TEX METAFONT

3–4

ZPRAVODAJ 98

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

ISSN 1211-6661 Ročník 8

OBSAH

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes `http://www.cstug.cz`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě anonymním ftp na `ftp.icpf.cas.cz` do adresáře `/wagner/incoming/`, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Současně zašlete elektronickou poštou upozornění na `mailto:bulletin@cstug.cz`. Uvedený adresář je pro vás „write/only“. Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

Disketu formátujte nejlépe pro DOS, formát Macintosh 1.44 MB je též přijatelný. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí CSTEXu), zejména v případě, kdy vás nelze kontaktovat e-mailem.

ISSN 1211-6661

Typografie přelomu tisíciletí

Jan Rajlich

Na úvod mi dovolte rozvést malou úvahu o světě kolem nás, o světě vizuálních komunikací, ***světě na přelomu tisíciletí,*** jak jsem, možná trochu v duchu současného světa reklamy a nadsázky, zdůraznil v názvu přednášky.

Žijeme totiž ve světě informací, informací zejména vizuálních, informací, které na nás útočí od rána do večera, a to doslova na každém kroku. Jaké informaci ze stovek nabízených dáme přednost, kterého z vizuálních podnětů si vůbec povšimneme, závisí na tom, jak informace vypadá, jak je schopna zaujmout naši pozornost, jak rychle a jak srozumitelně ji jsme schopni přijmout... Konkurence informací, přesycenost jimi, je tak veliká, že nakonec může vést ke ztrátě komunikativnosti, k popření cíle sdělované informace – kterým jednoznačně je přenos určité výpovědi zaměřené na příjemce.

Dovolte mi, abych na začátku pojmenoval dva negativní jevy světa vizuálních komunikací současnosti. Vizuální chaos (visual chaos) a vizuální znečištění (visual pollution). Oba nejsou úplnou novinkou konce tisíciletí, ale postupně svou intenzitu stupňují. Jde o znečištění, které komplikuje lidský život stejně jako kterékoliv jiné. Jde o část globálního znečištění životního prostředí, které doléhá zejména na člověka, ale má vliv i na ostatní články přírodního prostředí. Vizuální znečištění nemá přímý vliv na tělesné zdraví člověka, ale jen přenesený, prostřednictvím jeho psychiky. Dalo by se tedy říci, že snad není životu nebezpečné, protože nehrozí smrt přímá jako např. při atomové katastrofě.

Vizuální chaos je způsoben překrýváním jednotlivých informací a jejich vzájemným popíráním. Správně aplikovaný grafický design má však sílu z tohoto nepřehledného chaosu vytvořit pole pro organizované vnímání vizuálních informací. Organizované – tudíž rychlejší, přesnější, bezchybnější.

Vizuální znečištění se týká zejména formy vizuální komunikace, má negativní estetický i citový faktor. Pomíjí vžité estetické normy a pravidla, nezná základní principy tvorby a užití prostředků vizuální komunikace, vypadá jednou jako kýč, podruhé jako škrabopis, ale někdy se prohlašuje i za vrchol umění. Většinou jde o díla vytvořená neuměle a neuvědoměle, která kazí, jak se říká, dobrý vkus. Jak se ovšem na ně bude dívat budoucnost za 50, za 100 let, je otázka. Málo si totiž uvědomujeme, že díla čím jsou ze starší doby, tím již takto příkře odsuzována nejsou. (Např. secese také do začátku 60. let 20. století nebyla uznávána, písma 19. století ještě František Muzika ve svém Krásném písmu II. z 50. let označuje za úpadková.)

U diletantsky vytvořených vizuálních informací, a to jak tradičně ručně psaných, kreslených či malovaných, tak zpracovaných pomocí počítače, je myslím proto závažnějším negativem to, že sdělovaná informace je předávána složitě, komplikovaně nebo vůbec nečitelně, že přenášený význam je zkreslen, zamlžen, a tak nepochopen.

Vizuální chaos a znečištění má ovšem dvě roviny. Jedna rovina vizuálního znečištění je spíše materiální – jde o konkrétní projevy – popsání a zaplavení prostředí – reklamou, graffiti, abych jmenoval dvě v posledních 5 letech v České republice nejvíce frekventované „činnosti" na tomto poli. Reklama i graffiti naoko vystupují v protikladu – reklama konzumní, graffiti antikonzumní – ale jejich praktický vliv, účinek na člověka a prostředí je podobný. Obojí – zaplavení prostředí hloupou reklamou i devastování všeho uspořádaného samozvanými „umělci" graffiti, má ale už velice blízko k ekologii.

V druhé – odborné – „grafické rovině" jde o již řečená diletantská pseudografická díla, která brzdí rozvoj lidské kultury, protože neznají vývoj ani současný stav rozvoje, a v důsledku toho přispívají k celkovému úpadku kultury (a kulturnosti), který je v posledních desetiletích (a to i díky nástupu počítačové všem přístupné grafiky) více než patrný. Takzvaní grafici se tu objevili jakoby přímo z „doby kamenné"... (Je to podobné tomu, jako kdyby automobilový průmysl se pokoušel vyrábět moderní automobil ze dřeva – jako žebřiňák.) Tato rovina se tedy týká spíše vědomí a podvědomí, lidské psychiky.

Schází profesionální přístup odborníka, grafického designéra, podložený studiem a znalostí problematiky. Grafický design – jako umění vizuálních komunikací – může podstatně přispět k informativní i estetické hodnotě informace, k optimálnímu zrychlení a čitelnosti.

Na 100 % však nelze tvrdit, že jen „čitelná" vizuální informace má naději na úspěšnou percepci. Každý člověk si vybírá podle svého tajemného klíče, který není veden vždy jen rozumem, ale často i citem a zkušeností; a tyto předpoklady se u každého liší diametrálně...

Grafická mluva jde horizontálně a může překonat jazykové bariéry, používá-li obecněji srozumitelných postupů, znaků apod. Musíme si také uvědomit, že my si všímáme pouze vizuální komunikace jen z oblasti latinkového písma, zatímco většina lidstva používá jiné prostředky vizuální komunikace – jiná písma. Čínské písmo, arabské, hebrejské a azbuka (která už dnes naší mladší generaci také nic nesděluje) jsou jenom čtyři příklady ze stovek rozdílných písem, které se ve světě užívají. Vždyť třeba jen na indické bankovce je text ve 14 různých jazycích (a písmech).

Vraťme se nyní k vlastnímu tématu

Na přelomu 20. a 21. století zažíváme třetí komunikačně informační revoluci, která je srovnatelná s předešlými velmi důležitými historickými etapami ve způsobu jak uchovat a šířit informace.

1. informační revoluce – vznik písma (od 4.500 př. n. l.)
2. informační revoluce – vynález knihtisku (v Evropě po 1440)
3. informační revoluce – informační technologie (od 50. let 20. stol.)

Současná typografie na jedné straně čerpá z celé mnohatisícileté historie vizuálních komunikací a jednak jí nová média staví před dosud nepoznané možnosti. Při vytváření současných textových informací, ať už tištěných nebo obrazovkových, jsou využívány tisíce nových písmových typů, včetně možností jejich modifikací, tisíce barev, podkladů, kombinací se znaky, obrazy ap. Dobře se orientovat v tomto takřka nekonečně variabilním prostředí svede jen zkušený nebo nadaný typograf – grafik, schopný vyvarovat se lákavých ale povrchních řešení a nezabřednout do samoúčelné hry s nepřebernými možnostmi. Naléhavě si uvědomuji, že 3. informační revoluce může vyvrcholit popřením všeho, co tu kdy bylo vytvořeno.

V současné typografii lze zhruba sledovat několik různých poloh:

Mezinárodní styl
navazuje na konstruktivní východiska 20. a 30. let. Od 40. let se nazýval „švýcarská škola“, od 50. let se označuje jako „mezinárodní styl“.

Charakterizuje ho

- racionální rozvrh (vyznává jednoduchost a uměřenost)
- používání rastrového schématu
- dříve pouze bezpatková (bezserifová – lineární) písma – grotesky
- sazba titulků na prapor a sloupců spíše na prapor ale i do bloku

V dnešní době využívá typografie v mezinárodním stylu i jiných písem než pouze grotesků.

Pravidelný rastr je základem úpravy i většiny novin a časopisů.

Např. anglický odborný časopis Baseline (=Účaří) užívá rastru tak hustého, že na první pohled patrný není, ale přesto přispívá k určité optické organizovanosti celku časopisu.

Klasická typografie
využívá klasických tradičních projevů typografie, jak se vyvíjela od 15. století do 20. let 19. století. Vyznačuje se

- respektováním pravidel kompozice stránek vycházejících z konstrukce zlatého řezu (kde jsou např. zajímavé zejména dnes již nezvyklé okraje),

- typografickou „šedostí“ textů (sloupců, stránek) – texty bez kontrastů zaplněné plochy a mezer, texty bez světel, tzv. řek, aby pozornost oka při čtení nebyla přerušována a odváděna
- významovým nakládáním s písmem (vyznačování ap.)
- sazbou titulků spíše na střed, sloupců do bloku
- užíváním (ozdobných) iniciál na začátky kapitol
- používáním hlavně antikvových písem (dnes používá i písma mladší)

Volná typografie
má optický, spíše expresivní přístup k textové ploše, k celé stránce, vytváří „obrazy“ z textu – má nebývalý rozmach díky počítačové sazbě

- malířský přístup – litografie, písmomalířství
- dekorativní – secese, psychedelická písma (hippies)
- povrchový přístup – nová vlna, „nečitelnost“

Obrazovková typografie (screen design)
– ač jde spíše o technologii, která aplikuje styly tiskové typografie – přináší i zcela nové možnosti, které se zpětně přenášejí do tisku (neboť počítač se stal hlavním médiem tiskové přípravy). Rozdíl je ve formátu – obrazovka – jedna strana a horizontální, tiskovina – vertikální stránka nebo dvoustránka.

Obrazovkovou typografii bych rozdělil na interní a externí:

- interní – může využívat dostupné škály typografických prostředků, které si nese s sebou včetně fontů (týká se multimediálních programů, CD ROMů).
- externí – je doposud omezena možnostmi zejména přijímání nesdílených písmových typů, možnostmi rozlišení (WEBové stránky.)

Poslední styl bych nazval pracovně **Počítačový běs**
Jak už bylo nakonec řečeno v úvodu – jde o různé novodobé diletantské kreace laických typografů, jak jim jsou umožněny rozvojem uvedeného nástroje – počítače, a které patří do škatulky vizuálního znečištění. Přílišná snadnost a lehkost práce na počítači svádí k hypertrofickému používání co největšího množství písem, výplní, modifikací atd. Příklady z tohoto soudku naleznete opravdu na každém kroku, zejména v tzv. malé reklamě na různých vývěskách, v inzertních a interních časopisech, ale dostávají se i do knih, na obálky, i na různé drobné tiskoviny jako vstupenky, oznámení, pozvánky atp. a samozřejmě na stránky internetové a do řady počítačových programů. Lze tak hodnotit i většinu tzv. šablon a vzorů např. ve Windows.

