The preparation of TEX Version 3 and MF Version 2 has taken me much longer than expected, but at last I've been able to look closely at the concept of virtual fonts. The need for such fonts is indeed much greater now than it was before, because TEX's new multilingual capabilities are significantly more powerful only when suitable fonts are available.

Donald Ervin Knuth: *Virtual Fonts: More Fun for Grand Wizards.* TUGboat, Volume 11 (1990), No. 1

Jak TEX pracuje s PostScriptem

PostScript je dnes jeden z nejpoužívanějších jazyků pro popis stránek a je jedním z mála alternativ, jak získat sazbu na profesionální úrovni. Většina kvalitních tiskáren a osvitových jednotek pracuje pouze v tomto formátu. V tomto článku probereme vazby a softwarové možnosti mezi tímto jazykem a TEXem.

Zmíníme se též o jedné velmi silné možnosti práce s fonty v TEXu – o tak zvaných virtuálních fontech, které autor TEXu uvedl svým známým článkem v TUGboatu z roku 1990 (viz citát v záhlaví článku).

PostScript

Protože použití TEXu s výstupem do PostScriptu sčítá možnosti obou softwarových produktů, které se vzájemně nevylučují, bude užitečné se nejdříve věnovat podrobněji vlastnostem PostScriptu. Odborníci mi snad prominou mé laické vysvětlování.

Snad každé výstupní tiskové zařízení z počítače s výjimkou plotterů je založeno na myšlence obarvit nějak výstupní plochu obdélníkového tvaru (většinou stránku papíru), přičemž toto zařízení si danou plochu rozdělí na síť bodů (raster) a redukuje svůj úkol na vyplňování některých zvolených bodů barvou. Toto vyplňování lze na některých zařízeních přímo řídit počítačem. Například když posíláme na jehličkovou tiskárnu grafický obraz stránky, pak software v našem počítači sám řídí, který bod se má vyčernit a který ne. Pokud ale posíláme na tutéž tiskárnu soubor v textovém režimu, tiskárna přečte jednu řádku textu a rozhodne sama, jaké body vybarví.

Na podobném principu, ale podstatně vyšší úrovni, jsou založena tisková zařízení vybavená PostScriptem. V počítači připravíme textový soubor – tak zvaný PostScriptový popis stránek, nebo též PostScriptový program. Takový soubor pošleme tiskovému zařízení ke zpracování. Zařízení je vybaveno interpretrem, který čte náš textový soubor a na základě přečtených informací zpracovává pomyslný obraz stránky (image) v dané hustotě bodů (raster) a v okamžiku, kdy narazí v PostScriptovém programu na příkaz konce stránky, většinou spustí vlastní tisk stránky podle vytvořeného obrazu. Tento interpretr je součástí softwarového vybavení výstupního zařízení, takže výpočet obrazu stránky probíhá až na místě, kde se stránka tiskne a nikoli v našem počítači. Příslušnému interpretru se říká RIP (Raster Image Processor).

Jazyk PostScriptu definovala firma Adobe. Popis jazyka najdeme v knize

PostScript Language Reference Manual, "The Red Book", Adobe Systems Incorporated, 2nd ed., Addison Wesley 1990.

Tato kniha stojí zhruba $28 a její druhé vydání popisuje tzv. PostScript Level 2, který je výrazným rozšířením původního návrhu (především v oblasti práce s barvou). Kromě této referenční knihy existují knihy dalších „barev", např. *Orange Book* je určena pro profesionální použití, zatímco *Blue Book* je spíš úvodem a *The Black Book* specifikuje smluvený formát fontů Type 1. Samozřejmě, mnoho bylo napsáno i v jiných knihách.

Jazyk PostScriptu je (zhruba řečeno) nezávislý na výstupním zařízení. Znamená to, že v našem PostScriptovém popisu stránek se nemusíme starat o technické záležitosti výstupního zařízení (např. rozlišovací schopnost). To je věcí konkrétní implementace PostScriptového RIPu. PostScript tedy vytváří jakési rozhraní mezi hardwarovým prostředím tiskového zařízení a softwarovým prostředím našeho počítače.

