){$x_5$}
\tlabel[bc](1.7*pi,-0.4){$x_6$}
\tlabel[bc](2*pi,-0.4){$x_7$}
\tlabel[bl](2.1*pi,-0.4){$x_8=b$}
\endmpic

```

Znázornění horního integrálního součtu příslušného určitému dělení intervalu  $\langle a, b \rangle$ .

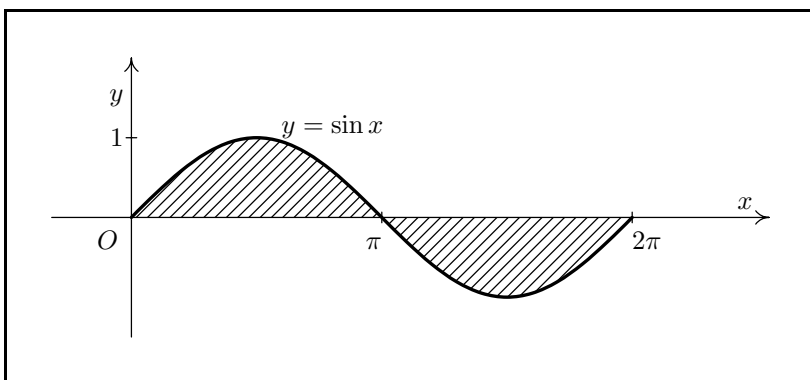
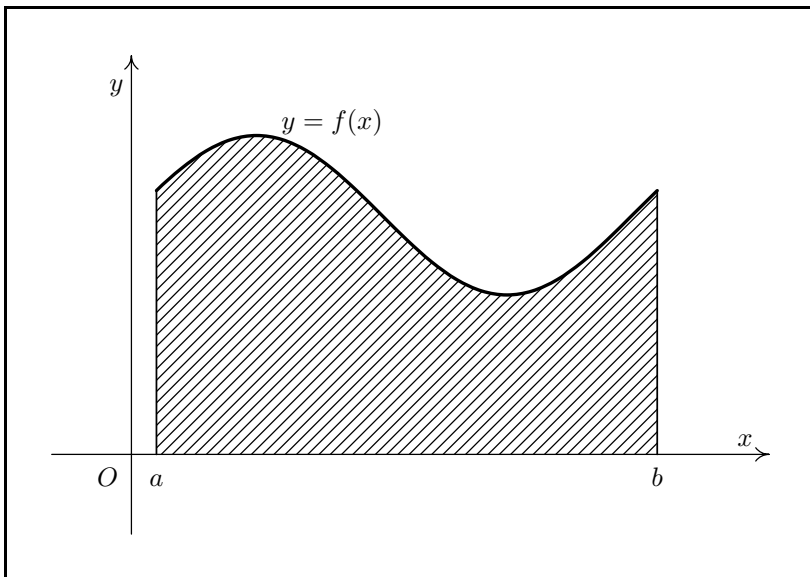