Dnes se stává typografem každý, kdo pracuje s počítačem, a tzv. umělcem i ten, kdo si koupí sprej a vyrazí do ulic. (Počítačník, který pracuje jenom doma

a pro sebe, nikomu druhému neškodí, zatímco sprejer svým vandalismem ničí často hodnoty.)

Některé graficky a typograficky neodborné počítačové práce lze tolerovat, s některými pracemi lze polemizovat, některé je také nutno odmítnout.

Otázkou zůstává, nakolik nám to bude platné, protože lidská paměť opravdu postupně absorbuje vše vyprodukované a vytváří si svět nových preferencí či pravidel. Jde totiž o zvyk a o nic jiného. Bráno do důsledků – pak by ovšem nemělo cenu vůbec se např. problémy aplikace typografických pravidel zabývat a raději se oddat osudu. A ač tedy prvně řečené nemyslím nevážně, přesto mi dovolte v komentářích k promítaným ukázkám některé soudy vyslovovat.

Ukázky

(*promítání ukázek ze zahraničních zejména grafických a typografických časopisů: Baseline, Velká Británie; U&lc (Upper and Lower Case), USA; IDEA Magazine, Japonsko; Design Issues, USA; REKLAMNÁ revue vizuálných komunikácií, Slovensko; Da!, Rusko; KAK, Rusko; Design News, Japonsko; Design Exchange, Čína aj.*)

Uvedu nyní i tři příklady z vlastní praxe na časopisech, jejichž grafickou a typografickou podobu jsem vytvářel.

Časopis CLUB, vydával do loňska SCHENK Volleyball Club Brno

Pro tento časopis jsem navrhl v roce 1994 hlavičku ze speciálně zkonstruované současné podoby statické antikvy a základní styl úpravy prvních dvou čísel. Časopis měl na jedné straně vytvářet dojem „podnikatelské“ exkluzivity a serióznosti, na druhé straně nebyl uzavřen i extrémnějším pohledům na design. Proto jsem pracoval hodně s prázdnou plochou, velkými iniciálami, velmi krátkými texty a kontrastem malých a celostránkových fotografií. Redakční klubové materiály byly zpracovávány v jednotném duchu, ostatní příspěvky pak již víceméně individuálně, i když za použití stejného základního písma (*Weidemann*). I inzerce byla vesměs řešena tak, aby její grafická forma byla v souladu s vizuální koncepcí časopisu. Časopis měl zejména podnikatelům ukázat, že není všechno zlato, co se třpytí, že účelem není zaplnění tiskoviny do posledního volného místečka.

TICHO – současník hudby, vydává Skleněná louka v Brně

Doposud jsem navrhl čtyři čísla tohoto nepravidelně vycházejícího časopisu – spíše almanachu, který je zaměřen na současnou vážnou hudbu nekonformního ražení. Proto i úprava je nekonvenční, jednotlivé materiály jsou koncipovány graficky samostatně, je používáno několika základních písem (*Franklin, Garamond*), řady titulkových písem a zejména obrazového principu v kompozici některých stránek (sazba do různých tvarů, podtisky ap.), a to bez jednotícího zrcadla či rastru. Optické hledisko někdy převažuje nad funkčním. Je využíváno účelově i různých papírů – ofset, křída, kulér. I zde je ale dbáno při kompozici na soulad vždy celé dvoustránky, což by mělo být vždy pravidlem při grafické úpravě tiskovin.

BVV MAGAZIN, vydávají Brněnské veletrhy a výstavy

BVV magazín upravuji od roku 1993. V podniku BVV byl zaváděn jednotný vizuální styl již od poloviny 70. let (spolupracoval jsem tehdy na Design-manuálu BVV, jednom z prvních v Československu). Od té doby byl manuál již několikrát inovován. Jeho zásady platí pro všechny projevy podnikového designu, tiskoviny atd. a tedy i pro podnikový časopis. Poslední inovace vizuálního stylu BVV v roce 1994 byla však navržena, jako ostatně více věcí v té době, neodborníky, a tak bohužel některé zásady praxi nevyhovují. V časopise to znamenalo stanovit míru, do které bylo nutno styl používat. Zpočátku byla vytvořena jednotná trojsloupcová úprava s horní silnou vodící linkou v záhlaví a prázdnou lištou využitelnou na titulky, popisky a doběh obrázků, případně i textů. Používáno bylo podnikové písmo *Helvetica* a pro titulky *Avantgarde Gothic*. Sazba na prapor s dělením slov, rastrová úprava byla velmi racionální a technická, až málo časopisecká. A tak (i díky určité kritice) bylo časem přistupováno k oživení stylu časopisu použitím více různých typů písem, více variant sloupcových stránek (2, 3, 4 sloupce) atd. To však znamenalo odklon od vizuálního stylu podniku, proto rok 1999 má znamenat návrat k strohé rastrové podobě s jednotným podnikovým písmem.

Jedním z již uvedených aspektů dobré vizuální komunikace je nepochybně její čitelnost. Aby přednáška měla také malý praktický výstup, připravil jsem **desatero čitelnosti** (asi ale nebude úplně originální). Co a jak je tedy čitelnější:

1. širší řez písma je čitelnější než úzký (ovšem opět do určité míry)
2. minusky jsou čitelnější než verzálky – text z verzálek je dekorativní, ale hůře čitelný
3. dynamická antikvová písma jsou čitelnější než statická
4. grotesky (lineární písma) jsou hůře čitelné v delších textech, lépe v krátkých textech a za pohybu než písma antikvová

5. volně psaná písma, kaligrafická písma, lomená písma, ozdobné varianty nejsou nejčitelnější a v delších textech a řádcích jsou málo čitelná a působí při čtení únavu zraku
6. pro čitelnost je třeba dodržet určitý kontrast textu a pozadí – platí to zejména u barev, které ač jinak výrazně odlišné v kontrastu světlostním i sytostním mohou být stejné a pak je text nečitelný
7. pro dobrou čitelnost se vyhněme používání textů na komplikovaných pozadí (rastry, struktury, fotografie)
8. tmavé písmo na světlé ploše je čitelnější, mění se to ale světelnými podmínkami (ve zhoršených světelných podmínkách je naopak světlé písmo na tmavém pozadí čitelnější – viz reklama)
9. malé mezery mezi slovy (a rovnoměrné) = plynulejší text, z toho vyplývá, že i způsob sazby na prapor je čitelnější než sazba na blok
10. text, který má kolem sebe dostatek prostoru, je čitelnější než text, který vyplňuje danou plochu od kraje ke kraji

Chyby

Největší chyby a nedostatky vídáme tehdy, když se do práce s písmem pouštějí odborníci jiných profesí.

Několik příkladů:

- strojírenství – popis výkresů
- texty k pracovním návodům
- textové pokyny k obsluze strojů
- konstruktéři používají šablonové písmo, které je sice poměrně čitelné, ale zato velice nehezké
- ručně psané plakáty – tzv. písmomalířská písma vycházející z nejsnadnější praxe psaní, písma štětcem bývají nejen nehezká, ale i málo čitelná (příliš úzká písma)
- a samozřejmě již uváděné počítačové pokusnictví

Nejhorší je, když se neodborník pokouší

- o umělecké písmo
- o módní písmo
- vylepšovat písmo
- zdobit písmo

vznikají písma pseudoumělecká (ADAST) a pseudomódní.

Proto vždy doporučuji používat dobré vzory typografických úprav a ověřená písma, a to zejména z odborné typografické a grafické literatury – knih i časopisů. Bohužel většina počítačové literatury o typografii je psána po typografické

stránce z laického pohledu a občas některé tam uváděné rady nejsou doporučeníhodné.

Kapitolou samou pro sebe jsou počítačové fonty. Málokterý program obsahuje slušnou písmovou rodinu s alespoň 3–4 řezy (a českými akcenty). Většina fontů nabízených jako součást západních software je nepoužitelná.

Závěr – iluze po roce 2000

Na závěr mi dovolte malou úvahu ke konci tisíciletí – anebo spíše k začátku nového?

Člověk od pradávna miluje náhražky, napodobování a iluzi; čím je iluze dokonalejší, tím jsme více nadšeni... O iluze se starali jak mágové, tak umělci. Prezentují se iluze formou iluze: napodobeniny, kopie, makety, duchové na míhajících se obrázcích filmu a obrazovkách videa a počítačů, stíny a světla. Virtuální realita. Myslím, že iluze dnešní, tedy iluze elektronická, se přenese (už tomu tak leckde je) do knihoven a poslucháren a odtud dále do domácností. Neboť přístup k elektronické obrazové, textové i zvukové podobě čehokoliv bude mít každý doma – na svém terminálu. Tento týden proběhly tiskem informace o vývoji elektronické knihy – která je nabita a podrží si obsah asi 20 dní. Knihu, jak ji známe dnes, a dosud trváme na skutečnosti knihy, na materiálu – různý papír různých gramáží, jeho struktura, povrch, vůně atd., si asi bude moci zájemce vyrobit „na jedno použití“ doma. Už dnes máme Internet a brzy tak budeme moci shlédnout naprosto vše – pracují na tom na celém světě milióny počítačových mravenečků. Shlédnout vše, ale jen zprostředkovaně na obrazovce monitoru (i když třeba trojdimenzionální...), eventuálně jako výtisk z domácí tiskárny (i když třeba superkvalitní...).

Jak to bude pohodlné nikam nechodit a sedět či ležet si ve virtuálním čemkoli... Virtuální realita se nám sice snaží zprostředkovat skutečnost, ta nám však prozatím moc skutečná nepřipadá. Mám však obavy, že si nakonec budoucnost bude myslet opak a iluze bude ještě více než dosud cennější než skutečnost, neboť se samozřejmě bude stále snažit být i dokonalejší než skutečnost. Otázkou pak tedy zůstává, bude-li v budoucnu vzdálenějším o autenticitu čehokoliv zájem i z pohledu našich robotických potomků. Neberte tento můj závěr pesimisticky, ale vypadá to opravdu, a to jen pozoruji a konstatuji, že svět lidský je skutečně postupně přeměňován na svět umělý (alespoň jsme my – lidé – a to většina z nás – jak už řečeno – tím, co je umělé, vždy *více* nadšeni).