Uvedená nezávislost ovšem nemůže být absolutní. PostScriptový program je schopen např. testovat konkrétní rozlišení a na základě toho větvit svůj výpočet. Nebo jiná záludnost: napíšeme-li v PostScriptu třeba `0.5 setgray`, dali jsme tím najevo, že další objekty budou kladeny do obrazu stránky v šedé barvě někde uprostřed mezi černou a bílou, ale naprosto nevíme, co s tímto požadavkem udělá konkrétní RIP. Na nízkém rozlišení dostaneme třeba puntíkované plochy typu „co puntík, to deset mil" a může to naprosto zhatit náš výtvarný záměr, zatímco na vysokém rozlišení to dopadne většinou dobře. Kdo tedy mluví o nezávislosti PostScriptu na zařízení, má na mysli nezávislost z jakéhosi subjektivního optického hlediska (a to jen tehdy, když se jeho PostScriptový program nevětví podle hardwarových parametrů výstupního zařízení). Přitom matematicky nelze snadno definovat, co v této věci znamená nezávislost na zařízení.

Zastavme se u možností, jaké nám programování v PostScriptu nabízí. V této souvislosti se přiznám ke své jisté profesionální deformaci. Když čtu reklamy, většinou nečtu jejich obsah, ale hodnotím typografickou úroveň a v pozadí jakéhokoli grafického triku vidím konkrétní PostScriptový příkaz, kterým to bylo zařízeno. Nicméně návrháři těchto „písmeno-obrázků" většinou nepracují s PostScriptem přímo, ale prostřednictvím nějakého uživatelsky přítulného rozhraní, jakým je třeba CorelDraw!, či podobné programy. Jinými slovy, aby uživatel vytvořil PostScriptový program, vůbec nemusí umět PostScript. A nyní už k možnostem PostScriptu.

- Pracuje se v souřadnicích s absolutním měřítkem (např. centimetry, tiskařské body apod.), které jsou nezávislé na výstupním rastru. Naše měřítko lze navíc lineárně transformovat (rotace, posunutí a zvětšení). PostScript totiž pracuje s tzv. transformační maticí, pomocí níž vše, co my formulujeme v našich pracovních souřadnicích, transformuje do skutečných souřadnic rastru. Tím lze dosáhnout různé rotace a deformace písmen.
- V každém okamžiku se kromě změny podle transformační matice všechny objekty předzpracovávají na „barvu" podle tzv. grafického stavu. Můžeme nastavit například červenou a pak vše, co bude dále kladeno do obrazu stránky bude červené. Jak to je technicky zařízeno, to je záležitostí konkrétního RIPu (např. se provede barevná separace nebo se objekt vytiskne v jistém stupni šedi).
- Jsme schopni formulovat jisté abstraktní tahy podél úseček, částí kružnic nebo Beziérových křivek, tedy objektů, které jsou matematicky popsány několika málo souřadnicemi či reálnými čísly. Je to podobné, jako „path" v METAFONTu. Tento tah

může tvořit hranici abstraktního objektu, který dostane konkrétní podobu až po přepočítání do rastrového obrazu RIPem. Také lze definovat tloušťku čáry a požadovat, aby RIP vytvořil objekt typu „čára dané tloušťky, jejíž středem vede zadaný abstraktní tah“. Na rozdíl od METAFONTu nelze definovat tvar pera. Konce čar jsou buď hranaté nebo zaoblené.