```

\mfpic[1]{-2}{8}{-1}{5}
\mfpicunit=30pt
\xaxis
\headlen=5pt          %% pro dosažení stejné délky
\arrow\lines{(-1,-1),(-1,5)} %% šipky jako v makru \xaxis;
\tlabel[cr](-1.1,4.6){$y$} %% nebo lze použít:
\tlabel[bc](7.7,0.1){$x$} %% \arrow[1 5pt]\lines{...}
\tlabel[bc](-1.3,-0.4){$0$}
\def{f}{x}{(sin x)+3}
\pen{1.3pt}
\function{.1*pi,2.1*pi,.1*pi}{f(x)}
\pen{.7pt}
\draw\rhatch\rect{(.1*pi,f(.3*pi)),(.3*pi,0)}
\draw\rhatch\rect{(.3*pi,f(.5*pi)),(.6*pi,0)}
\draw\rhatch\rect{(.6*pi,f(.6*pi)),(.9*pi,0)}
\draw\rhatch\rect{(.9*pi,f(.9*pi)),(1.1*pi,0)}
\draw\rhatch\rect{(1.1*pi,f(1.1*pi)),(1.3*pi,0)}
\draw\rhatch\rect{(1.3*pi,f(1.7*pi)),(1.7*pi,0)}
\draw\rhatch\rect{(1.7*pi,f(2*pi)),(2*pi,0)}
\draw\rhatch\rect{(2*pi,f(2.1*pi)),(2.1*pi,0)}
\pen{.5pt}
\tlabel[br](.1*pi,-0.4){$a=x_0$}
\tlabel[bc](.3*pi,-0.4){$x_1$}
\tlabel[bc](.6*pi,-0.4){$x_2$}
\tlabel[bc](.9*pi,-0.4){$x_3$}
\tlabel[bc](1.1*pi,-0.4){$x_4$}
\tlabel[bc](1.3*pi,-0.4){$x_5$}
\tlabel[bc](1.7*pi,-0.4){$x_6$}
\tlabel[bc](2*pi,-0.4){$x_7$}
\tlabel[bl](2.1*pi,-0.4){$x_8=b$}
\endmfpic

```

Geometrický význam určitého integrálu  $\int_a^b f(x) dx$  funkce  $f$  a náčrt obsahu obrazce ohraničeného grafem funkce  $f: y = \sin x$  v intervalu  $\langle 0, 2\pi \rangle$  a osou  $x$ .



```

\mfpic[1]{-1}{8}{-1}{5}
\mfpicunit=30pt
\axes
\tlabel[cr](-0.1,4.6){$y$}
\tlabel[bc](7.7,0.1){$x$}
\tlabel[bc](-0.1,-0.4){$0$}
\def{f}{x}{(sin x)+3}
\pen{1.3pt}
\function{.1*pi,2.1*pi,.1*pi}{f(x)}
\pen{.5pt}
\rhatch\btwnfcn{.1*pi,2.1*pi,.1*pi}{0}{f(x)}
\pen{.7pt}
\lines{(.1*pi,0),( .1*pi,f(.1*pi))}
\lines{(2.1*pi,0),(2.1*pi,f(2.1*pi))}
\tlabel[bl](.6*pi,4){$y=f(x)$}
\tlabel[bc](.1*pi,-0.4){$a$}
\tlabel[bc](2.1*pi,-0.4){$b$}
\endmfpic