Účastnil jsem se různých setkání, kde zaznívaly velmi varovné hlasy – ať už to bylo třeba na akcích při ekologickém summitu v Riu de Janeiro před 6 lety (tehdy byla také dopracována známá teze o trvale udržitelném rozvoji), designérské konferenci Design Renaissance v Glasgowě v roce 1993 anebo na konferenci washingtonské Společnosti pro vědu techniku a umění r. 1996 s naléhavými apely

prezidenta České akademie věd Rudolfa Zahradníka, který vidí na pozadí současných vědeckých analýz budoucnost lidstva velmi katastroficky (pokud se ve vývojových trendech něco nezmění). Tyto otázky samozřejmě souvisí s naším tématem, neboť nebude-li člověk, možná bude něco jiného a to se týká také typografie. Nemyslím však, že bychom zase měli odevzdaně sedět a čekat na konec světa. Konec tisíciletí nás ale zanedlouho nemine – a ten s sebou prý přináší podobné úvahy jako tato, kterou si dovoluji zakončit své vystoupení.

Doc. ing. arch. Jan Rajlich
ze stejnojmenné přednášky
přednesené 12. 12. 98
na FI MU v Brně

Krátka (oneskorená) reportáž z EuroTEXu '98

Janka Chlebíková

Na prelome marca a apríla 1998 sa vo francúzskom St. Malo konal jubilejný desiaty ročník EuroTEXu. Konferencia prebiehala súbežne s inými konferenciami v rámci dvoch týždňov venovaných elektronickému publikovaniu a typografii. Prednášky sa konali v kongresovom paláci priamo na brehu Atlantického oceánu. A tak zatiaľ čo uši (a hlava) sa venovali TEXu, oči mohli obdivovať prekrásne miesta známe zo zemepisných encyklopédií ako miesta s najväčším rozdielom prílivu a odlivu...

Najhorúcejšou témou na konferencii bol vzťah elektronického publikovania a TEXu. Príspevky sa dotýkali dvoch najrozšírenejších formátov elektronického publikovania PDF a HTML (XML) (spomenutie Wordu v uvedenom kontexte prijalo publikum s úsmevom, pretože nebolo jasné, o ktorú konkrétnu implementáciu sa jedná...).

Do súčasného trendu multifunkčnosti dokumentu zapadol aj tutoriál Hansa Hagena o PDF formáte a jeho možných aplikáciach. Ďalšie príspevky sa týkali rozšírenia TEXu (pdfTEX z brnenskej dielne) ako prostriedku na vytváranie PDF súborov, resp. generovania PDF súborov z DVI súborov. Časť príspevkov pojednávala o vzťahu TEXu a v súčasnosti najrozšírenejšom formáte pre elektronické dokumenty HTML (jazyk WWW). Odzneli príspevky dotýkajúce sa nových smerov na WWW: XML, MathML, či XSL. Tieto by mali uspokojivým spôsobom konečne vyriešiť problém zobrazovania matematických výrazov na webe (len musíme ešte počkať na vhodné prehliadače). TEX pritom môže

slúžiť ako prostriedok pre dokonalé vytlačenie dokumentu plného „matematických chrobákov“. Do danej problematiky spadá aj účastníkmi odmenená najlepšia prezentácia o Techexploreri od firmy IBM slúžiaca ako prehliadač časti TEXových príkazov s priamym prepojením na web.

Ďalšou aktuálnou témou bola problematika okolo matematických fontov v súvislosti so štandardizáciou UNICODE.

Niekoľko postrehov: medzi účastníkmi konferencie rozhodne neprevládali ľudia z akademickej obce (kde má TEX živnú pôdu), ale predovšetkým ľudia z nakladateľstiev, resp. osoby zaoberajúce sa typografiou profesionálne, čo je dobrou zárukou dlhej životnosti TEXu. EuroTEX bola zo všetkých tamojších konferencií najväčšia (cca 120 ľudí) a najdružnejšia. Bolo pre mňa osobným zážitkom spoznať Barbaru Beeton (pravá ruka D. E. Knutha pre TEX) v jej charakteristickom klobúku a môcť stráviť pár príjemných dní s ľuďmi s rovnakou diagnózou číslo 3.1415... (resp. 2.71...)

A čo na záver: torta s klasickým levíčaťom k 10. výročiu francúzskej TUG Gutenberg bola prekrásna (a výborná). Rovnako výborná bola aj podávaná ruská vodka, ako sme sa svorne zhodli s ďalšími členmi CSTUGu Petrom Sojkom a H. T. Thanhom. A čo my? Neupečieme si niečo spoločne k desiatemu výročiu CSTUGu?

Export Mapleovských zápisníků do TEXu

Roman Plch

Maple jako jeden ze systémů počítačové algebry je stejně jako TEX matematiky často využíván. Nabízí se tedy otázka, jak spolu tyto dva programy spolupracují. Následující materiál se snaží poskytnout základní informaci o tom, jak výsledky práce Maplu zařadit do TEXovského dokumentu. (Při tvorbě ilustračních příkladů byly TEX a Maple provozovány pod operačním systémem Linux. Při použití jiného operačního systému můžeme v Maplu obdržet výstupy mírně modifikované.)

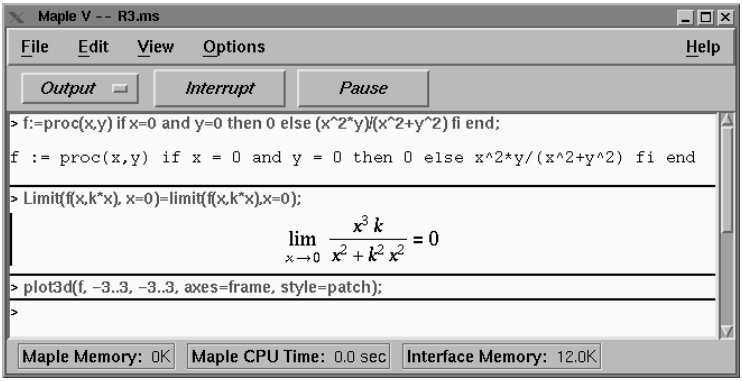
Maple V R3

K začlenění vstupů a výstupů z verze Maple V R3 používáme stylu `mapleenv.sty` (přístupný například na `http://www.hensa.ac.uk/ftp/mirrors/maple/MTN/`). Mapleovský zápisník konvertujeme do LATEXu (2.09) pomocí příkazu

`Export as LaTeX` z menu `File` (na rozdíl od verze R5 zde export funguje bez problémů). K dispozici máme prostředí `mapleinput` pro Mapleovské vstupy, `maplelatex` pro výstupy a `maplettyout` pro tzv. „prettyprint“ a pro chybová hlášení. Obrázek generovaný Maplem uložíme do postscriptu volbou `Postscript` z menu `File` v okně s obrázkem. Získaný postscriptový soubor začleníme do TEXovského dokumentu např. pomocí makra `psfig`. Obrázek je v orientaci „landscape“, proto ho otáčíme pomocí parametru `angle`, parametrem `width` určujeme šířku obrázku, přičemž výška se dopočítá automaticky (můžeme zadat i výšku parametrem `height`, v tomto případě se šířka dopočítá automaticky). Standardně se generuje rámeček kolem obrázku. Pokud je tento nežádoucí, můžeme ho odstranit například editací postscriptového souboru, ve kterém umažeme následující řádky (nacházejí se na konci souboru):

```
%%%% The following draws a box around the plot.
/bd boundarythick 2 idiv def
[] 0 setdash
NP bd bd m bd 5000 bd sub l
6666 bd sub 5000 bd sub l
6666 bd sub bd l
bd bd l S
```

Následující obrázek ukazuje Mapleovský zápisník (R3) použitý jako příklad exportu do LATEXu, dále je uvedena ukázka získaného zdrojového textu:




```
\documentstyle[mapleenv,psfig]{article}
\begin{document}
\begin{mapleinput}
f:=proc(x,y) if x=0 and y=0 then 0 else (x^2*y)/(x^2+y^2) fi
end;
\end{mapleinput}%
\begin{maplettyout}
f := proc(x,y) if x = 0 and y = 0 then 0 else x^2*y/(x^2+y^2) fi
```

```
end
\end{maplettyout}%
\begin{mapleinput}
Limit(f(x,k*x), x=0)=limit(f(x,k*x),x=0);
\end{mapleinput}%
\begin{maplelatex}
\[
{\displaystyle \lim_{{x} \rightarrow 0}} \,{\displaystyle \frac {
{x}^{3}\,{k}}{{x}^{2} + {k}^{2}\,{x}^{2}}}=0
\]
\end{maplelatex}%
\begin{mapleinput}
plot3d(f, -3..3, -3..3, axes=frame, style=patch);
\end{mapleinput}%
\end{document}
```

Na závěr je uveden příklad zařazení postscriptového souboru získaného verzí R3:

```
\begin{center}
\vbox{\psfig{figure=obr3.ps,%
width=11cm,%
angle=270}}}
\end{center}
```

Maple V R5

Začlenění Mapleovských vstupů a výstupů z verze Maple V R5 provádíme pomocí balíku `maple2e` pro LaTeX 2$_\varepsilon$. Balík je distribuován zároveň s Maplem, na Internetu se nachází na `ftp://ftp.maplesoft.com/pub/maple/share/5.5/share/program/maple2e.sty`. Základní prostředí jsou stejná jako u předcházející verze, tedy `mapleinput` pro Mapleovské vstupy, `maplelatex` pro výstupy, `maplettyout` pro tzv. „prettyprint“ a pro chybová hlášení a makro `\mapleplot` pro začlenění postcriptových obrázků generovaných Maplem. Navíc je použito makro `mapleinline` pro zápis matematických výrazů v Maplu.

Příkaz Maplu `latex(expr,"filename")` zapíše odpovídající LaTeXovou konstrukci do souboru `filename`. Pokud soubor s tímto jménem existuje, bude přepsán. Pokud parametr `filename` neuvedeme, výstup příkazu jde na obrazovku a můžeme ho použít ke kopírování do zdrojového kódu LaTeXu, např.:

```
> latex(Limit(int(f(x), x=-n..n),n=infinity));

\lim _{n\rightarrow \infty }\int _{-n}^{n}\!f(x){dx}
```

Celý zápisník můžeme do LaTeXu exportovat volbami `Export`, `LaTeX` z menu `File`. Vygenerovaný soubor `.tex` je připravený pro zpracování LaTeXem. Pokud

převáděný zápisník obsahuje obrázky, Maple vygeneruje odpovídající postscriptové soubory a příkazy LaTeXu potřebné pro jejich začlenění. Bohužel ale export v současné verzi Maplu nefunguje spolehlivě, při řadě pokusů (např. i na příkladu uvedeném v první části) Maple V R5 „havaruje“.

