Zpravodaj Československého sdružení uživatelů TeXu

Martin Budaj

Použitie METAPOSTu ako knižnice

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 22 (2012), No. 1, 2–8

Persistent URL: http://dml.cz/dmlcz/150195

Použitie METAPOSTu ako knižnice

Martin Budaj

Článok popisuje použitie METAPOSTu ako knižnice (v tomto kontexte nazývanej MPlib) na tvorbu vektorovej grafiky pre programy písané v jazykoch C, C++ a Lua. V článku sú predstavené príklady, akým spôsobom je možné rozšíriť funkčnosť METAPOSTu (MPlib), ak je grafika vytvorená pomocou MPlib dodatočne spracovaná v aplikácii, ktorá MPlib používa (napr. opakované vzory, priehľadnosť, vrstvy, unikódové texty pomocou OpenType písiem).

Kľúčové slová: MPlib, METAPOST, C/C++, *post-processing* grafiky, knižnica.

Úvod

METAPOST je zvyčajne používaný ako samostatný program, ktorého výstupom sú grafické súbory vo formáte EPS alebo SVG. Od verzie 1.060, zverejnenej v roku 2008, je však možné použiť METAPOST ako knižnicu na spracovanie vektorovej grafiky pre programy napísané v jazykoch C, C++ alebo Lua. Na odlíšenie od názvu METAPOST je knižnica samotná nazvaná MPlib. Vývoj knižnice zabezpečuje Taco Hoekwater, projekt bol finančne podporený aj zo strany CSTUG.

Knižnica MPlib je v súčasnosti používaná samozrejme v METAPOSTe. Ďalším programom, ktorý používa MPlib je LuaTEX. V nasledovnom texte poukážeme na možnosti použitia MPlib vo vlastných programoch.

Základná dokumentácia k MPlib [4] popisuje rozhranie pre jazyky C a Lua. Dokument je stručný a nepokrýva všetky detaily použitia MPlib, môže byť preto často potrebné nahliadnuť aj do zdrojových súborov METAPOSTu. V poslednej časti článku ukážeme niekoľko tipov na použitie nezdokumentovaných častí MPlib.

V súčasnosti sú k dispozícii tri verzie METAPOSTu, resp. MPlib:

- *1.212:* stabilná verzia;
- *1.504 beta:* nepoužíva mem súbory s formátom (číta priamo definície v jazyku METAPOST, napr. `plain.mp`); statická alokácia pamäte je nahradená dynamickou;
- *1.750 pre-alpha:* ako alternatívu k aritmetike s pevnou desatinnou čiarkou umožňuje použiť aritmetiku s pohyblivou desatinnou čiarkou, čo rádovo zvyšuje rozsah prípustných číselných hodnôt.

Zo skúsenosti odporúčame použitie stabilnej verzie. Pri experimentovaní s použitím rôznych verzií MPlib v programe narazíme na drobné zmeny v API medzi jednotlivými verziami (viď posledná časť článku).

Princíp práce s MPlib je nasledovný: popis grafiky v jazyku METAPOST je knižnici odovzdaný v premennej typu reťazec, MPlib tento kód spracuje a (v prí-

 DOI: 10.5300/2012-1/2

pade, že je kód korektný) naplní štruktúry dostupné v jazyku C (resp. tabuľky v jazyku Lua), reprezentujúce kresbu. MPlib taktiež poskytuje podporné funkcie pre prácu s týmito štruktúrami.

Použitie knižnice

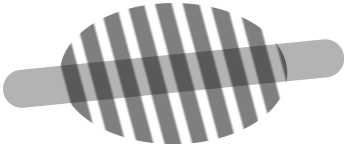
Jednou z výhod použitia MPlib ako grafickej knižnice vo vlastnom programe je nezávislosť na (ne)prítomnosti TEXu a METAPOSTu v systéme, prípadne na ich verziách.

Hlavnou výhodou však je, že v programe získame priamy prístup k interným grafickým štruktúram kresby. Je to dôležité najmä v prípade, ak nechceme použiť priamo grafické výstupy vo formáte EPS alebo SVG, ale chceme grafiku ďalej spracovať a zobraziť ju iným spôsobom. Dodatočné spracovanie grafiky môže slúžiť na doplnenie funkcionality, ktorú METAPOST nepodporuje (napr. opakované vzory, priehľadnosť objektov, gradientné výplne a pod.), resp. na konverziu do iného formátu.