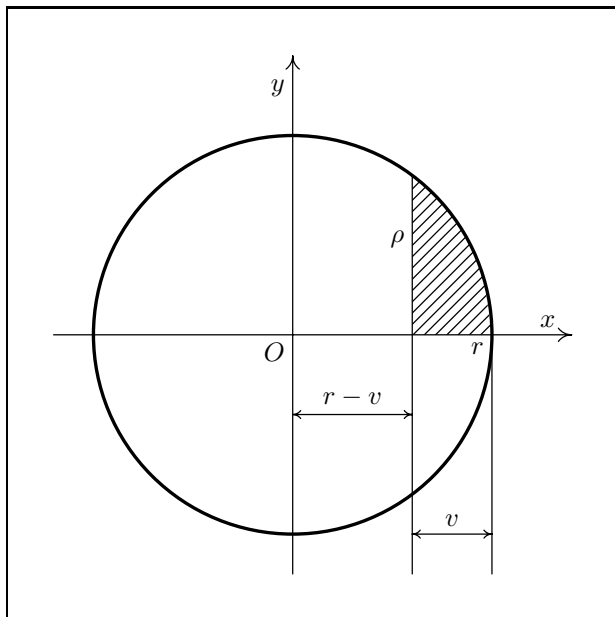
```

```

\mfpic[1]{-1}{8}{-1.5}{2}
\mfpicunit=30pt
\axes
\ymarks{1}
\xmarks{pi,2*pi}
\tlabel[cr](-0.1,1.5){$y$}
\tlabel[bc](7.7,0.1){$x$}
\tlabel[bc](-.3,-.4){$0$}
\def{f}{x}{sin x}
\pen{1.3pt}
\function{0,2*pi,.1*pi}{f(x)}
\pen{.7pt}
\rhatch\btwnfcn{0,2*pi,.1*pi}{0}{f(x)}
\pen{.5pt}
\tlabel[br](pi,-.4){$\pi$}
\tlabel[bl](2*pi,-.4){$2\pi$}
\tlabel[cr](-.1,1){$1$}
\tlabel[bl](.6*pi,1){$y=\sin{x}$}
\endmfpic

```

Náčrt k odvození vzorce pro objem kulové úseče výšky  $v$ , která je vyřata z koule o poloměru  $r$ .

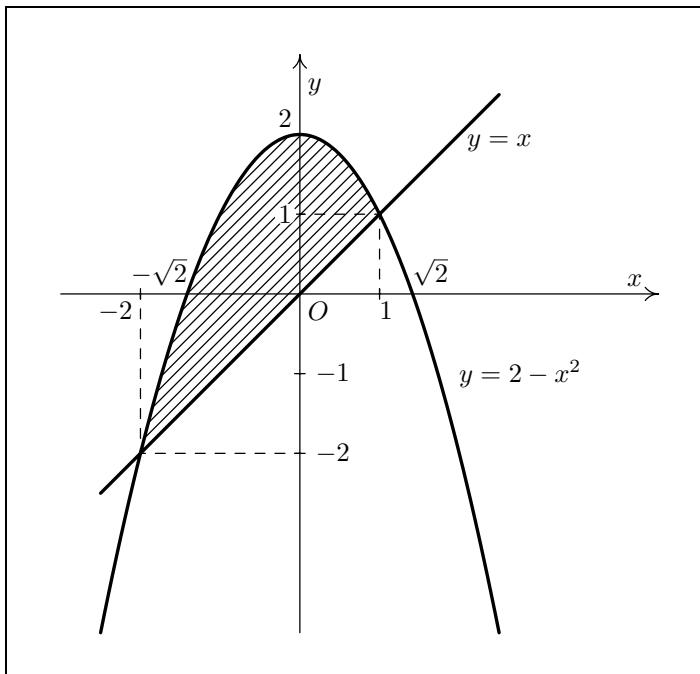


```

\mfpic[1]{-3}{3.5}{-3}{3.5}
\mfpicunit=30pt
\axes
\tlabel[cr](-.1,3.1){$y$}
\tlabel[bc](3.2,.1){$x$}
\tlabel[tr](-.1,-.1){$0$}
\pen{1.3pt}
\circle{(0,0),2.5}
\pen{.5pt}
\lines{(1.5,sqrt((2.5*2.5)-(1.5*1.5))),(1.5,-3)}
\lines{(2.5,0),(2.5,-3)}
\arrow\reverse\arrow\lines{(0,-1),(1.5,-1)}
\arrow\reverse\arrow\lines{(1.5,-2.5),(2.5,-2.5)}
\pen{.7pt}
\rhatch\btwnfcn{1.5,2.5,.1}{0}{sqrt((2.5*2.5)-(x*x))}
\tlabel[cr](1.4,1.2){$\rho$}
\tlabel[tr](2.4,-.1){$r$}
\tlabel[bc](.75,-.9){$r-v$}
\tlabel[bc](2,-2.4){$v$}
\endmfpic

```

Obsah obrazce ohraničeného grafy funkcí  $f: y = 2 - x^2$ ,  $g: y = x$ .





```

\def\obd#1{%
%% makro pro odbarvení plochy
\mfsrc{picture obr; obr = image()} %% pod textem ve tvaru kruhu;
\tlabel#1%
%% příkaz center určuje střed
\mfsrc{}; unfill fullcircle
%% bounding boxu
scaled (2*(abs( center obr -
llcorner obr)))
shifted (center obr);
draw obr;}%