- Lze požádat o vykreslení tvaru, který je definovaný bitmapově. Zde ovšem pozor! Věrný obraz bitmapy dostaneme pouze tehdy, je-li zrovna transformační matice rovna jednotkové matici. Za jiných okolností RIP konvertuje podle transformační matice (a celého grafického stavu) příslušnou bitmapu na jinou bitmapu, což nedopadá většinou dobře. Proto se bitmapové obrázky s nízkým rozlišením skládají z takových legračních čtverečků, které jsou viditelné pouhým okem a nepomůže ani vysoké rozlišení finálního výstupu.
- PostScript je vybaven podobným (ovšem poněkud chudším) makrojazykem, jako TeX. O vlastnostech PostScriptu se v této věci nebudu pro nedostatek místa rozepisovat.
- Jazyk PostScriptu je vybaven mechanismem práce s fonty. Tyto fonty mohou být definovány jakkoli tzv. rutinou fontu, která je popsána PostScriptovými příkazy. Jde o to, že pokud se např. napíše do PostScriptu string `(AHOJ) show`, je PostScript schopen postupně zpracovat písmena A, H atd. rutinou zvoleného fontu, jejíž výsledkem je (obvykle) grafický objekt na stránce. Tyto rutiny většinou pracují podle principu abstraktních tahů (viz výše), ale mohou též existovat fonty, jejichž rutiny pracují způsobem přenesení předem daných bitmap. První případ je častější – jedná se o tzv. „outline“ PostScriptové fonty, ovšem druhý případ nelze přehlédnout, protože to je jedna z možností, jak dostat TeXovské fonty, které nejsou přístupné v PostScriptové podobě, do PostScriptu.
- Absolutní překrývání. RIP si před tiskem drží obsah stránky v bitmapové podobě v paměti a jeho úkolem je zcela překrýt předchozí grafické objekty novými, třebaže mají světlejší barvu. Toho například využívá program *Mathematica* při kresbě svých prostorových obrázků. Program vůbec neřeší problém viditelnosti, ale prostě kreslí PostScriptem „odzadu dopředu“ a tím přední plošky zakryjí ty zadní. Také velmi hezké stínování ob-

rázků je z 90% zásluhou PostScriptu, který míchá barvy podle úhlu dopadu světelného paprsku na plošku sám. Tady vidíme odpověď na otázku, proč program *Mathematica* nepodporuje jiný grafický formát. Obrázky z tohoto programu si mohl čtenář prohlédnout v prvním čísle bulletinu z r. 1993.

Neuvedl jsem zdaleka všechny možnosti, ale je na čase přejít k věci.

TEX a `dvips`

Ačkoli je TEX a METAFONT podstatně starší, než PostScript, existuje velké množství možností, jak z hlediska TEXu pracovat s PostScriptem. Přitom vlastní program TEX nebyl kvůli tomu vůbec měněn. I v tom je možno spatřovat obrovskou životnost programu a genialitu autora. Navíc pro uživatele TEXu je PostScript pouze jednou z dalších možností výstupu z TEXu, ale sazba v TEXu vůbec není na této možnosti životně závislá.

Postupně si naznačíme nejčastější úkoly, které je nutno řešit v TEXu v souvislosti s PostScriptem. Začněme tím nejběžnějším požadavkem. Máme sazbu připravenou v TEXu (za použití klasických fontů používaných v TEXu) a chtěli bychom ji tisknout na zařízení, které se s námi baví pouze v PostScriptu. V takové situaci použijeme ovladač `dvips`, který transformuje výstup z TEXu (`dvi`) do PostScriptového popisu stránek. Jedná se o volně šířený program, vyrobený Tomem Rokickim (Stanford University). Část tohoto balíku (bez virtuálních fontů) je zařazena do CSTEXu. Ovladač čte `dvi` soubor a bitmapy fontů ve tvaru `pk` nebo `fli` a vytváří PostScriptový soubor, případně výstup rovnou posílá na PostScriptovou tiskárnu. V tomto PostScriptu jsou fonty zapsány v bitmapové podobě a fontový mechanismus PostScriptu pracuje metodou přepisování bitmap. Ve výstupním PostScriptovém souboru proto najdete nejprve hlavičky maker, dále hexadecimální popis bitmap použitých fontů a nakonec vlastní text, který je víceméně nečitelný, protože jsou často mezi jednotlivá písmena vkládány kerningové informace. Taky akcentované znaky jsou přepsány do čísla kódu.

Tady určitě tušíte zradu. Programu `dvips` je nutno v parametrech říci, v jakém rozlišení má soubor `dvi` transformovat. Program podle tohoto údaje použije bitmapy fontů daného rozlišení. Výsledný PostScriptový soubor tedy není zcela 100% nezávislý na výstupním zařízení. Nejvýhodnější by totiž bylo, kdyby příslušná transformační matice byla jednotková. Bitmapy fontů by pak nebyly RIPem následně znova přepočítávány a nedocházelo by k dalším chybám typu „plus minus pixel“. Pro dosažení nejvyšší kvality je proto velice důležité vědět, v jakém rozlišení bude pracovat výstupní zařízení.