Dodatočné spracovanie grafiky vytvorenej v METAPOSTe je možné aj pri tradičnom prístupe, kedy METAPOST zapíše grafiku do EPS súboru na disku. Ako príklad môžeme uviesť tvorbu máp jaskýň v programe Therion [3]. Mapa je v ňom rozdelená do mnohých fragmentov, z ktorých každý je v METAPOSTe spracovaný ako samostatný obrázok. Therion analyzuje vytvorené EPS súbory, doplní pravidelné vzory a priehľadnosti, konvertuje príkazy PostScriptu do PDF, fragmenty PDF vloží pomocou príkazu `\pdfliteral` do dokumentu spracúvaného pdfTEXom, ktorý vytvorí mapu alebo viacstranový atlas vo formáte PDF. Tento prístup, využívaný v Therione od jeho skromných začiatkov (por. [2]) až do súčasnosti, je však značne obmedzujúci. Použitie MPlib umožní omnoho efektívnejší priamy prístup k jednotlivým grafickým elementom kresby vytvorenej METAPOSTom.

Vlastné rozšírenie jazyka METAPOST môžeme dosiahnuť dvoma spôsobmi:

- použiť príkaz **special** *<text>*; ktorý v MPlib uloží *<text>* do špeciálneho objektu (a ktorý METAPOST zapíše na začiatok EPS súboru pri ukladaní na disk),
- v METAPOSTe 1.000 a novšom špecifikovať **withprescript** *<text>* alebo **withpostscript** *<text>* pri kresliacom príkaze (**fill**, **draw**). Na rozdiel od globálneho **special** *<text>*; je v tomto prípade *<text>* zapísaný tesne pred, resp. za konkrétny grafický element.



Obrázek 1: Elipsa vyplnená opakovaným vzorom s 50% priehľadnosťou. Šedá čiara je pod opakovaným vzorom.

Príklady rozšírenia funkčnosti METAPOSTu

Opakované vzory

Opakované vzory (*tiling patterns*) je možné v METAPOSTe používať vďaka inovatívnej práci P. Boleka [1]. Na označenie objektov, ktoré majú byť vyplnené vzorom, sú v ním navrhnutých makrách použité špeciálne farebné hodnoty (napr. $i \times \epsilon \times white$, kde $i = 1, 2, \ldots$ je poradové číslo vzoru a ϵ je najmenšie kladné číslo v METAPOSTe). Definície samotných vzorov a procedúry mapujúce špeciálne farby na opakované vzory sú zapísané na začiatok EPS súboru pomocou príkazu **special**. Vďaka skutočnosti, že jazyk PostScript umožňuje definovať a volať procedúry, môže interpreter jazyka PostScript pri zobrazení EPS súboru namapovať a zobraziť opakované vzory namiesto špeciálnych farieb, ktoré sú v súbore v skutočnosti použité.

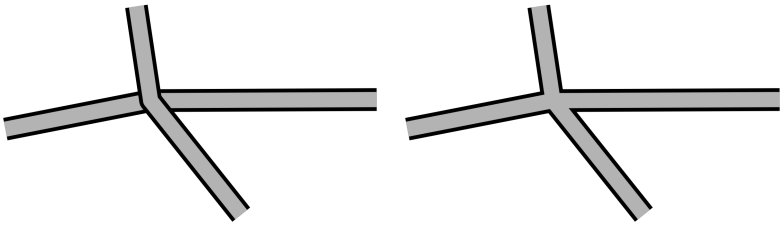
Túto metódu nie je možné analogicky použiť napríklad pri výstupnom formáte SVG, v ktorom nie je možné definovať procedúry zabezpečujúce mapovanie farieb na zodpovedajúce vzory. Pri použití MPlib môže bez problémov zabezpečiť mapovanie špeciálnych farieb na opakované vzory samotná aplikácia, ktorá MPlib využíva, a výsledok uložiť v ľubovoľnom formáte.

Používanie špeciálnych farieb kódujúcich použité vzory je v súčasnosti možné nahradiť napr. špecifikovaním **fill** <...> **withpostscript**(`"pattern:"` & *id*), kde *id* je reťazec s identifikátorom vzoru. Úvodná časť `pattern:` bude v tomto prípade volajúcej aplikácii slúžiť na identifikáciu typu špeciálneho reťazca. Tento prístup zároveň umožní nezávisle nastaviť opakovanému vzoru aj farbu.

Priehľadnosť

Ešte jednoduchším spôsobom je možné definovať priehľadnosť pre jednotlivé objekty, napríklad použitím **fill** <...> **withpostscript**(`"opacity:"` & decimal *o*), kde *o* je číselná hodnota. Opäť je na volajúcej aplikácii, aby v grafickej štruktúre vytvorenej METAPOSTom nahradila výskyty tohto špeciálneho označenia zodpovedajúcim nastavením priehľadnosti pre daný objekt.

