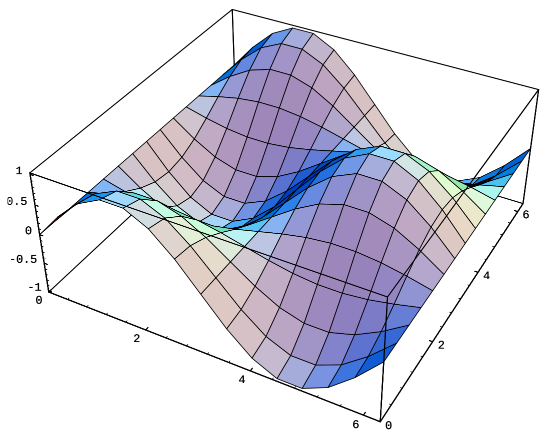
## Jak dostat obrázky z programu *Mathematica* do TEXu

Program *Mathematica* Stephena Wolframa má charakteristický dovětek ke svému názvu: „System for doing Mathematics by computer". Tím je zhruba řečeno, o co jde. Dále je známo, že TEX je „sazeč" textů, zvláště technických, např. s komplikovanou matematikou. Z toho vyplývá, že uživatelé TEXu se často stávají i uživateli programu *Mathematica* a obráceně. Každý, kdo se stane uživatelem obou systémů, si zcela zákonitě položí otázku, formulovanou v názvu tohoto článku. Také já jsem si ji položil a musím přiznat, že odpověď se mi hledala nesnadno.

Jako příklad uvažujme reálnou funkci dvou proměnných $f(x) = \sin(x)\cos(y)$. Studentům většinou říkám, že představu o této funkci získají tím, že si prostudují formu na vajíčka. Druhá možnost je, nechat si tuto funkci znázornit programem *Mathematica*. Spusťme program *Mathematica*, který nám nabídne prompt ve tvaru `In[1]:=` . Napíšeme požadavek zobrazení naší funkce třeba takto:

```
In[1]:= Plot3D[Sin[x] Cos[y], {x,0,2Pi}, {y,0,2Pi}]
```

Po odeslání tohoto požadavku se nám na obrazovce objeví obrázek:




Chceme-li obrázek uložit do souboru, napíšeme v programu *Mathematica* příkaz

*In[2]:=* `Display ["obraz.g", %1]`

Příkaz způsobí, že se obrázek, vytvořený předchozím (prvním) příkazem uloží do souboru se jménem `obraz.g` . Pokud máme připojenou tiskárnu, můžeme ještě zkusit obrázek vytisknout. Provedeme to například příkazem

*In[3]:=* `Hardcopy [%1]`

K počítači jsem měl připojenu laserovou tiskárnu bez PostScriptu a po vytisknutí obrázku jsem se nestačil divit*. V tisku byly dosti podstatné chyby. Obrázek se rozdělil na jakési vodorovné pásy a ty na sebe vůbec nenavazovaly! Pokud jsem tentýž pokus udělal na maticové tiskárně (24 jehliček), dopadlo to ještě katastrofálněji.

Pro zařazení obrázku do TEXu potřebujeme spíš vědět, jak dopadlo uložení obrázku do souboru. V programu *Mathematica* proto ukončíme činnost pomocí příkazu

*In[4]:=* `Quit`

a podíváme se do souboru `obraz.g`. Jedná se o textový soubor, v němž je obrázek popsán jazykem PostScript. Tento soubor ale nelze poslat bez doplňujících údajů ani na tiskárnu vybavenou PostScriptem, protože mu chybí hlavička (soubor definic), které jazyk PostScriptu potřebuje načíst dřív, než bude zpracovávat tento obrázek.