Program `dvips` tedy předpokládá, že dané rozlišení je definitivní a proto změní nejprve pracovní souřadnice PostScriptu tak, aby jedna jednotka odpovídala jednomu pixelu. Transformační matice pak bude (v případě, že bude rozlišení dodrženo) jednotková. Navíc program `dvips` zaokrouhluje rozměrové údaje z absolutně přesného `dvi` souboru na jednotky pixelů výstupního zařízení. Takže v PostScriptovém výstupu jsou pro rozměry použita pouze celá čísla (násobky pixelů). Desetinná čísla jsou sice v PostScriptu také možná, ale těm program `dvips` nevěří, protože například „Red Book“ praví, že reprezentace těchto čísel je závislá na implementaci. (To je další argument proti zastáncům názoru, že PostScript je nezávislý na zařízení.)

Když jsem se dozvěděl o možnosti sazby mého seriálu „Kapitoly o TEXu“ (Softwarové noviny) na osvitové jednotce, musel jsem nejprve zjistit, v jakém rozlišení mi to budou dělat. Jednalo se o rozlišení 1270dpi. Pak jsem doma METAFONTem připravil soubory `pk` s tímto rozlišením a programem `dvips` jsem *v tomto rozlišení* připravil soubor `ps`, který jsem zaslal redakci jako soubor pro osvit. Paradoxní na věci je, že takto kvalitní sazbu lze připravit na hloupém AT-čku (a teoreticky i na XT-čku), takže není zcela pravidlem, že k přípravě kvalitní sazby potřebujeme mocné stroje. Navíc takto připravené fonty jsou předem v počítači digitalizované METAFONTem a proto se RIP v osvitce už nezdržuje digitalizováním žádných outline fontů a sazba probíhá svižně. Kdo platí za strojový čas osvitové jednotky, pro toho je tento argument velice důležitý. Digitalizace „outline“ fontů RIPem totiž trvá dosti dlouho. Navíc se zákazník nemusí starat, zda požadovaný font v osvitce mají či nikoli.

Ještě jedna zajímavost, která není zanedbatelná. Jsou schopni uživatelé komerčních DTP programů vidět na obrazovce přesně a nefalšovaně to, co bude sázeno v osvitce? Pokud se programy opírají o „outline“ fonty, pak to technicky není možné. Takže WYSIWYG je vlastně lež. Pokud ale pracujeme s TEXem, pak jsme schopni vidět na obrazovce *pixel per pixel* totéž, co se bude svítit. A stačí nám k tomu obyčejné AT-čko. Bitmapy připravené METAFONTem do tohoto rozlišení lze prostě nabídnout ke čtení prohlížeči `dviscr`. Pak například při zvětšení jedna ku jedné a rozlišení 1270dpi jsou na obrazovce VGA vidět zhruba dva řádky obrovského textu, který na šířku zabírá asi šest písmen. S výřezem lze samozřejmě posunovat a pixel na obrazovce se skutečně rovná pixelu v osvitce. Při zmenšení 10×10 pixelů osvitky na jeden pixel obrazovky se už text dá na obrazovce docela dobře číst. Je to zajímavý zážitek (vynikne krása Computer Modern fontů), ale prakticky to asi není příliš použitelné.

Virtuální fonty

Věnujme se nyní trochu jiné úloze. Chceme zařadit do TEXu běžná PostScriptová písma, jako například Times, Helvetica apod. Výstup chceme realizovat v PostScriptu, přičemž chceme použít pravé „outline" fonty. Máme-li kompletní balík programu `dvips`, najdeme tam TEXovské metriky pro nejběžnější písma už připravené. Například `ptmr.tfm` je font Times-Roman (`p`: PostScriptový, `tm`: Times-Roman, `r`: Regular weight). V TEXu nám tedy stačí napsat

```
\font\muj=ptmr at 10pt
\muj Zde je text ve fontu Times-Roman
\bye
```