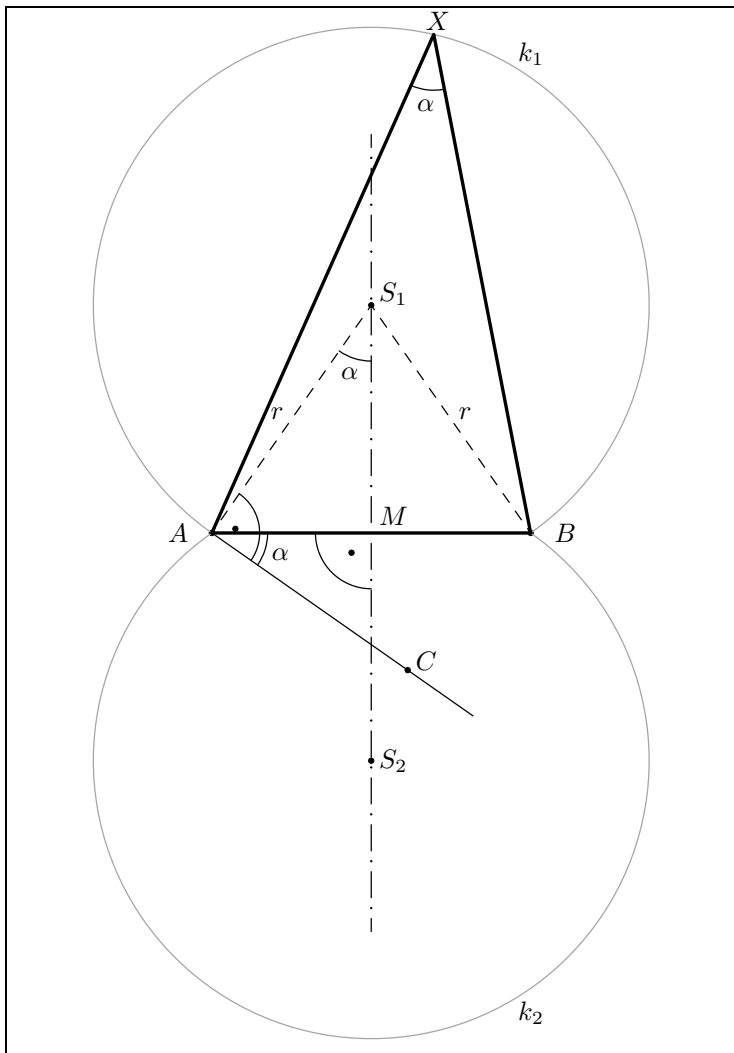
\mpic[1]{-3}{4.5}{-4.25}{3}
\mpicunit=30pt
\axes
\xmarks{-2,1}
\ymarks{-2,-1,1}
\tlabel[cl](0.1,2.6){$y$}
\tlabel[bc](4.2,.1){$x$}
\tlabel[tl](.1,-.1){$0$}
\def{f}{x}{x}
\def{g}{x}{2-(x*x)}
\pen{1.3pt}
\function{-2.5,2.5,.1}{f(x)}
\function{-2.5,2.5,.1}{g(x)}
\pen{.7pt}
\rhatch\btwnfcn{-2,1,.1}{g(x)}{f(x)}
\pen{.5pt}
\dashed\lines{(-2,0),(-2,f(-2))}
\dashed\lines{(0,-2),(-2,f(-2))}
\dashed\lines{(1,0),(1,f(1))}
\dashed\lines{(0,1),(1,f(1))}
\tlabel[tl](2.1,2){$y=x$}
\tlabel[cl](2,-1){$y=2-x^2$}
\tlabel[tr](-2.1,-.1){$-2$}
\tlabel[br](-sqrt 2,.1){$-\sqrt{2}$}
\tlabel[tl](1,-.1){1}
\tlabel[bl](sqrt 2,.1){$\sqrt{2}$}
\tlabel[cl](0.2,-2){$-2$}
\tlabel[cl](0.2,-1){$-1$}
\obd{[cr](-.1,1){1}}
\tlabel[cr](-.1,2.2){2}
\endmpic

```

### 18.4. Množiny bodů dané vlastnosti

Množina všech bodů  $X$ , z nichž vidíme úsečku  $AB$  pod úhlem  $\alpha$ .

U tohoto obrázku si všimněme zejména vyznačování úhlů a tvorby čerchované čáry.