Analogicky je možné vyznačiť aj skupiny priehľadnosti (*transparency groups*).



Obrázek 2: Križovatka dvoch ciest. V oboch prípadoch je najprv vykreslená východo-západná cesta, potom severo-juhovýchodná (vždy len jedenkrát). Obrázok vpravo využíva vrstvy na zmenu poradia grafických elementov v porovnaní s poradím v zdrojovom súbore.

Vrstvy

Dodatočné spracovanie môžeme využiť aj na podstatnejšie zásahy do vytvorenej grafiky ako len na jednoduché nahradenia uvedené vyššie. Zaujímavým príkladom môže byť automatické rozdelenie grafiky do vrstiev, ktoré môžu byť usporiadané v inom poradí, ako je poradie objektov vo vstupnom súbore.

Typickým príkladom použitia je mapový symbol pre cestu, tvorený šedým pásom s čiernymi okrajmi. Symbol je možné jednoducho vykresliť v dvoch krokoch: najprv širokým čiernym perom a následne tenším šedým perom po tej istej ceste, čím dosiahneme požadovaný grafický efekt. Problém nastáva pri križovaní dvoch ciest, kedy cesta vykreslená neskôr zanechá neželané artefakty (obr. vľavo). Pri použití preusporiadania sú najprv vykreslené široké čierne čiary pre obidve cesty, až potom tenšie šedé čiary (obr. vpravo).

Definícia symbolu v tomto prípade môže byť:

```
def cesta(expr p) =
    linecap := 0;
    draw p withpen pencircle scaled 3mm withcolor black
            withpostscript("level:1");
    draw p withpen pencircle scaled 2mm withcolor 0.7 * white
            withpostscript("level:2");
enddef;
```

Aplikácia spracúvajúca výstup MPlib musí samozrejme grafiku spracovať v toľkých prechodoch, koľko vrstiev bolo definovaných. V každom prechode vykreslí len tie objekty, ktoré do vrstvy prislúchajú.

Unikódové texty

MPlib nepodporuje zobrazenie textov uzavretých medzi **btex** ... **etex**, na sadzbu ktorých METAPOST (na rozdiel od MPlib) vie použiť (La)TEX. Veľmi dobrý výsledok je však možné dosiahnuť použitím niektorej zo špecializovaných knižníc na sadzbu unikódového textu, napr. Pango [6]. Pango nebude volané priamo z MPlib, ale z aplikácie, ktorá MPlib používa. Negatívom tohto prístupu je, že MPlib nemá k dispozícii informáciu o rozmeroch textu, takže nie je možné napríklad nakresliť okolo neho rámik (rozmery textu budú známe až volajúcej aplikácii, ktorá sa musí postarať o správne zarovnanie a ostatné transformácie textu).

Pri použití knižnice Pango je teda potrebné prostredníctvom operátoru **withpostscript** zapísať pomerne veľké množstvo parametrov: samotný text v kódovaní UTF-8, jeho polohu, zarovnanie, odsadenie, zväčšenie a použitý typ písma, ktoré naša aplikácia vyhodnotí a podľa ktorých text v knižnici Pango vysádza. Keďže v jazyku METAPOST musí byť operátor **withpostscript** viazaný na grafický objekt, môžeme namiesto textu zobraziť napríklad bodku, ktorú volajúca aplikácia síce bude ignorovať, avšak bude sa z nej dať zistiť informácia o prípadných transformáciách a použitej farbe.

Implementácia makra s podobným spôsobom použitia ako makro *label*, známe z Plain METAPOSTu, môže byť napríklad nasledovná:

```
vardef thepangolabel_@#(expr s, z) =
    save p; picture p;
    p = image(draw origin withpen pencircle scaled 1bp
        withpostscript("pangotext:" & s)
        withpostscript("pangooptions:"
            & decimal xpart z & ":"
            & decimal ypart z & ":"
            & decimal labxf_@# & ":"
            & decimal labyf_@# & ":"
            & decimal(labeloffset * xpart laboff_@#) & ":"
            & decimal(labeloffset * ypart laboff_@#) & ":"
            & decimal(10pt * defaultscale) & ":"
            & defaultfont));
    p
enddef;
def pangolabel = draw thepangolabel enddef;

defaultfont := "Linux Libertine O";
defaultscale := 1.2;

pangolabel_urt("Pango <i>text</i>", origin) rotated 30 withcolor red;
```

Značnou výhodou použitia knižnice Pango je jednoduchý a priamy prístup k OpenType fontom inštalovaným v systéme.