Začlenění ukázek Maplu tedy často provádíme „ručně“. Přitom ukládání obrázku do postscriptu se provádí jiným způsobem než bylo popsáno v části zabývající se verzí R3. Nejprve příkazem:

```
> plotsetup(ps,plotoutput='obr1.eps');
```

Maplu sdělíme, že má grafický výstup ukládat do postscriptového souboru `obr1.eps`. Poté v Maplu generujeme obrázek, jehož výstup se neobjeví na obrazovce, ale uloží do postscriptového souboru `obr1.eps` do aktuálního adresáře. Zamezit kreslení rámečku kolem obrázku můžeme použitím doplňujícího parametru `plotoptions='noborder'`. Výstup grafiky zpět do zápisníku vrátíme příkazem:

```
> plotsetup(default);
```

(Praktické pokusy také ukázaly, že pro generování postcriptových obrázků je výhodnější nastavit výstup do zápisníku, než do samostatného okna.)

Při pokusech se zařazováním obrázků generovaných touto verzí ale zjistíme, že Maple v této verzi chybně nastavuje parametry příkazu `BoundingBox`. Editací souboru `obr1.eps` zjistíme tyto parametry (zde jsou umístěny na konci souboru):

```
%%Trailer
%%BoundingBox: 72 72 719 540
%%EOF
```

K úpravě parametrů `BoundingBox` můžeme použít např. postupu uvedeného v [3] nebo v [1]. Pro následující obrázek (`obr2.eps`) byly použity parametry:

```
%%BoundingBox: 72 72 537 717
```

Upravený obrázek pak začleníme do textu:

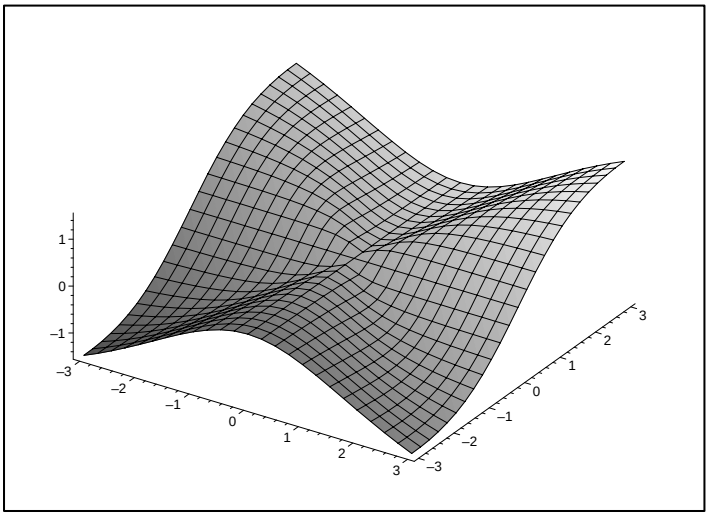
```
\begin{center}
\includegraphics[angle=270,width=9.25cm]{obr2.eps}%
\end{center}
```

Druhou možností je parametry `BoundingBox` v souboru `obr1.eps` ignorovat a použít místo nich parametry explicitně zapsané za příkazem `\includegraphics`, tedy např.:

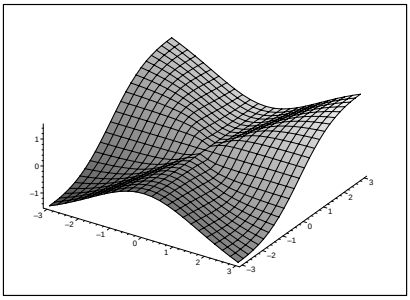
```
\includegraphics[bb=72 72 537 717,angle=270,width=10cm]{obr1.eps}
```

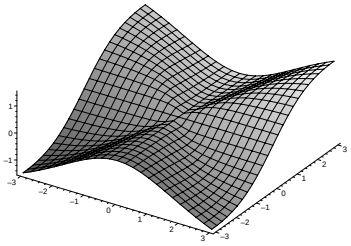
Po úpravě parametrů příkazu `BoundingBox` již můžeme používat i složitějších konstrukcí při zařazování obrázků, např. následující příkazy umístí dva obrázky vedle sebe (u obrázku na pravé straně (`obr3.eps`) byl ostraněn rámeček):

```
\begin{center}
\begin{minipage}[b]{.46\linewidth}
\centering\includegraphics%
[angle=270,width=.95\linewidth]{obr2.eps}%
\end{minipage}
```




```
\begin{minipage}[b]{.46\linewidth}
\centering\includegraphics%
[angle=270,width=.95\linewidth]{obr3.eps}%
\end{minipage}
\end{center}
```





Poslední možností (podle mých zkušeností nejméně vhodnou) je použít přímo makra `\mapleplot` ze stylu `maple2e`. Potom rozměry obrázku ovlivňujeme pomocí `\MaplePlotHeight` a `\MaplePlotWidth` a místo nad a pod obrázkem pomocí `\AboveMaplePlot` a `\BelowMaplePlot`. Při použití makra `\mapleplot` používáme postscriptový soubor generovaný Maplem bez dalších úprav.

Závěrem si ukážeme, jak vypadá zápisník z první části exportovaný (ručně) pro LaTeX 2_ε:

```
\documentclass{article}
```

```
\usepackage{maple2e}
\begin{document}
\begin{mapleinput}
\mapleinline{active}{1d}{f:=proc(x,y) if x=0 and y=0 then 0 else
(x^2*y)/(x^2+y^2) fi end;}{%
}
\end{mapleinput}
\begin{maplettyout}
f := proc(x, y) if x = 0 and y = 0 then 0 else x^2*y/(x^2 + y^2)
fi end
\end{maplettyout}
\begin{mapleinput}
\mapleinline{active}{1d}
{Limit(f(x,k*x), x=0)=limit(f(x,k*x),x=0);}{%
}
\end{mapleinput}
\mapleresult
\begin{maplelatex}
\[
{\displaystyle \lim _{x\rightarrow 0}} \,{\displaystyle \frac {x
^{3}\,k}{x^{2} + k^{2}\,x^{2}}} =0
\]
\end{maplelatex}
\begin{mapleinput}
\mapleinline{active}{1d}{
plot3d(f, -3..3, -3..3, axes=frame, style=patch);}{%
}
\end{mapleinput}
\end{document}
```

Získaný dokument po zpracování TEXem pak vypadá takto:

```
> f:=proc(x,y) if x=0 and y=0 then 0 else (x^2*y)/(x^2+y^2) fi
end;

f := proc(x, y) if x = 0 and y = 0 then 0 else x^2*y/(x^2 + y^2)
fi end

> Limit(f(x,k*x), x=0)=limit(f(x,k*x),x=0);
```

$$\lim_{x \to 0} \frac{x^3\, k}{x^2 + k^2\, x^2} = 0$$

```
> plot3d(f, -3..3, -3..3, axes=frame, style=patch);
```

Odkazy

[1] M. Goosens, S. Rahtz, F. Mittelbach, *The LaTeXGraphics Companion*. Addison Wesley 1997. ISBN 0-201-54199-8.

[2] K.M. Heal, M.L. Hansen, K.M. Rickard, *Maple V Learning Guide*. Springer-Verlag 1998. ISBN 0-387-98399-6.

[3] Petr Olšák, *Jak dostat obrázky z programu Mathematica do TeXu*. Zpravodaj Československého sdružení uživatelů TeXu, **3** (1), 34–40 (1993).

Roman Plch
plch@math.muni.cz

Odborný dokument pre TeX[1] a Web

Janka Chlebíková

Nasledujúci príspevok je počítačovou sondou do odborného (predovšetkým matematického) dokumentu bez ohľadu na hĺbku odborných výsledkov v ňom obsiahnutých...

Nástup osobných počítačov a počítačových sietí priniesol výrazné zmeny v súvislosti s odborným dokumentom. Niekoľko storočí existujúcu papierovú formu dokumentu (resp. mikrofiše) dopĺňa nová elektronická forma dokumentu s rôznymi formátmi pre uchovávanie (Postscript, PDF, HTML a ďalšie) na rôzne typy médií.

Elektronická forma pridáva dokumentom nové rozmery (napr. „živé referencie“, multimediálne prvky, či vyhľadávanie v dokumente), čo v spojení s novými počítačovými technológiami znamená predovšetkým zmenu v *sprístupnení* a *multifunkčnom* využití odborného dokumentu [2].

Zmena zasiahla aj priamo proces vytvárania tlačenej podoby dokumentu. Počítače, tlačiarne, či osvitové jednotky s kvalitným softvérom takmer úplne vytlačili klasické sádzacie stroje.

1. TeX a odborné dokumenty

Počítačová sadzba odborných dokumentov sa takmer od jej vzniku nesie jednoznačne v znamení TeXu. Napriek tomu, že sa neustále vyvíjajú čoraz kvalitnejšie

[1] Pre naše účely TeX ∼ LaTeX ∼ $\mathcal{AMS}$-TeX.

DTP systémy, TEX ako typografický systém pre náročné odborné dokumenty ostáva stále na stupni víťazov.

Navyše TEXový jazyk je najrozšírenejším jazykom v elektronickej komunikácii medzi matematikmi (e-mail, news-groupy, mailing listy), ktorú so sebou priniesol Internet. TEX pre svoju platformovú nezávislosť má svoj leví podiel aj na jednoduchom *sprístupňovaní* odborných dokumentov (napr. výmena elektronických dokumentov).

2. Odborný dokument a súčasný Web

Internet však priniesol svetu i populárny Web spolu s HTML jazykom. V tomto jazyku je viditeľná snaha o vytvorenie dostatočného množstva značiek (tzv. tagov) na vyjadrenie štruktúry univerzálneho typu dokumentu (univerzálny dokument = odborný matematický článok, dopis, slide, ...). Z tagov potrebných pre popis štruktúry samotného matematického výrazu je podporovaný len index a exponent. Akékoľvek zložitejšie matematické formulky predsa typický užívateľ nepoužíva a nieto aby ich ešte zverejňoval na Webe:-)

Stručne povedané, jazyk HTML neposkytuje dostatočné množstvo tagov pre popis štruktúry matematických výrazov. Navyše HTML má fixnú sadu tagov a tak si ich užívateľ nemôže sám pridávať podľa potreby. Na druhej strane, Web je najpopulárnejší prostriedok na *sprístupňovanie* dokumentov. Z tohto dôvodu sa venuje pozornosť rôznym prístupom zobrazovania odborných dokumentov na Webe. Ich spoločnou črtou je:

- dôraz len na primárny cieľ zobrazenia matematického výrazu na Webe
- v podstate všetky predpokladajú vstupné kódovanie v TEXovom jazyku

2.1. Riešenia založené na báze HTML

Uveďme niekoľko najbežnejších prístupov:

1. Najrozšírenejším prístupom je preklad matematických výrazov, ktoré sa nedajú v HTML popísať, do obrázkov (najčastejší formát GIF) a tie zobraziť ako súčasť dokumentu. Takýto prístup ale prináša rad nevýhod:

- je potrebný ľudský faktor na zachytenie kontextu obrázku, nie je možné vyhľadávanie v takýchto obrázkoch, resp. akékoľvek využitie ich obsahu,
- pri zmene veľkosti fontu sa veľkosť obrázku nemení,
- problém s vhodným riadkovaním, odsadzovanie za sebou idúcich riadkov,
- kvalita tlačenia je závislá od kvality obrázkov a nezodpovedá kvalite vytlačeného textu okolo výrazu,
- pomalé načítavanie stránok s množstvom obrázkov.