```
\mfpicunit=30pt
\mfpic[1]{-4}{4}{-6}{6}
\mfsrc{pair A,B,C,D,S,O,X,Y,Z;
A=(-2,0); B=(2,0);      alpha:=35;
```

```

C=3*dir(-alpha)+A;      D=4*dir(-alpha)+A;
S=A+whatever*dir(90-alpha);  S=whatever*up;  O=-S;}
\lines{A,B}      \lines{A,D}      \point{A,B,C,S,O}
\dashed\lines{A,S}  \dashed\lines{B,S}
\dashpattern{cerchovana}{10pt,4pt,0pt,4pt}
\gendashed{cerchovana}\lines{(0,-5),(0,5)}
\point{.3*dir(45-alpha)+A}
\arc[p]{A,-alpha,90-alpha,.6}  \arc[p]{A,-alpha,0,.7}
\point{(.35*dir 225)}
\arc[p]{.5[A,B],180,270,.7}  \arc[p]{S,270-alpha,270,.7}
\draw[.65white]\arc[p]{S,alpha-90,270-alpha,abs(S-A)}
\coords      \mirror{A}{B}
\draw[.65white]\arc[p]{S,alpha-90,270-alpha,abs(S-A)}
\endcoords
\mfsrc{X=S+abs(S-A)*dir(77);}
\pen{1.3pt}      \polygon{A,B,X}      \pen{.5pt}
%\mfsrc{path a;  %% vyznačení úhlu lze nakreslit také takto
%a=fullcircle scaled 42pt shifted zconv (X);
%draw subpath (ypart (zconv ((X--A)) intersectiontimes a),
%ypart(zconv((X--B)) intersectiontimes a)) of a;}%
\mfsrc{x=angle(X-A); y=angle(X-B);}
\arc[p]{X,x+180,y+180,.7}
\tlabel[cr](xpart A~-.3,ypart A){$A$}
\tlabel[cl](xpart B+.3,ypart B){$B$}
\tlabel[bl](xpart C+.1,ypart C){$C$}
\tlabel[bl](xpart S~+.1,ypart S){$S_1$}
\tlabel[cl](xpart O~+.1,ypart O){$S_2$}
\tlabel[bl](0.1,0.1){$M$}
\tlabel[bc](xpart X+0.05,ypart X+0.05){$X$}
\tlabel[br](xpart .5[A,S]-.1,ypart .5[A,S]){$r$}
\tlabel[bl](xpart .5[B,S]+.1,ypart .5[A,S]){$r$}
\mfsrc{Y=S+abs(S-A)*dir(60);}
\tlabel[bl](xpart Y+.1,ypart Y){$k_1$}
\tlabel[tl](xpart Y+.1,-ypart Y){$k_2$}
\mfsrc{pair f,g,h;f=.9*dir((x+y)/2+180)+X;
      g=.9*dir(-alpha/2)+A;
      h=.9*dir(270-(alpha/2))+S;}
\tlabel[cc](xpart f,ypart f){$\alpha$}
\tlabel[cc](xpart g,ypart g){$\alpha$}
\tlabel[cc](xpart h,ypart h){$\alpha$}
\endmpic

```

## Literatura

- [1] Adobe Systems Incorporated *PostScript Language Reference Manual*. Massachusetts: Addison-Wesley, 3. vydání, 1999. ISBN 0-201-37922-8  
<http://adobe.com:80/products/postscript/pdfs/PLRM.pdf>
- [2] Hobby, J. D. *A User's Manual for MetaPost*. Součást dokumentace programu METAPOST.  
<http://cm.bell-labs.com/cm/cs/cstr/162.ps.gz>
- [3] Knuth, D. E. *The METAFONTbook*. Massachusetts: Addison-Wesley, 1986. ISBN 0-201-13445-4
- [4] Kuben, J. *Diferenciální počet funkcí jedné proměnné*. Brno: VA, 1999
- [5] Kuben, J. *Zpravodaj Československého sdružení uživatelů T<sub>E</sub>Xu 2/94*. Brno.  
<http://bulletin.cstug.cz/pdf/bul942.pdf>
- [6] Olšák, P. *T<sub>E</sub>Xbook naruby*. Brno: Konvoj, 2. vydání, 2001. ISBN 80-7302-007-6  
<ftp://math.feld.cvut.cz/pub/olsak/tbn/tbn.pdf>