a TEX začne sázet v tomto fontu. Uvědomme si, že program TEX pracuje pouze s pomyslnými boxy. Jeho předmětem zájmu vůbec není způsob, jakým dostanou nakonec jednotlivé znaky ve fontu svůj tvar. V `dvi` souboru jsou tedy odkazy na font `ptmr`. Nyní použijeme program `dvips`, který nenajde font `ptmr` v souborech typu `pk` ani `fli`, ale najde soubor `ptmr.vf`, který je také součástí balíku programu. Z tohoto souboru si přečte, co má s uvedeným odkazem na font dělat. Zjistí z něj, že se jedná o font `Times-Roman`, provede případné překódování (například ligatury jsou ve fontu `Times-Roman` na poněkud jiné pozici, než s nimi pracuje TEX) a do PostScriptového výstupu vloží příkaz na zavedení klasického PostScriptového (outline) fontu, jehož znaky jsou pak digitalizovány RIPem až v okamžiku tisku.

Na našem jednoduchém příkladě to funguje, ale holá skutečnost je poněkud komplikovanější. Zatím nebylo nikde řečeno, jak takové soubory `tfm` a `vf` vyrobit v případě, že máme k dispozici nějaký méně běžný PostScriptový font, pro který zmíněné soubory v balíku `dvips` nenajdeme. Taky jsem neprozradil, že to nebude fungovat s háčky a čárkami, pouze jsem schválně zvolil takovou ukázku, kde tyto akcenty nejsou.

K tomu, abychom mohli mluvit o možnostech, jak tyto problémy řešit, musím aspoň zhruba nastínit myšlenku virtuálních fontů. Protože se ale jedná o „Další radosti pro velké kouzelníky“ (viz citát v záhlaví kapitoly), a přitom bych chtěl být stručný, bude se mi tato problematika vysvětlovat velice těžko.

Soubor `vf` je tak zvaný „virtuální font“. Dnes již každý ovladač `dvi` souboru umí v případě, že nenajde font v souborech `pk` nebo `fli`, hledat i v souborech `vf`. Tam jsou uloženy informace o tom, jakým způsobem se má každá pozice fontu (neboli každý znak) realizovat. Nejběžnějším případem je vytáhnout znak z jiného `pk` fontu třeba i z jiné pozice. Nebo lze znak realizovat složením více znaků k sobě, přitom každý z těchto elementárních znaků může být „vytažen“ z jiného `pk` souboru, nebo může být realizován jiným způsobem. Takto lze například zařídit sazbu akcentovaných písmen. V neposlední řadě může být znak realizován jako posloupnost příkazů nějakého jazyka, který umí příslušný ovladač zpracovat (podobně, jako v TEXu příkaz `\special`). V případě ovladače `dvips` může tedy být takovým virtuálním znakem posloupnost PostScriptových příkazů.

Tato myšlenka virtuálních fontů umožňuje nejen využít plně jazyka výstupního zařízení (v našem případě tedy PostScriptu), ale umožňuje překódovat font do libovolného kódování srozumitelného pro výstupní zařízení a sestavovat písmena z dílčích elementů, jako třeba akcenty. Uvědomme si, že tento mechanismus se nemusí použít jenom při práci s PostScriptem, ale je obecný.

Základní myšlenka je formulovaná a dále už jsou zapotřebí jen jednoduché konverzní rutiny. Především formát `vf` je binární a tedy lidsky nečitelný. Proto Knuth definoval ve svém článku konverzní rutiny `VFtoVP` a `VPtoVF`, které konvertují soubor `vf` do formátu `VPL` (virtual property list) a zpět. Tento formát je čitelný textový soubor s přesně definovanou syntaxí. Zde se může TEXový kouzelník vyřádit. V souboru `VPL` jsou kromě informací potřebných pro `vf` formát uloženy také informace pro metriky `tfm`. Dají se zde tedy definovat nejen akce, co provést s každým písmenem na úrovni `vf`, ale také lze zadat kerningové páry, tabulky ligatur, velikosti boxů pro jednotlivá písmena a plno dalších věcí. Rutina `VPtoVF` pak nevytváří jen soubor `vf`, ale též `tfm`.