Grafický výstup

Aplikácia, ktorá používa MPlib, získa prístup k štruktúram obsahujúcim jednotlivé grafické elementy kresby. Po ich prípadnom dodatočnom spracovaní (ako bolo ukázané v príkladoch uvedených vyššie) je väčšinou potrebné kresbu zobraziť, alebo uložiť do grafického súboru.

Túto úlohu môže uľahčiť špecializovaná knižnica. Ako príklad môžeme uviesť knižnicu Cairo [5], ktorá umožňuje grafiku zobraziť alebo uložiť v rastrovom (PNG) alebo vektorovom (PDF, PS, SVG) formáte. Je zároveň dobre previazaná s knižnicou Pango.

Spojenie MPlib, Pango a Cairo umožňuje spracovanie komplexnej vektorovej grafiky pre programy v C alebo C++. Táto kombinácia bola použitá aj na vykreslenie ilustrácií v tomto článku.

Tipy a triky pri použití knižnice

Posledná časť článku poukazuje na nezdokumentované oblasti MPlib. Dokumentácia je miestami veľmi stručná a popisuje len stabilnú verziu. Pri použití knižnice je často potrebné nahliadnuť do zdrojových súborov `psout.w` alebo `svgout.w`.

Najmarkantnejším rozdielom verzie 1.750 oproti starším verziám je, že číselné hodnoty sú spravidla vyjadrené v PostScriptových bodoch. V starších verziách boli – kvôli použitiu celočíselnej aritmetiky – vyjadrené v *scaled points* (ktorých je 65536 v 1 PostScriptovom bode).

Ďalšou všadeprítomnou zmenou je zmena typu štruktúry, v ktorej MPlib ukladá (grafickú) cestu, z `mp_knot *` na `mp_gr_knot`, ako aj štruktúry samotnej (pre detaily viď súbory `mplib.h` a `mplibps.h`). Konverzia tejto štruktúry do podoby, v akej cestu zapisuje PostScript alebo SVG, nie je úplne triviálna; značnou pomocou môže byť inšpirácia funkciou `mp_gr_ps_path_out` a funkciami z nej volanými v súbore `psout.w`.

Rovnaká štruktúra je použitá aj na uloženie informácií o eliptickom pere, ak je ním cesta vykreslená. Spôsob získania týchto informácií najnázornejšie ilustruje funkcia `mp_svg_pen_info` zo súboru `svgout.w`. Parametre pera je však potrebné dodatočne upraviť (na vykreslenie eliptického pera sa používa lokálna transformácia súradnicového systému a je potrebné zabezpečiť, aby transformačná matica nebola singulárna). Na úpravu parametrov je možné použiť postup podľa funkcie `@<Tweak the transformation...@>` zo súboru `psout.w` (v stabilnej verzii METAPOSTu).

Zoznam literatúry

[1] Bolek, Piotr. *METAPOST and patterns*. TUGboat, 1998, roč. 19, č. 3, s. 276–283; dostupné na `http://www.tug.org/TUGboat/Articles/tb19-3/tb60bolek.pdf`

[2] Budaj, Martin. *METAPOST nielen na nakreslenie loga*. Zpravodaj Československého sdružení uživatelů TEXu, 1999, roč. 9, č. 4, s. 195–201. DOI 10.5300/1999-4/195

[3] Budaj, Martin – Mudrák, Stacho. *Therion – Digital Cave Maps*. Proceedings, 4th European Speleological Congress, Vercors, in: Spelunca, 2008, č. 33, s. 138 – 141, ISBN 978-2-900894-15-6; dostupné na `http://therion.speleo.sk/downloads/tharticle-vercors-2008.pdf`

[4] Hoekwater, Taco. *MPlib API documentation* 2008, 21 s. Dostupné v zdrojovej distribúcii METAPOSTu od verzie 1.090 (júl 2008).

[5] `http://cairographics.org`

[6] `http://pango.org`

Summary: Using METAPOST as a Library

The article informs about using METAPOST as a library (called MPlib in this context) for creating vector graphics in programs written in C, C++ or Lua languages. The article presents a few examples of extensions of the features of METAPOST (MPlib), which is possible if the graphics created by MPlib is post-processed in the application calling MPlib (e.g. tiling patterns, transparency, layers, unicode labels using OpenType fonts).

Key words: MPlib, METAPOST, C/C++, graphics post-processing, library.

Martin Budaj, `m.b@speleo.sk`
c/o CSTUG c/o FEL ČVUT, Technická 2
Praha, CZ-166 27, Czech Republic