- [7] Polák, J. *Přehled středoškolské matematiky*. Praha: Prometheus, 2000. ISBN 80-7196-196-5
- [8] rybicka Rybička, J. *L<sup>A</sup>T<sub>E</sub>X pro začátečníky*. Brno: Konvoj, 2. vydání, 1999. ISBN 80-85615-74-6

Zajímavé odkazy týkající se tématu:

- <http://cm.bell-labs.com/who/hobby/MetaPost.html>
- <http://comp.uark.edu/~luecking/tex/mfpic.html>
- <http://ftp.cstug.cz/pub/tex/CTAN/graphics/mfpic/>  
(bohužel zatím je zde pouze starší verze)
- <http://ftp.cstug.cz/pub/tex/CTAN/systems/knuth/mf/mfbook.tex>

## Poznámka

Tento článek je věnován `mfpic` verzi 0.4.05, poslední dostupné verzi v době uzávěrky Zpravodaje. Již brzy bude dostupná verze `mfpic` 0.5.0. Nové vlastnosti budou popsány v dodatku článku v následujícím čísle Zpravodaje.

## Rejstřík

`active_plane`, 109  
`\applyT`, 107  
`\arc ...`, 77  
    `\arc [c]`, 77  
    `\arc [p]`, 77  
    `\arc [s]`, 77  
    `\arc [t]`, 77  
`\arrow`, 72, 73, 78, 86  
`\axes`, 72, 73, 78  
`\axisheadlen`, 72, 78  
  
`\bclosed`, 87  
`\begin{connect}`, 89  
`\begin{coords}`, 91  
`\begin{mpic}`, 70  
`\begin{patharr}`, 107  
`\begin{tile}`, 85  
`\boost`, 90  
`\btwnfcn`, 87, 97  
  
`\cbclosed`, 88  
    `centeredcaptions`, 71  
`\circle`, 76, 87  
    `clip`, 71, 72  
`\clipmpic`, 71  
    `clipped`, 108  
    `clipsto`, 112  
    `clipto`, 108  
`\closegraphsfile`, 70  
`\coloredlines`, 101  
`\connect`, 89  
`\coords`, 91  
`\curve`, 76, 77  
`\cyclic`, 77, 87  
  
`\dashed`, 72, 80, 95  
`\dashedlines`, 101  
`\dashlineset`, 72  
`\dashlen`, 72, 80  
  
`\dashpattern`, 82  
`\dashspace`, 72, 80  
`\datafile`, 100, 101  
`\datapointsonly`, 101  
    `debug`, 71  
`\dotlineset`, 72  
`\dotsize`, 73, 80  
`\dotspace`, 73, 80  
`\dotted`, 73, 80  
`\draw`, 80, 95  
`\drawcolor`, 74  
`\drawpen`, 73  
  
`\ellipse`, 76, 87  
`\endconnect`, 89  
`\end{connect}`, 89  
`\endcoords`, 91  
`\end{coords}`, 91  
`\endmpic`, 70, 73, 101  
`\end{mpic}`, 70  
`\endpatharr`, 107  
`\end{patharr}`, 107  
`\endtile`, 85  
`\end{tile}`, 85  
  
`\fcncurve`, 77  
`\fdef`, 96  
`\fillcolor`, 74  
`\function`, 97  
funkce  
    `abs`, 96  
    `acos`, 96  
    `acosh`, 97  
    `asin`, 96  
    `asinh`, 97  
    `atan`, 96  
    `atanh`, 97  
    `ceiling`, 96  
    `cos`, 96

cosd, 96	invzconv, 108
cosh, 96	
cot, 96	\lclosed, 87
cotd, 96	\lhatch, 83
csc, 96	\lines, 75
cscd, 96	
exp, 96	metafont, 71
floor, 96	metapost, 71
invcos, 96	\mfobj, 79, 107
invsin, 96	\mfpdatacomment, 100, 101
invtan, 96	\mfdefinecolor, 74
ln, 97	\mfpic, 70, 71, 73
max, 96	\mfpicdebugfalse, 71
mexp, 96	\mfpicdebugtrue, 71
min, 96	\mfpicunit, 72
mlog, 96	\mfpicwidth, 73
round, 96	\mfpicheight, 73