Príklady implementácií: LaTeX2HTML [`http://www-dsed.llnl.gov/files/programs/unix/latex2html`], TeX4ht [`http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html`].

2. Substitúcia TEXových elementov do symbolov z fontov dostupných bežne na každom počítači (napr. Symbol font). Každý výraz je prekladaný do štruktúry, ktorá môže byť popísaná priamo v HTML jazyku s použitím „bežných“ symbolov. Z dôvodu rýchleho zobrazenia (ale zlej kvality) je takýto prístup vhodný len na získanie prvej informácie o obsahu webovej stránky.

Príklady implementácií: TeX2HTML [`http://www.tex2html.com`].

3. Zobrazovanie matematických výrazov je zabezpečované cez rôzne applety, či plug-iny. Vstupné kódovanie v tomto prípade nie je viazané len na TEXový jazyk, ale rôzne webové nadstavby sú schopné zobraziť aj iné vstupy. Nevýhodou tohto prístupu je závislosť na konkrétnom browseri, či platforme, ktorý je schopný zobraziť vstupné kódovanie. Často je nutnosťou inštalácia podporných programov a potrebný dlhý čas na zobrazenie dokumentu.

Uveďme niektoré implementácie tohto prístupu:

- IBM techexplorer [`http://www.ics.raleigh.ibm.com`]. Veľmi zaujímavé plug-iny pre IE (Internet Explorer) a Netscape Navigator na najrozšírenejšie platformy. Vedia priamo interpretovať podmnožinu TEXových, (LaTEXových) príkazov a cez plug-in zobraziť takéto dokumenty na Webe. V jednom zo svojich projektov realizovali aj prepojenie elektronických dokumentov s computer algebra systémom AXIOM.
- WebEQ Equation Rendering [`http://www.geom.umn.edu/~rminer/jmath`]. Java applet, ktorý zobrazuje matematické výrazy vložené cez embedded elementy (LATEXový popis) do HTML kódu.

2.2. Riešenia bez HTML – Web je iba sprostredkovateľ

V súčasnosti najrozšírenejším spôsobom pre sprístupňovanie odborných dokumentov je použitie niektorého zo štandardných elektronických formátov a Web len ako informačný sprostredkovateľ. Najčastejšie sú používané formáty PS (Postscript), PDF (Portable Document Format), DVI (Device Independent). Takouto formou dnes publikujú na Webe desiatky odborných časopisov [`http://www.emis.de/journals`]. Výhodou tohto prístupu je kontrola autora dokumentu nad konečným vzhľadom dokumentu, čo pri použití HTML jazyka zďaleka nie je možné. Nevýhodou je ale veľkosť prenášaných súborov, nutnosť inštalácie prehliadačov pre každý formát a v niektorých prípadoch problémy s nekompatibilitou formátov.

3. Odborný dokument a budúci Web

Konzorcium w3c [`http://www.w3.org`] zodpovedné za vývoj štandardov na Webe sa po dlhých úvahách rozhodlo uvoľniť ohraničenosť HTML jazyka. Prijatie XML (Extensible Markup Language) vo februári 1998 znamená pre uží-

vateľov predovšetkým možnosť pridávania vlastných tagov, resp. tagov podľa zvoleného DTD (Document Type Definition). XML je totiž „mladším bratom“ SGML a nielen jedným z mnohých DTD, ako tomu bolo v prípade HTML. Jedno výstižné prirovnanie z XML FAQ „XML je skôr SGML-- ako HTML++“.

Prijatím XML sa otvoril nový priestor pre odborné dokumenty na Webe. MathML (Mathematical Markup Language) ako nové DTD v XML poskytuje nový štandard pre kódovanie (značkovanie) matematických výrazov na Webe [`http://www.w3.org/MathML`].

Prirodzená otázka je: na základe akej prezentácie bude browser zobrazovať nové elementy[2], nakoľko XML je jazyk len na popis štruktúry dokumentu. V HTML jazyku je situácia jednoduchšia, lebo fixná množina tagov je dopredu známa. Browser tak môže mať v sebe predpísanú prezentáciu jednotlivých elementov (veľkosť a typ fontov, veľkosť odsadenia, ...). Tento fakt bol ale príliš obmedzujúci a preto bola v HTML jazyku pridaná možnosť kontroly niektorých prezentačných vlastností elementov pomocou CSS (Cascading Style Sheets). Podstatne silnejšie možnosti prezentačnej kontroly poskytuje XSL (Extensible Style Language), ktorý je navrhnutý ako nový štandard pre popis prezentácie XML dokumentov.

Je len otázkou času, kedy dva najrozšírenejšie browsery Netscape a IE začnú „rozumne“ podporovať XML. (Nejaká malá podpora už existuje aj dnes, posledná verzia IE nezobrazuje XML tagy.) Zobrazovanie matematických výrazov na Webe by mohlo byť potom rovnako prirodzené a jednoduché, ako je to v súčasnosti s textom bez nich. Že nerozprávame o príliš vzdialených métach je možné presvedčiť sa na browseri Amaya (experimentálny browser w3c), ktorý podporuje zobrazovanie a jednoduché štrukturované (WYSIWYG) editovanie časti MathML tagov [`http://www.w3.org/Amaya/`].

Vráťme sa však teraz k MathML a pozrime sa bližšie na jeho ciele a možnosti, ktoré poskytuje.

3.1. Ciele MathML

Základným cieľom MathML je, aby matematické výrazy mohli byť podávané, obdržané a spracované na Webe práve tak, ako je to možné pri použití HTML s jednoduchým textom.

Podstatný rozdiel od doterajších prístupov je, že dôraz je kladený nielen na *sprístupnenie* dokumentu, t.j. jeho „pekné“ zobrazenie cez Web, ale i na *multifunkčnom* využití obsahu matematických výrazov (automatické spracovávanie, vyhľadávanie, indexovanie, ..., resp. prepojenie na computer algebra systémy).

Voľba vhodného kódovania dokumentu je dôležitá pre životnosť dokumentov. Vzhľadom k tomuto aspektu MathML je navrhnutý tak, aby umožňoval kódovať

[2]Element je štrukturálna časť dokumentu uzavretá medzi dvoma párovými tagmi, napr. `<LI>...</LI>`.

nielen štruktúru matematického výrazu (syntax), ale i jeho sémantickú časť. Takto je možné realizovať prepojenie s ďalšími aplikáciami.

Dokument označkovaný MathML tagmi je „ľudsky čitateľný“, čo umožňuje jeho jednoduché generovanie z a do iných systémov (napríklad komunikácia s TEXom), automatické spracovanie softvérom, či v krajnom prípade priame editovanie v jednoduchých textových editoroch. (V každom prípade však nie je určený pre „ručné“ editovanie!)

MathML je navrhnutý tak, aby umožňoval pridanie informácie pre špeciálne prehliadače (napríklad zvukový výstup výrazu pre handicapovaných) a rôzne aplikácie. Pokrýva všetky existujúce matematické materiály vhodné pre vedecké a študijné účely.

3.2. Čo vplývalo na MathML

V nemalej miere to bol TEX, ktorý je však prezentačným jazykom a preto nedostatočný pre multifunkčné ciele MathML dokumentov. Z dôvodu premenlivej veľkosti okna a fontu, je potreba i častého zalamovania matematického výrazu. MathML jazyk musí poskytovať dostatočné prostriedky na špecifikáciu miest riadkového zlomu.

Na MathML vplývalo aj ISO 12083 Maths DTD, ktoré však zachytáva tiež viac prezentačné vlastnosti matematických výrazov ako sémantické.

Na sémantickú časť MathML výraznou mierou vplýval Open Math, zastrešujúci computer algebra systémy ako Mathematica, či Maple, ako i samostatné zmienené systémy.

4. Všeobecné princípy MathML

MathML pozostáva z dvoch samostatných skupín tagov: *prezentačných* a *obsahových*. Skupiny sú navzájom nezávislé (t.j. výraz sa popisuje buď v prezentačných, alebo obsahových tagoch), ale za istých pravidiel sa tagy oboch skupín môžu miešať.

4.1. Prezentačné tagy

Cieľom *prezentačných* tagov je popísať štruktúru matematických výrazov tak, aby bolo možné dosiahnuť vysoko kvalitný výstup na výstupných zariadeniach: obrazovke či tlačiarni. Sú v podstate analógiou TEXu s tým rozdielom, že užívateľ je nútený presne implicitne špecifikovať čo je operátor, relácia, premenná, či číslo. Odpovedajú 2-dimenzionálnemu označeniu výrazov typu zlomok, exponent, index, či matica. Atribúty môžu nepriamo ovplyvňovať prezentáciu, napríklad či zátvorky majú vertikálne prekrývať obsahujúci výraz, predpisovať minimálnu veľkosť fontu pri indexoch a podobne.

Príklad použitia prezentačných tagov pre výraz: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.

```
<mrow>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mrow> <mo>-</mo> <mi>b</mi> </mrow>
      <mo>&PlusMinus;</mo>
      <msqrt>
        <mrow>
          <msup> <mi>b</mi> <mn>2</mn> </msup>
          <mo>-</mo>
          <mrow>
            <mn>4</mn>
            <mo>&InvisibleTimes;</mo>
            <mi>a</mi>
            <mo>&InvisibleTimes;</mo>
            <mi>c</mi>
          </mrow>
        </mrow>
      </msqrt>
    </mrow>
    <mrow> <mn>2</mn> <mo>&InvisibleTimes;</mo>
           <mi>a</mi>
    </mrow>
  </mfrac>
</mrow>
```

4.2. Obsahové tagy

Obsahové tagy vyjadrujú sémantiku výrazu pomocou prostriedkov daných v príslušnej matematickej oblasti (teória množín, lineárna algebra, ...). Uveďme jednoduchý príklad pre porovanie s prezentačnými tagmi. Pokiaľ pre prezentáciu výrazu x^2, je postačujúci zápis:

```
<mrow> <msup> <mi> x</mi> <mo> 2</mo></msup></mrow>
```

tak pre jeho sémantiku je nutné vyjadrenie, že horný index znamená umocnenie na príslušný exponent:

```
<expr> <mi> x </mi> <power/> <mn> 2</mn> </expr>
```

Obsahové tagy zahŕňajú širokú škálu prázdnych elementov pre operátory, relácie a pomenované funkcie, ako napríklad `<plus/>`, `<leq/>`, či `<tan/>`.