Dále existuje rutina `afm2tfm`, která konvertuje PostScriptové metriky `afm` (Adobe font metric) do TEXových metrik `tfm` a navíc vytváří „virtual property list“ pro možné další změny a úpravy. Tyto úpravy jsou v případě české sazby s akcenty nutné. Po těchto úpravách se teprve rutinou VPtoVF vytvoří definitivní soubory tfm a vf, vhodné pro českou sazbu PostScriptovými fonty.

Rutina ps2pk dokáže číst fonty v PostScriptovém formátu a digitalizovat je do bitmap tvaru pk. Tím pádem lze využít tisíce dnes nabízených PostScriptových fontů v čisté TEXovské sazbě bez použití výstupu do PostScriptu! Tato rutina je též vhodná pro vytvoření bitmap pro prohlížeč, které jsou většinou nutné i v případě, že zamýšlíme finální výstup do PostScriptu. Samozřejmě lze pomocí virtuálních fontů provizorně směrovat prohlížeč na jiné fonty běžně dosažitelné ve formátu pk. Této možnosti lze využít v případě, že drahá PostScriptová písma nemáme v počítači k dispozici, ale plánujeme finální výstup na zařízení, kde k dispozici budou.

Poznamenejme, že s virtuálními fonty dokážou dnes pracovat všechny DVI-ovladače, tedy například i Mattesovy programy dviscr, dvihplj a dvidot. Všechny zmíněné utility jsou samozřejmě volně šířené, tedy zadarmo. Většinu konverzních utilit lze navíc získat ve zdrojové podobě (jazyk C nebo WEB).

Bohužel uživatelé, kteří chtějí sázet v TEXu PostScriptovými fonty třeba v češtině, nemají dodnes jinou možnost, než stát se kouzelníky a formát VPL si sami připravit. CSTUG zatím tento formát ani pro běžné PostScriptové fonty pro národní sazbu nepodporuje a veřejně nešíří. Proto také virtuální fonty nenajdete v CSTEXu. Doufám, že tento stav je pouze dočasný (viz článek o CS-fontech, odstavec „co dělat“). V této oblasti tedy zbývá ještě hodně udělat z hlediska standardizace národní sazby v TEXu.

PostScriptový \special

Prostřednictvím příkazu \special lze přímo ve zdrojovém textu TEXu formulovat některý speciální požadavek pro pozdější zpracování konkrétním ovladačem. Tento požadavek se přepíše do souboru dvi bez interpretace TEXem a teprve ovladač s ním nějak naloží. V případě ovladače dvips lze zařazovat PostScriptové příkazy. Náš zdrojový text pak může být napůl TEX, napůl PostScript. Tím přebíráme možnosti obou systémů zcela do svých rukou.

Například obrázek ve čtvrté kapitole o TEXu v Softwarových novinách jsem nejprve zkoušel udělat METAFONTem. To fungovalo krásně, ovšem jen pro malá rozlišení. Při 1270dpi jsem narazil na limity METAFONTu a také ovladače dvips, který není schopen zpracovat bitmapu větší, než 64 kB. Rozhodl jsem se tedy pro čistý PostScriptový obrázek. Ve zdrojovém textu jsem zařídil vynechání místa pomocí prostředků TEXu a pak jsem napsal `\special{` a dále se to v mém textu hemžilo PostScriptovými slovíčky typu `moveto`, `lineto`, `curveto` a jinými, pomocí nichž jsem tento jednoduchý obrázek vykreslil. Po ukončovací závorce `}` jsem pak dál pokračoval v psaní textu v TEXu. Digitalizace obrázku pak probíhala až uvnitř osvitové jednotky. Pro „ladění" obrázku jsem použil GhostScript s výstupem na obrazovku.

Když si znovu uvědomíme možnosti PostScriptu, zjistíme, že příkaz `\special` a znalost PostScriptu nám otevírá veliké možnosti. Bohužel nevhodným kombinováním těchto možností většinou dokráčíme až k typografickému zmetku (viz níže), ale přesto bude asi čtenáře zajímat, co lze dělat.

Lze psát do kopce nebo z kopce nebo ěvolbacrz i jinak.

Článek „Jak jsem se sázel svisle" z bulletinu č.4, r.1992 od O. Ulrycha není nutné při vhodném použití PostScriptu brát příliš vážně.