sec, 96	\mfplinestyle, 102
secd, 96	\mfpverbtext, 114, 116
sin, 96	\mfsrc, 107, 113
sind, 96	\mirror, 89
sinh, 96	mplabels, 71, 72, 105
sqrt, 96	
tan, 96	\nocenteredcaptions, 71
tand, 96	\noclipmfpic, 71
tanh, 97	\nomplabels, 71
	\notruebbox, 71
\gclear, 83	
\gendashed, 82	\opengraphsfile, 70, 71
\gfill, 83, 95	
\grid, 75	\parafcn, 97
	\patharr, 107
\hashlen, 73, 78	\pen, 73
\hatch, 73, 83	\plot, 72, 73, 81, 101
\hatchcolor, 74	\plotdata, 101, 102
\hatchspace, 73, 83	\plotnodes, 81, 101
\hatchwd, 73, 83	\plotsymbol, 72, 75, 81, 101
\headcolor, 74	\plr, 78
\headlen, 72	\plrfcn, 97
\headshape, 73	\plrregion, 87, 97
	\point, 72, 75
invvconv, 108	\pointdef, 74

<code>\pointedlines</code> , 101	<code>\tlabel</code> , 71, 103
<code>\pointfillfalse</code> , 72, 75	<code>truebbox</code> , 71, 72
<code>\pointfilltrue</code> , 72, 75	<code>\turn</code> , 89
<code>\pointsize</code> , 72, 75, 81	<code>\turtle</code> , 79
<code>\polkadot</code> , 73, 84	
<code>\polkadotSPACE</code> , 73, 84	<code>\uclosed</code> , 88
<code>\polkadotwd</code> , 73, 84	<code>\unsmoothdata</code> , 100
<code>\polygon</code> , 76, 87	<code>\usecenteredcaptions</code> , 71
<code>\polyline</code> , 75	<code>\usemetafont</code> , 71
	<code>\usemetapost</code> , 71
<code>\rect</code> , 76, 87	<code>\usemplabels</code> , 71
<code>\reflectabout</code> , 89	<code>\usetruebbox</code> , 71
<code>\reverse</code> , 87	<code>\using</code> , 101
<code>\rhatch</code> , 83	
<code>\rotate</code> , 89	<code>vconv</code> , 108
<code>\rotatearound</code> , 89	
	<code>\xaxis</code> , 72, 73, 78
<code>\scaled</code> , 89	<code>\xhatch</code> , 83
<code>\sclosed</code> , 88	<code>\xmarks</code> , 73, 78
<code>\sector</code> , 79, 87	<code>\xscale</code> , 90
<code>\sequence</code> , 103	<code>\xslant</code> , 90
<code>\setrender</code> , 95	<code>\xyswap</code> , 90
<code>\shade</code> , 83	
<code>\shift</code> , 89	<code>\yaxis</code> , 72, 73, 78
<code>\smoothdata</code> , 100	<code>\ymarks</code> , 73, 78
<code>\store</code> , 79, 108, 109	<code>\yscale</code> , 90
<code>\symbolSPACE</code> , 73, 81	<code>\yslant</code> , 90
<code>\tcaption</code> , 71, 105	<code>zconv</code> , 108
<code>\tess</code> , 85	<code>\zscale</code> , 90
<code>\thatch</code> , 83	<code>\zslant</code> , 90
<code>\tile</code> , 85	

## Summary: METAPOST and mfpic—the second part

Although METAPOST is not hard to learn, most users prefer the `mfpic` macro package which can be used in plain,  $\text{\LaTeX}$  and  $\text{\pdfTeX}$ . This article presents almost all features of this macro package in examples (full source code for each is included).