Umožňujú tiež vyznačiť konštruktor univerzálnej funkcie a jej argumenty, ako i analogické sémantické informácie potrebné na prepojenie napríklad na computer algebra systémy.

```
<apply> <minus/> <ci>a</ci> <ci>b</ci> </apply>
```

Pre porovnanie uveďme opäť zápis $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ v obsahových tagoch:

```
<reln>
  <eq/>
  <ci>x</ci>
  <apply>
    <over/>
    <apply>
      <fn occurrence="infix"><mo>&PlusMinus;</mo></fn>
      <apply> <minus/> <ci>b</ci> </apply>
      <apply>
        <root/>
        <apply>
          <minus/>
          <apply>
            <power/> <ci>b</ci> <cn>2</cn>
          </apply>
          <apply>
            <times/> <cn>4</cn> <ci>a</ci> <ci>c</ci>
          <apply>
        </apply>
        <cn>2</cn>
      </apply>
    </apply>
    <apply>
      <times/> <cn>2</cn> <ci>a</ci>
    </apply>
  </apply>
</reln>
```

4.3. Miešanie obsahových a prezentačných tagov

Rozhodnutie, či použiť obsahové alebo prezentačné tagy, závisí od účelu využitia dokumentu. Ak sú to študijné materiály s perespektívou ich ďalšieho využitia, je lepšie použiť obsahové tagy. Ak nie sú vhodné sémantické prostriedky (napríklad pre $\pm$ v príklade), či pre nové smery v matematike, môžeme použiť len prezentačné tagy

V spomínanom kontexte je vhodné spomenúť ešte jeden veľmi perespektívny element `semantics`. Používa sa na vyjadrenie časti štruktúry matematického výrazu nielen v MathML jazyku, ale i rôznych iných druhoch kódovania (vyjadrené atribútom). Tieto kódovania nemusia mať nič spoločné s XML, ale efektívne môžu byť využité (TEX, jazyk C, Maple) pri iných aplikáciach. Napríklad časť výrazu môže byť vyjadrená v prezentačných a zároveň i v obsahových tagoch, alebo v kódovaní v jazyku Maple, či v TEXu. Takýmto spôsobom sa rozširujú sémantické možnosti obsahových MathML tagov.

Jednoduchý príklad:

```
<semantics>
  <apply><plus/>
    <apply><sin/> <ci> x </ci> </apply>
    <cn> 5 </cn>
  </apply>
  <annotation encoding="TeX">
    \sin x + 5
  </annotation>
</semantics>
```

4.4. Ako je to s matematickými symbolmi?

XML podporuje UNICODE, čím je vytvorený priestor pre podporu veľkého množstva štandardizovaných matematických symbolov. MathML entity obsahujú funkčné pomenovania najpoužívanejších matematických symbolov z UNICODE (napríklad entita `&PlusMinus;` z príkladu).

V rámci projektu Stick je snaha o zjednotenie matematických symbolov a fontov používaných v TEXu, podľa ISO noriem, v UNICODE, computer algebra systémoch, či jednotlivými odbornými vydavateľstvami (napr. Elsevier).

5. Podpora MathML v súčasnosti

Dnes je už MathML podporované viacerými softvérovými produktami. Spomeňme aspoň najdôležitejšie:

- Computer algebra systémy Maple [`http://www.maplesoft.com/`] či Mathematica [`http://www.wolfram.com/news/mathml/`] sú schopné exportovať, importovať a vyhodnotiť MathML výrazy (v obsahových tagoch).
- IBM techexplorer cez plug-in zobraziť v IE alebo Netscape časť MathML tagov.
- WebEQ cez Java applet vie zobraziť v HTML dokumente MathML tagy.

Samostatnú časť je možné venovať vytváraniu MathML dokumentov.

5.1. Ako vytvárať odborné dokumenty pre Web?

TEXové dokumenty je bežné vytvárať „ručne“. TEX má totiž vysoký stupeň inteligencie a rozozná napríklad operátor od relácie. Tým pádom potrebujeme značku (riadiace slovo) vložiť len pri zmene prezentácie, resp. popisovaní dvojrozmernej štruktúry výrazu, teda optimalizovaný počet krát.

V MathML je nutné značkovať všetky štrukturálne elementy už na najnižšej úrovni, napríklad čísla a premenné. Ako vidieť aj z príkladov, v prípade MathML dokumentov by „ručné“ vytváranie bolo krokom späť. Preto je veľká pozornosť venovaná aj problematike vytvárania takto kódovaných dokumentov.

Pohodlným spôsobom možno MathML dokumenty vytvárať v štrukturovaných editoroch [1]. Ich výhodou je, že vytváraný dokument vždy zodpovedá zvolenému DTD. Umožňujú prípadne jednoduchý export do TEXu ako typografickému systému pre účely tlače. Pre experimentovanie so štrukturovaným editovaním možno doporučiť Amayu, či Euromath systém [`http://www.dcs.fmph.uniba.sk/~emt`].

V súčasnosti je rozbehnutých niekoľko nezávislých projektov pre automatické generovanie MathML dokumentov z TEXových a naopak. Nakoľko TEX je prezentačný jazyk, jedná sa o preklad len do prezentačných tagov. Tento prístup je ale veľmi dôležitý, pretože dnes naprostá väčšina matematických dokumentov ak je v elektronickej podobe, tak je v TEXu. Navyše TEX je prirodzeným jazykom matematikov pre ústnu i písomnú podobu a dá sa očakávať, že ho v dohľadnej dobe nič iné nenahradí. Spomínaný prístup je teda výhodný pre priamočiare sprístupnenie TEXových dokumentov na Webe.

6. Literatúra

[1] Janka Chlebíková, *Štrukturované editovanie.* Zpravodaj Československého sdružení uživatelů TEXu, **7** (4), 185–190 (1997).

[2] Philip Taylor, *Computer typesetting or electronic publishing? New trends in scientific publication.* Zpravodaj Československého sdružení uživatelů TEXu, **5** (1–4), 61–89 (1995).

Janka Chlebíková
Katedra vyučovania informatiky
MFF UK, Bratislava
`chlebikj@dcs.fmph.uniba.sk`

Počeštěte si PostScriptový font

Petr Macháček

Abstrakt

Mnoho uživatelů nejen TEXu už někdy pocítilo potřebu mít variantu nějakého PostScriptového fontu obsahující české znaky. *Počeštěním* zde budeme rozumět zprovoznění požadovaného fontu pro práci s českými znaky bez perfekcionistických nároků na typografickou kvalitu dosaženého výsledku. Čím dokonalejší má být výsledek, tím více ruční práce je třeba mu věnovat. Zaměříme se na použití tzv. kompozitních znaků, vytvářejících kresbu českého znaku složením kresby základního písmene a akcentu. Zvídavější či náročnější čtenáře odkážeme na obecný program `t1accent`.

Velice stručné vysvětlení několika pojmů

PostScriptových fontů může být několik druhů. Nejčastěji se jimi rozumí tzv. Type1 fonty, šířené v souborech `.pfa` nebo `.pfb` (Printer Font Ascii/Binary) — ty také většinou máme k dispozici, začneme-li uvažovat o počeštění. Font je spíše než statickým popisem jednotlivých znaků programem ve (zjednodušeném) PostScriptu. Každý znak je pojmenovanou procedurou, jejíž vyvolání (jedním z definovaných způsobů) vytvoří kresbu požadovaného znaku.

Jména těchto procedur jsou nezávislá na konkrétním rozložení znaků v používaném fontu. Jejich vyvolání při vypisování textu je řízeno tzv. Encoding vektorem, polem obsahujícím pro každý z 256 možných kódů znaků název procedury, která by měla znak s tímto kódem vykreslit (nejčastějším jménem je ovšem vyhrazené slovo `.notdef` — příslušný znak nemá definovaný vzhled). Při použití fontu v PostScriptu se dá snadno říci, jaký Encoding vektor má být aplikován.

Většina znaků je pojmenována prostě a intuitivně. Písmeno A se jmenuje `A`, mezera `space`, znaky samostatných akcentů třeba `caron` (háček) či `acute` (čárka). Nás budou zajímat hlavně znaky sestavené ze základního písmene a akcentu; i jejich názvy tuto skutečnost odrážejí — á je `aacute`, Ř je `Rcaron` (ale třeba ť a ď se jmenují `tcaron` a `dcaron`, přestože pro kresbu je většinou použit akcent `quoteright`).

Co potřebujeme

Chceme-li počešťovat font, máme už nejspíš font obsahující základní znaky a potřebujeme doplnit k některým z nich akcenty. Většina fontů, které za počeštění stojí, obsahuje i samostatné akcenty. V nejhorším můžeme sami vytvořit předpis pro vykreslení akcentů. Máme-li hotové základní znaky i akcenty, musíme také vědět, jak je sesadit k sobě. Ukazuje se, že právě toto bývá největší problém — výsledný znak by měl dobře vypadat nejen sám o sobě, ale i v hladkém textu začleněný do slov (o udržení jednotného stylu nemluvě).

Nakonec nám zbyde jen v podstatě triviální úloha — sesadit již existující kresby znaků a akcentů podle nějak získaných pravidel k sobě tak, aby vznikl nový použitelný font. Soubory `.pfa`, `.pfb` vypadají na první pohled nečitelně, neboť jsou nejen kvůli úspoře místa zakódovány. Naštěstí je jejich formát podrobně dokumentován a existují i volně dostupné utility pro převod do čitelné textové podoby a zpět. Úprava čitelného zdrojového textu je snadno proveditelná ať už editorem či automatizovaně. Navíc se někdy můžeme obejít i bez úprav výchozího fontu — což se může hodit, je-li například font dostupný jen v tiskárně nebo nechceme riskovat právní problémy.

Případné nadšení problematiky ne až tak znalých čtenářů nad jednoduchostí celého řešení možná zchladí výhled na obtíže, které přinesou pokusy o konečné začlenění vytvořeného fontu do pracovního prostředí. Ale to by byl námět na značně rozsáhlou a s TEXem ještě méně související studii.