Písmo je možné všelijak deformovat, různě naklánět, ale při všech těchto úpravách mějme na paměti, že při použití TEXovských fontů bude RIP přepočítávat bitmapu na bitmapu a dojde ke zkreslení. V ukázkách jsem sice mohl použít „outline" fonty, ale pro větší názornost jsem zůstal u bitmapových CM fontů, aby bylo toto zkreslení více patrné (rozlišení 300 dpi).

Všechny výše uvedené ukázky se opíraly o jedinou vlastnost PostScriptu – lineární konverze jakéhokoli objektu podle transformační matice. Tato matice se dá měnit nejen přímo, ale i skládáním elementárních geometrických operací, které mají v PostScriptu jména `translate`, `rotate` a `scale`. Takže například text otočený o 90 stupňů byl do tohoto článku zanesen takto:

```
... není nutné při vhodném použití \ps{}u brát \hskip2ex
\hbox to0pt{\special{ps: gsave -90 rotate}
   příliš vážně.\hss\special{ps: grestore}}
```

Všimněte si, že je zde nutné pracovat mírně schizofrenicky. Aktuální bod sazby se nám totiž „rozbíhá“. Něco jiného si o něm myslí TEX a něco jiného vykonává PostScript. Z PostScriptu toho moc na pochopení příkladu nepotřebujete. Příkaz `gsave` zkopíruje do zásobníku grafický stav včetně transformační matice a pracuje s kopií, `-90 rotate` mění transformační matici příslušným způsobem a `grestore` snižuje zásobník a tudíž se vrací k původnímu grafickému stavu. Protože grafický stav obsahuje i informaci o aktuálním bodu sazby, naše ztracené body se nám znova „sejdou“ na stejném místě, kde se rozešly. Z příkladu je rovněž patrné, co dostaneme (nebo spíš nedostaneme) na výstupu v obyčejném prohlížeči bez použití PostScriptu, tj. při ignorování příkazů `\special`. Proto je nutné na konečné doladění textu použít například GhostScript.

Možná vás ještě překvapí syntaxe PostScriptu: píšeme `-90 rotate` a nikoli `rotate(-90)`. Zde panuje totiž (podobně, jako v jazyce pro navrhování stylů v BibTEXu) zásobníkový způsob uvažování. Parametr `-90` se nejprve vloží do zásobníku a poté jej funkce `rotate` ze zásobníku sejme.

Další možností je práce s barvou (`setcolor`) a s rastrovanými stupni šedi (`setgray`) s využitím vlastnosti absolutního překrývání nakreslených objektů novými. V manuálu k programu `dvips` najdete příklad podobný tomuto:

TEX a PostScript

Tento nápis byl do článku zařazen pomocí následujících příkazů:

```
\font\veliky=csbx10 scaled\magstep5
\setbox0=\hbox{\veliky\TeX\ a PostScript}\dp0=0pt \ht0=0pt
{\offinterlineskip
\special{ps: 0.25 setgray}\hrule height2cm   % << šedý pás
\special{ps: 0 setgray}\vglue-20pt
\centerline{\hskip4pt \copy0}                % << černý stín
\special{ps: 1 setgray}\vglue-2pt
\centerline{\box0}                           % << bílé písmo
\special{ps: 0 setgray}\vglue22pt}
```

Myslím si, že tato ukázka nepotřebuje dalšího komentáře. Pouze je vhodné zdůraznit, že parametr pro `setgray` v intervalu $\langle 0, 1\rangle$ označuje množství světla a nikoli černé barvy. Proto 0 znamená černá a 1 bílá.

Čtenář zřejmě tuší, jak by vypadala naše ukázka bez použití PostScriptu. Odpověď: dostaneme jen velký černý obdélník

Komu se tyto konstrukce zdají příliš složité, ten může využít již hotových PostScriptových a TeXových maker, která jsou veřejně k dispozici v balíku zvaném `pstricks`. Bohužel, o tomto balíku nemohu podat podrobnější zprávu, protože nejsem sběratelem cizích maker.