Podívejme se proto podrobněji, jak funguje tisk obrázků z programu *Mathematica* pro DOS. Příkazem `Hardcopy` se nejdříve vytvoří přechodný soubor s podobným obsahem, jako je soubor `obraz.g`, a pak se z prostředí programu *Mathematica* spustí dávka s názvem `hardcopy.bat`, která může mít třeba tento obsah (je to závislé na instalaci)

```
@echo off
printps -printer hplj.300 -dev LPT1 -fonts c:\math\fonts %1
```

Je vidět, že se vlastně spustí program `printps.exe` s příslušnými parametry. V případě, že tiskárna nemá PostScript (parametr `hplj`), funguje

* Program *Mathematica* jsem měl možnost zkoušet ve verzi 2.0 pro DOS a pro UNIX (Open Windows). Více jsem se zaměřil na verzi pro DOS, a proto všechny následující technické vlastnosti se týkají této verze.

program `printps.exe` jako interpret PostScriptu (jak se ukazuje z pokusů použití příkazu `Hardcopy`, funguje v dané verzi jako *špatný* interpret). Pokud je tiskárna vybavena PostScriptem, program `printps.exe` pouze připojí ke vstupnímu souboru hlavičku s definicemi příkazů PostScriptu speciálně používaných programem *Mathematica* a soubor pouze „překopíruje“ na tiskárnu.

My budeme pro zařazení obrázku do TEXu potřebovat vytvořit PostScriptový soubor, který hlavičku s definicemi obsahuje. Přirozeně, že k tomu použijeme program `printps.exe`, který zavoláme například tímto příkazem:

```
printps -eps obraz.eps -fonts c:\math\fonts  obraz.g
```

Tento příkaz vytvoří ze vstupního textového souboru `obraz.g` výstupní soubor `obraz.eps`, který již je opatřen příslušnými hlavičkami. Takovému formátu se říká „Encapsulated PostScript Form“. Samozřejmě, pokud zpracováváme více obrázků, je vhodné použít například dávku s obsahem:

```
printps -eps %1.eps -fonts c:\math\fonts  %1.g
```

Skutečnost, že soubor `obraz.eps` již obsahuje hlavičku s definicemi, poznáme podle začátku souboru, který vypadá takto:

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 540 335
%%Creator: Mathematica
...
```

Všechny tyto úvodní řádky jsou komentáře jazyka PostScript, protože začínají procenty. (Kým byli asi programátoři firmy Adobe inspirováni, když navrhovali jazyk PostSriptu?) Zvláště charakteristický je druhý řádek s textem `BoundingBox`, kde je uveden rozměr výšky a šířky obrázku. Tuto informaci bude dále potřebovat software, který použijeme pro zařazení obrázku do textu.

Napišme nyní zdrojový text TEXu, který může vypadat třeba takto

```
\input epsf
...
... se na obrazovce objeví obrázek:

\centerline{\epsfysize=6cm \epsfbox{obraz.eps}}

\noindent Chceme-li ...
```

Makro `epsf.tex` obsahuje definice příkazů `\epsfbox`, `\epsfysise` a dalších. Toto makro načítá z úvodních řádků souboru `obraz.eps` parametry příkazu `BoundingBox`, z nichž zjistí poměr výšky a šířky obrázku. Na základě těchto údajů a požadované výšky obrázku (v příkazu `\epsfysize`) vypočítá rozměry boxu `\epsfbox` a tento box uloží do `.dvi` souboru. Můžeme také zadat šířku boxu pomocí `\epsfxsize`, přičemž výška se dopočítá automaticky. Do souboru `.dvi` se také uloží prostřednictvím příkazu `\special` název souboru s obrázkem (v našem případě `obraz.eps`) a PostScriptová instrukce na zvětšení nebo zmenšení obrázku do požadovaných rozměrů. Zmíněné makro je součástí programového balíku `dvips`. Je tam zdokumentováno i s příkladem použití.

Po zpracování TEXem si výsledek můžeme prohlédnout Mattesovým prohlížečem. Místo obrázku ovšem uvidíme jen prázdné místo, protože příslušný příkaz `\special` tento prohlížeč neakceptuje.

Máme-li „odladěný“ text dokumentu, můžeme přikročit k vlastnímu zařazení obrázku. K tomu použijeme program `dvips.exe`, který načítá soubor `.dvi` a vytváří z něj soubor `.ps`, ve kterém jsou PostScriptové popisy jednotlivých stránek dokumentu. Tento program dále v místě výskytu příkazu `\special` vloží do výstupního souboru kopii souboru `obraz.eps`, případně překopíruje příslušné PostScriptové instrukce, uvedené v příkazu `\special`.

Program `dvips.exe` potřebuje navíc znát, kde jsou uloženy metriky a bitové mapy fontů, používaných v TEXu. Tyto informace převede do PostScriptového popisu stránek. Příslušné adresáře sdělíme programu `dvips.exe` prostřednictvím jeho konfiguračního souboru `config.ps`, v němž je například napsáno:

```
* Claim 180kbytes memory
m 180000
* Default resolution.
D 300
* Paths
T e:\emtex\tfm
P e:\emtex\pixel300\dpi%d\%f.%p
*V e:\texfonts\vf
*L c:\texfonts\;lj_0;lj_h;lj_1;lj_2;lj_3;lj_4;lj_5a;lj_5b;lj_sli
*S .;e:\emtex\texinput
*H .;e:\emtex\ps
...
```

Další údaje nás pro začátek nemusí zajímat. Hvězdičkou je označen komentářový řádek. Znamená to, že jsme prostřednictvím příkazů `T` a `P` označili adresáře pro metriky a soubory `.pk`, zatímco ostatní řádky jsou „odkomentovány“ a slouží jako příklad, co lze ještě v konfiguračním souboru napsat. Jedná se například o adresáře k virtuálním fontům, knihovnám fontů apod.

Nyní bychom si chtěli prohlédnout na obrazovce, jak vypadá text i s obrázkem. K tomu lze použít public domain interpret PostScriptového popisu stránek, jehož výstupem může být jak obrazovka, tak tiskárna bez PostScriptu. Jmenuje se GhostScript. Pro prohlížení na obrazovce stačí napsat

```
gs386 soubor.ps > nul
```

kde `soubor` je název dokumentu. Přesměrování textových hlášení programu do `nul` je provedeno proto, aby se hlášení programu nemíchalo s grafikou (což se normálně děje).

Jsme-li rozmazleni komfortem Mattesova ovladače, asi nás to trošičku zklame. Lze skákat pouze po jednotlivých stránkách a zvětšení je pouze jediné — takové, aby na obrazovce byla vidět celá stránka textu. Všichni víme, že při takovém zvětšení se text nedá číst, a proto je nutné mít text odladěn už dříve. Zato obrázek vidíme dokonce barevně (máme-li barevnou grafiku).

V tuto chvíli jsem získal další negativní zkušenost. Obrázek jsem sice viděl v požadované velikosti společně s textem, ale nebyl na požadovaném místě. Byl poněkud výše. Tady se projevila další chyba drahého programu *Mathematica*. Konvertor obrázků programu *Mathematica* `printps.exe` nenastaví správné hodnoty parametru `BoundingBox`! Problém jsem vyřešil vytvořením krátkého testovacího „programu“ v TEXu s tímto zněním:

```
\input epsf
\vglue5cm
\vbox{\hrule\hbox{\vrule\epsfbox{obraz.eps}\vrule}\hrule}
\bye
```

Po spuštění programů `tex.exe`, `dvips.exe`, `gs386.exe` jsem na obrazovce viděl rámeček pro obrázek a navíc obrázek, který byl poněkud vpravo a nahoře. Nyní stačí editovat parametry `BoundingBox` v souboru `obraz.eps`. První dva parametry jsou souřadnice levého dolního rohu rámečku a druhé dva označují pravý horní roh. Zvětšováním souřadnice $x$ se posunujeme doprava a zvětšováním $y$ se posunujeme nahoru. Souřadnice jsou uvedeny v jednotkách `bp`, což je skoro totéž, co `pt`.

Po změně parametrů znova použijeme výše uvedený testovací program pro kontrolu, zda jsme změnu provedli správně. Například pro obrázek z první stránky tohoto článku byly použity parametry:

```
%%BoundingBox: 50 200 560 590
```

Někdy ovšem není editace souboru `obraz.eps` tak snadná, protože soubory pro složitější obrázky dosahují délky několika megabytů. K tomu ovšem nabízí makro `epsf.tex` snadnou výpomoc. Parametry `BoundingBox` lze totiž v souboru `obraz.eps` ignorovat, a použít místo nich parametry explicite napsané v hranatých závorkách za příkazem `\epsfbox`, tedy například:

```
\centerline{\epsfysize=6cm
  \epsfbox[50 200 560 590]{obraz.eps}}
```

Na závěr si něco řekneme o fontech pro popisy obrázků. (Například popisy os apod.)

Pokud se spokojíme s fonty, které používá *Mathematica*, výrazně nám to usnadní práci. Tyto texty jsou v PostScriptovém popisu uloženy s odkazem na PostScriptový font `courier`, případně další fonty. Jestliže GhostScript tyto fonty v instalaci nenajde, nahradí je fontem `ugly`, což je jednoduché, poněkud hranaté písmo. V obou případech se tyto popisy zvětšují a zmenšují společně s obrázkem v libovolném poměru. Tento způsob bývá většinou dostačující a dokonce při tak drobném písmu pro popisy os, jaké vidíte v ukázkách, není příliš patrný rozdíl mezi použitím originálních PostScriptových fontů a písmem `ugly`. Zvláště při použití „hrubého" rozlišení koncového zařízení (pouze 300dpi). Nicméně jsem přesvědčen, že Karel Horák se s tímto písmem nespokojí a instaluje pro účely tohoto článku příslušné PostScriptové fonty.

Chceme-li použít fonty z TEXu, musíme si rozmyslet jejich umístění na stránce a pomocí příkazů TEXu (např. v LATEXu je vhodné prostředí `picture` s příkazem `\put`) umístit texty přesně tam, kam potřebujeme. Zde je třeba upozornit na to, že texty popisů je nutno uvést *později*, než volání samotného obrázku příkazem `\epsfbox`. PostScript totiž překrývá dříve umístěné objekty na stránce novými. Výsledek je mnohem lepší, nicméně je to pracné a velikost popisů je pevná, tj. nemění se při dodatečné změně velikosti obrázku.

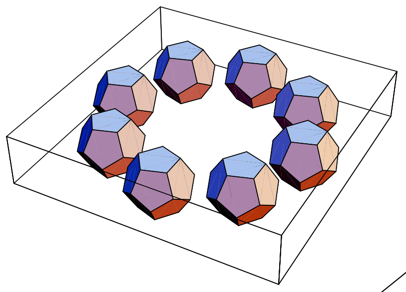
Pokud se nám už poloha obrázku včetně popisů líbí, přistoupíme k vlastnímu tisku. Pro tiskárnu, která není vybavena PostScriptem, stačí napsat například příkaz:

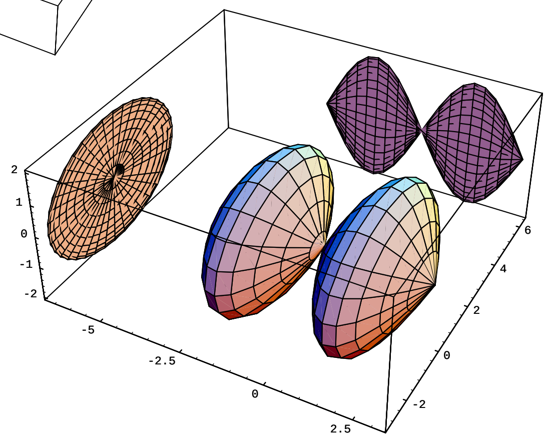
```
gs386 -sDEVICE=laserjet -r300 soubor.ps
```

Poznamenejme, že použití dávky `gslj.bat`, která je součástí instalace GhostScriptu nám na naší tiskárně nefungovala. Tiskárna začala vypisovat zmatené symboly stránku za stránkou v textovém režimu.

Při úspěšném tisku zmizí barva v obrázku a vytisknou se rastry s různými odstíny šedi. Máme-li ale barevnou laserovou tiskárnu, bude asi vybavena PostScriptem a pro závěrečný tisk nám stačí „poslat" soubor `.ps` na tiskárnu, přičemž obrázky zůstanou barevné.

Protože příprava bulletinu neprobíhá na barevných tiskárnách (a také následující reprografické zařízení nepracuje s barvou), spokojíme se s černobílými ukázkami obrázků programu *Mathematica*.






3.1.1993 *Petr Olšák*