Možné přístupy

Ač by se mohlo zdát, že počeštění PostScriptového fontu je jednoznačně definováno jako tvorba nového Type1 fontu, může být občas schůdnější či dokonce vhodnější dosáhnout požadovaného efektu i jinými prostředky.

Počeštění bez práce s PostScriptem

Je zřejmé, že půjde o řešení problému v případě použití nepočeštěného PostScriptového fontu v nějakém vyšším programu. Nemá valného smyslu zabývat se zde něčím jiným než TEXem. TEX disponuje mocným mechanismem virtuálních fontů, díky němuž lze takřka libovolně sestavit požadovaný font z jiných fontů, ba i dalších objektů. Jedním z programů umožňujících takto vytvořit český font je `accents` (varianta pro ISO-Latin-2 kódování fontů `l2accents`) Jiřího Zlatušky [2], dobře popsaný v článku [1].

Chceme-li si nachystat řešení použitelné ve více vyšších programech, můžeme se přiblížit PostScriptu použitím AFM (Adobe Font Metrics) souboru doplňujícího základní font o dodatečné informace dobře využitelné při sazbě, ale nezpracovávané samotným PostScriptovým RIPem. V AFM souboru jsou kromě

metrik jednotlivých znaků uvedeny například kerningové či ligaturní údaje umožňující sofistikovanějším programům generovat typograficky hodnotnější výstup. Lze v něm též popsat složení kompozitních znaků. Použití takto zadefinovaných znaků v TEXu umožňují už standardní utility pro práci s PostScriptovými fonty (`afm2tfm`). Vytvoření `.afm` souboru s potřebnými složeninami za nás může obstarat program Petra Olšáka `a2ac` [3], jehož součástí je i tradičně obsáhlá a poučná dokumentace.

Využití PostScriptu, nezasahování do Type1 fontu

Už výše byla zmíněna poměrně značná flexibilita PostScriptu v zacházení s fonty. Jsou-li dodržena jistá pravidla, lze vytvořit font použitelný (z hlediska interpretu PostScriptu) stejně jako Type1 fonty, ale využívající plné síly PostScriptu. Procedura příslušná znaku takovéhoto fontu není omezena na použití několika málo vybraných příkazů, ale může kupříkladu vyvolat vykreslení znaků z jiného fontu. Tyto tzv. Type3 fonty ([6, strana 278]) lze chápat obdobně virtuálním fontům TEXu (samozřejmě je ale nutné mít na paměti, že TEX je sázecí systém, kdežto PostScript jazykem pro popis stránek — tedy něco mezi METAFONTem, TEXem a `DVI` formátem).

Vytváření Type3 fontů je rozumným kompromisem mezi uplným obcházením PostScriptu vázaným na konkrétní vyšší program (TEX) a úpravou Type1 fontů, která může být v rozporu s licenčními podmínkami jejich šíření. Použití Type3 fontů je ovšem vázáno na plnohodnotný interpreter PostScriptu (RIP – Raster Image Processor), což postačuje například pro úpravy tiskových výstupů z různých programů (XFig, Netscape,...), ale nelze je použít tam, kde jsou očekávány Type1 fonty (např. v X Window systému).

Pomocí Type3 fontů se požadavkem na tisk nestandardních znaků vyrovnává balík `ogonkify` [10] od Juliusze Chroboczka. Primárním cílem je zajistit tisk PostScriptu obsahujícího znaky s akcenty generovaného například Netscapem (ale i ApplixWarem ci StarOfficem). K tomu ale je ale zapotřebí font obsahující kresby takovýchto znaků. V balíku je pro tento účel program pro vytváření Type3 fontů s kompozitními znaky, jejichž složení ale musí popsat uživatel — přibaleny jsou patrně ručně doladěné metriky pro základní PostScriptové fonty (rodiny Times, Courier, Helvetica), pro vytvoření dalších je možné použít výše zmíněný program `a2ac`.

Skutečné počeštění Type1 fontu

Úprava a vytvoření nového `.pfa/.pfb` souboru obsahujícího definice všech českých znaků a správný Encoding vektor je bez pochyb nejuniverzálnější metodou, jak se s počeštěním vypořádat. Výsledný font lze použít všude tam, kde výchozí, včetně případů, kdy by jiné řešení ani nebylo možné (např. zobrazo-

vání v X Window). Do úvahy však místo technických problémů přichází problémy právní [7, strany 5,6]. Mezi činnosti pokryté copyrightem patří kromě šíření i úpravy programu (jak je Type1 font chápán) a bez souhlasu držitele copyrightu jsou tyto aktivity ilegální. Což nemusí příliš vadit při osobním používání (pro které jsou u některých fontů úpravy povoleny), ale pokud byste je chtěli zahrnout třeba do dokumentu vystaveného na Webu nebo je zaslat s dokumentem k dalšímu zpracování, můžete se dočkat nemilých překvapení. Různé fonty mají ovšem různé licence, takže překvapeni můžete být i příjemně.

K převodu zakódovaného fontu do čitelné podoby slouží program `t1disasm` z balíku `t1utils` [11]. (Jeho dvojče `t1asm` pak zařídí převod upraveného čitelného textu fontu do zakódované podoby.) Nad takto získaným programem lze provádět různé úpravy. Fakt, že se nevytváří nový program, ale upravuje již existující a s dodatečnými úpravami většinou nepočítající zdrojový kód, s sebou nese nebezpečí nepředpokládaného formátu vstupního souboru. Vypořádat se se všemi možnostmi by totiž vyžadovalo implementovat poměrně věrně některé části PostScriptového RIPu.

Pro praktické použití sice nevhodný, ale zmínku si jistě zasluhující program je `mkt1font` z balíku `accfonts` [12] indologa Johna D. Smithe. `mkt1font` je těžko konfigurovatelný (akcenty umisťuje na střed, většina parametrů by se musela upravovat v Perlovém zdrojáku, složení znaku určuje z jeho názvu), poměrně málo odolný vůči změně formátu vstupního souboru (například doplnění fontu od BitStreamu používajícího jiné konvence než Adobe je do značné míry záležitostí ručních oprav) a složené znaky vytváří kopírováním příkazů základního znaku a akcentu do nové procedury. Nicméně tento program měl primárně posloužit svému autorovi k vytvoření fontů vhodných pro sazbu indických jazyků a svůj účel patrně plní.

Po hraní si se skládáním kompozitních znaků, kdy na vykreslení celého znaku stačily v podstatě tři příkazy (základní znak, posun, akcent), může někomu doslovné kopírování všech příkazů pro vykreslení potřebných tahů připadat neefektivní. Bude-li hledat jinou cestu, najde možná operátor `seac` — Standard Encoding Accented Character. Na první pohled se jeví jako to pravé řešení pro vytváření kompozitních znaků v Type1 fontu. Podle definice ovšem vyžaduje, aby jím spojované znaky (základní a akcent) měly stejný kód, jaký jim byl přidělen ve vektoru Adobe Standard Encoding — což bohužel o akcentech, chceme-li font počeštit v souladu s ISO-Latin-2 normou, neplatí. Samotná firma Adobe dnes již používání tohoto operátoru nedoporučuje [9]. Nicméně při bližším prozkoumání kupříkladu Ghostscriptu [13] lze zjistit, že při použití číselných kódů akcentů podle Adobe StandardEncoding, bez ohledu na jejich pozici v právě vytvářeném fontu, operátor `seac` pracuje tak, jak bychom chtěli. Experimentálně bylo ověřeno, že takto vytvořené fonty fungují správně i v X serveru a na HP LaserJet 4 a 5. Pokud ale někdo upravuje font jen pro vlastní potřebu, může pro snazší práci a úsporu paměti zkusit porušit definici Type1 formátu. `;-)`. Operátor `seac`

má poměrně jednoduchou syntaxi vhodnou pro snadné vytváření kompozitních znaků přesně podle metriky vytvořené např. programem `a2ac`. Rozhodne-li se někdo nedbat dobrých rad a použít ho, měl by si dát pozor na správný význam argumentů [8].

Adobe doporučovanou [9] metodou je uložení kresby akcentů (a v podstatě i základních znaků) do pomocných procedur, kde budou uloženy jen jednou, a při tvorbě konkrétních znaků se na ně odvolat. Tohoto postupu se poměrně věrně drží program `t1accent` [4] Petra Olšáka použitý autorem při tvorbě PostScriptové varianty $\mathcal{CS}$-fontů [5]. `t1accent` je velmi obecný program umožňující mnohem víc, než jen prosté sesazení základního znaku a akcentu (příkladem budiž možnost různých tvarů akcentů pro minusky a verzálky). Není patrně primárně určen pro zběžné počeštění fontu na laickému uživateli dostačující úrovni, ale dovoluje (vzhledem k šíři svých možností) relativně snadno aplikovat typografické speciality. Za to ovšem platí poněkud neoperativním ovládáním. Osobní poznámka — jinak velice robustně napsaný program se odmítne zabývat fontem, který nezavedl zkratku pro často používané operátory (`noaccess put`), a pokusy o nápravu v případě Céčkového programu mne osobně dovedly ke `core` souboru, což se mi s Perlovými skripty dosud nepodařilo`;-)`. Přesto se zdá, že chce-li někdo počešťovat kvalitně font na úrovni `.pfa/.pfb` souboru, je `t1accent` nejdokonalejší dostupný nástroj.

Poznámky

Něco málo o kódování

Laskavý čtenář už patrně pochopil, že vlastní kresba znaků je na kódování fontu nezávislá (pomiňme nepodporovaný operátor `seac`). Přesto je třeba věnovat kódování jistou péči. Podprogram daného jména *musí* vykreslovat znak tímto jménem označený. To často není splněno u fontů převedených do Type1 třeba z TrueType pomocí různých okenních přípravků. V původním fontu samozřejmě nebyly jednotlivé znaky pojmenované a při převodu jsou znakům přiřazena nějaká implicitní jména, buď podle Adobe StandardEncoding nebo i hůře. Takový font pak není rozumně použitelný.

Uznávané standardy znakových sad jsou obsahem RFC1345, trochu horší je to s Encoding vektory pro Type1 fonty — kanonický Adobe StandardEncoding je všude, ISOLatin1Encoding též, ale oficiální vektor pro kódování PostScriptových fontů podle ISO-8859-2 jsem nikde nenašel. Téměř každý z výše zmíněných balíků obsahuje nějaký přepis popisu znaků do PostScriptových názvů. Otázka, zda se třeba znak 187, `SMALL LETTER t WITH CARON` má skutečně jmenovat `tcaron` kvůli kompatibilitě s `Tcaron`, nebo `tquoteright`, aby odpovídal název grafickému provedení, se nicméně jeví býti rozhodnuta ve prospěch `tcaron`. V prostředí TEXu lze najít i rozšířené schéma obsahující na pozicích neobsazených v ISO-8859-2 pro sazbu užitečné znaky.