PostScriptové obrázky v TeXu

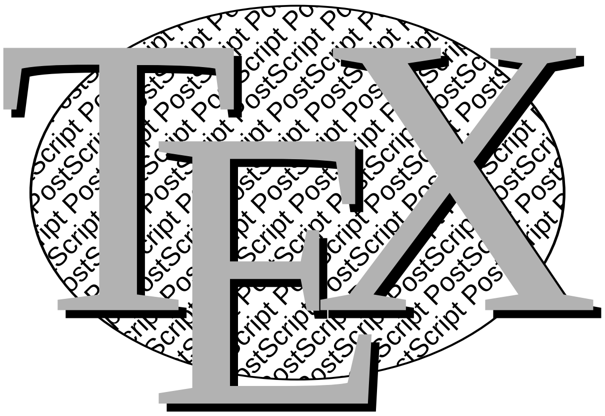
V balíku `dvips` je makro `epsf.tex`, které umožňuje pohodlně vložit PostScriptový obrázek do TeXu. Obrázek by měl být v tzv. „encapsulated PostScript form", což lidově znamená, že splňuje jistou konvenci pro použití jen jistých PostScriptových příkazů. Například v tomto formátu není dovoleno „natvrdo" nastavovat transformační matici, ale pouze je možné relativní nastavení (vzhledem ke stávajícímu stavu matice). Také tento formát nemůže obsahovat například `showpage`, příkaz na „vyjetí" stránky. Jsou-li tyto konvence splněny, pak lze pomocí „obkladových" příkazů PostScriptu zařídit umístění a velikost obrázku na stránce, případně i obrázek lineárně deformovat. „Encapsulated PostScript form" tedy představuje jakousi konvenci bezkonfliktního vkládání PostScriptu do PostScriptu.

V bulletinu č.1 z r. 1993 jsem se zařazováním takových obrázků pomocí makra `epsf.tex` zabýval (v článku „Jak zařadit obrázky z *Mathematicy* do TeXu"), takže tím bych považoval tuto problematiku za probranou.

GhostScript

Nakonec pár slov o volně šířeném simulátoru PostScriptu, který čte PostScriptový soubor a výstupem je buď obrazovka počítače, nebo tiskárna bez PostScriptu (jehličková i laserová). Jedná se o GhostScript, šířený společností GNU. Již jsme se zmínili, že pokud si chcete prohlédnout dokument včetně všech PostScriptových efektů, které nejsou záležitostí TeXu samotného (například barvy, zařazení PostScriptových obrázků, deformace písma apod.), je možné použít GhostScript. Není to úplně plnokrevný PostScript (to by nebyl v „public domain"), ale v podstatě vše, co dokáží PostScriptové RIPy, zvládne taky. Někdy GhostScript zhavaruje na nedostatek paměti počítače nebo HP tiskárny (zvláště, pokud jde o rozsáhlé obrázky). Ovšem buďme rádi, že ho máme.

V balíku je též obsažena knihovna nejběžnějších PostScriptových fontů v souborech s příponou `gsf`, ovšem tyto fonty jsou (z důvodu copyrightu) uloženy pouze v bitmapovém formátu. Budete-li je tedy moc zvětšovat, dočkáte se nepěkných zubatých tahů. Například pro šedý text v závěrečné ukázce tohoto článku bylo použito písmo Times-Roman. Pokud finální tisk článku proběhne pomocí GhostScriptu (a ne skutečného PostScriptového RIPu), pak to poznáme podle toho, že písmo nebude zpracováno jako „outline“, ale bude značně hranaté. Obrázek ilustruje programovací jazyk PostScriptu.



```
\special{" gsave  /Helvetica findfont 10 scalefont setfont
/ps {(PostScript ) show} def /rad {ps ps ps ps ps} def
gsave 30 30 translate 1 .7 scale
newpath 110 110 100 0 360 arc gsave stroke grestore clip
1 1.43 scale 45 rotate -70 70 moveto
{10 -10 rmoveto gsave rad grestore currentpoint pop 100 ge{exit}if}
loop grestore
/Times-Roman findfont 150 scalefont setfont .7 setgray
/tex {(T) show -35 -35 rmoveto (E) show -30 35 rmoveto (X) show} def
30 60 moveto 0 setgray tex  27 63 moveto .7 setgray tex grestore }
```

18.9.1993 *Petr Olšák*