Jak určit sesazení znaků s akcenty

Umístění akcentů ke znakům lze řešit buď individuálně pro každý případ, nebo se lze pokusit o zautomatizování této činnosti. Extrémní přístupy (například vycentrování všech akcentů nad základním znakem) však natropí více škody než užitku. Vhodným nástrojem je dobře konfigurovatelný program `a2ac`, vytvářející AFM soubor s (nejen) popisem sesazení jednotlivých kompozitních znaků. Excesy lze pak snadno opravit ruční editací přehledného AFM souboru. Z informací obsažených v AFM je pak možno vyjít při vlastním sestavení výsledného fontu.

Jaký přístup k řešení zvolit

Nejpoužitelnějšího výsledku lze bezpochyby dosáhnout počeštěním Type1 fontu. Na druhou stranu, pokud vám jde třeba jen o sazbu v TEXu, je zbytečné odvolávat se až do výsledného PostScriptového souboru na svůj nový font a muset ho případně dodávat v něm. Navíc, snažíme-li se používat pro češtinu některé oblíbené leč nepříliš volně šířené fonty, mohli bychom se při příliš hlubokých zásazích dostat do sporu s licenčními podmínkami.

Pokud se rozhodneme řešit počeštění nějakého písma kompozitními znaky sestavenými z již hotových znaků a akcentů, je patrně nejvhodnější vytvořit si jeden soubor popisující složení — nabízí se AFM metrika — a z něj pak vycházet při dalším upravování. Ať už jde o virtuální fonty, Type3 nebo Type1 fonty. To nám umožní kupříkladu mít vytvořený Type1 font pro soukromou potřebu bez ohledu na právní aspekty a využívat ho třeba v grafických nadstavbách (LyX), a vyskytne-li se potřeba šířit vlastní PostScriptový dokument obsahující tento font, jednoduše nahradit Type1 font Type3 fontem stejného vzhledu.

Klademe-li si vysoké typografické nároky, nemůžeme se spokojit s kompozitními znaky (tvar akcentu bude například muset záviset na tvaru základového znaku). V tom případě již není možné vyjít jen z metriky, ale musíme nějak ovlivnit přímo kresbu znaků. To ze zde uvedených utilit podporuje pouze `t1accent`. Nemělo by být ovšem příliš těžké podle definičního souboru případně opět vytvořit Type3 font, nastaly-li by nějaké problémy s používáním Type1 fontů.

Přestože asi neexistuje jednoznačný návod, vyplatí se vždy alespoň tušit, co se při použití toho kterého programu děje a jak věci fungují.

Závěr

Nezbývá než přiznat, že přečtení tohoto TEXtu vás jistě nenaučilo bez dalšího úsilí snadno počešťovat PostScriptové fonty. Snažil jsem se ale poskytnout jistý vhled do problematiky a souhrn dostupných pomůcek včetně svého, ne nutně správného, názoru na ně.

Pokud se vám úspěšně podaří počeštit nějaké písmo, můžete se o svou radost zkusit podělit se správcem stránky Fontanasia [14]. Budete-li si chtít otestovat svůj PostScriptový font, můžete velmi dobře použít Ghostscript (např. soubor `prfont.ps`), pokud chcete přidat PostScriptový font v ISO-8859-2 kódování do Xserveru, bývá často potřeba uvést font v souboru `fonts.dir` pod označením končícím kódováním `-adobe-fontspecific`, a teprve v souboru `fonts.alias` mu vytvořit přezdívku se správným označením kódování `-iso8859-2`.

Elektronická verze tohoto článku, jakož i některé pokusy autora o hrátky s PostScriptovými fonty, lze najít na URL `http://www.fi.muni.cz/~xmachac/slt98/`.

Zdroje informací

[1] Sojka, Petr: *Virtuální fonty, accents a přátelé*, `ftp://ftp.fi.muni.cz/pub/tex/local/cstug/sojka/aboutacc`

[2] Zlatuška, Jiří: `accents, l2accents`, `ftp://ftp.fi.muni.cz/pub/tex/local/fontware/accents`

[3] Olšák, Petr: `a2ac`, `ftp://math.feld.cvut.cz/pub/olsak/a2ac/`, `CTAN:fonts/utilities/a2ac`

[4] Olšák, Petr: `t1accent`, `ftp://math.feld.cvut.cz/pub/olsak/t1accent/`

[5] Olšák, Petr: `t1csfont`, `ftp://math.feld.cvut.cz/pub/olsak/t1csfont/`

[6] Adobe: *PostScript language reference manual (RedBook)*, ISBN 0-201-18127-4

[7] Adobe: *The Adobe Type 1 Font Format (BlackBook)*, ISBN 0-201-57044-0, `http://www.adobe.com/supportservice/devrelations/PDFS/TN/T1_SPEC.PDF`

[8] Adobe: *Adobe Technical Note 5015, Type 1 Font Format Supplement*, `http://www.adobe.com/supportservice/devrelations/PDFS/TN/5015.Type1_Supp.pdf`

[9] Adobe: *Composite Characters*, `http://www.adobe.com/supportservice/devrelations/typeforum/composit.html`

[10] Chroboczek, Juliusz: `ogonkify`, `http://www.dcs.ed.ac.uk/home/jec/programs/ogonkify/`

[11] Hetherington, I. Lee: `t1utils`, `CTAN:fonts/utilities/t1utils`

[12] Smith, John D.: `accfonts`, `ftp://bombay.oriental.cam.ac.uk/pub/john/software/fonts/accfonts/`

[13] `Ghostscript`, `http://www.cs.wisc.edu/~ghost/`

[14] Kvasnička, Michal: *Fontanasia*, `http://www.econ.muni.cz/~qasar/fontanasia/`

Petr Macháček, `xmachac@fi.muni.cz`

Zkušenosti s přípravou CD ROM Equadiff 9

Jaromír Kuben

Poslední týden v srpnu 1997 proběhla v Brně mezinárodní konference o diferenciálních rovnicích *Equadiff 9*, které se zúčastnilo téměř 300 matematiků z celého světa. Tato akce je pořádána od roku 1962 střídavě v Praze, Brně a Bratislavě obvykle po čtyřech letech. Údajně jde v současnosti o nejstarší pravidelnou konferenci v tomto oboru na světě. Byl jsem členem přípravného výboru a očekávalo se ode mne, že se budu věnovat přípravě tiskových materiálů, zejména po konferenci. Obvykle byl vydáván sborník. Tentokrát se mělo navíc objevit zvláštní číslo časopisu *Archivum mathematicum*, vydávaného PřF MU. Pak ale někoho napadlo, že bychom mohli jít s dobou a vydat navíc materiály v elektronické podobě na CD ROMu. Původní představa umístit tam jen sborník a časopis v PostScriptu se postupně měnila a definitivní podoba byla zcela odlišná. Na CD ROMu se nakonec objevily nejen všechny materiály připravené před konferencí, jako abstrakta a program, ale kromě již zmíněného sborníku a časopisu adresář účastníků, 77 fotografií a řada dalších drobných materiálů. Navíc bylo všem účastníkům nabídnuto zaslat jakékoli další příspěvky libovolného rozsahu přednesené na konferenci. Celkově to vyšlo na téměř 1 100 stran textu. Pro případné zájemce o přípravu něčeho obdobného v dalším popíšu, jak byl CD ROM vytvořen.

Formát souborů na CD ROM

Přípravné práce začaly již před vlastní konferencí. Po konzultacích s panem Sojkou bylo nakonec rozhodnuto připravit všechny dokumenty ve formátu PDF (Portable Document Format), který se pro uvedený účel jevil jednoznačně jako nejvhodnější. Jde o rozšířený hypertextový standard, k jehož prohlížení je k dispozici volně přístupný program Adobe Acrobat Reader pro všechny běžné operační systémy (kromě DOSu). Navíc je ho možné prohlížet i pomocí ghostscriptu.

Tento formát je možné získat z PostScriptu pomocí komerčního programu Adobe Acrobat Distiller nebo přímo jako výstup z programu pdftex. Vzhledem k tomu, že tento program jsem neměl v té době nainstalovaný a tudíž jsem s ním neměl žádné zkušenosti, ale měl jsem k dispozici Distiller 2.1, rozhodl jsem se pro první cestu. PostScriptové soubory budou výstupem z programu dvips.

TEXovská část

Jelikož jsem uživatelem LATEXu, byl tím dán základní TEXovský formát. Jako třídu dokumentu jsem se na doporučení pana Sojky rozhodl použít `llncs.cls` (LATEX Lecture Notes for Computer Sciences) ve verzi LATEX 2_ε, který používá nakladatelství Springer pro přípravu sborníků — viz `http://www.springer.de/author/sc-lncs.zip`. Pan Sojka nám současně připravil první verzi stylu `equadiff.sty`. Ten definoval okolí `contribution` pro každý příspěvek, čímž bylo zajištěno, že vlastní definice autorů byly lokální, a zároveň se testovalo, zda autor zadal všechny povinné položky. Tento styl jsem později modifikoval a rozšiřoval. Současně byl vytvořen pro autory vzorový příspěvek, který si mohli účastníci buď zkopírovat během konference nebo stáhnout z Internetu.

Autorům příspěvků do sborníku a časopisu bylo nabídnuto, že nemusí požadovanou formu dodržet, po ostatních jsme to chtěli vyžadovat. Bohužel ze všech příspěvků jen asi dva nepotřebovaly téměř žádnou úpravu a použily doporučené styly. Ostatní se pohybovaly od LATEX 2.09 a LATEX 2_ε (asi 60%), $\mathcal{AMS}$-TEX (asi 40%) až po plain (jeden článek). Nakonec bylo přijato vše a muselo se předělat přes 800 stran, což mi spolu s ostatním zabralo téměř devět měsíců.

Mezi dalšími použitými stylovými balíky byly `amsmath`, `amsfonts`, `amssymb`, `graphicx`, `texdraw`, `xy` a `array`.

Klíčovou roli sehrál balík maker S. Rahtze `hyperref`. Nikdy předtím jsem ho nepoužil, a tak jsem postupně odhaloval, co vše dokáže. Je toho hodně. Mimo jiné předefinuje řadu standardních LATEXovských příkazů a způsobí, že veškeré odkazy na literaturu, sekce, rovnice, věty, definice atd. ale i položky rejstříku a obsah změní na hypertextové linky. Ovšem jen ty, které byly vytvořeny pomocí mechanismu `\label` — `\ref` a `\bibitem` — `\cite`, což bylo bohužel naprostou výjimkou a představovalo další obrovskou mechanickou práci předělat tisíce „tvrdých“ odkazů. Navíc tyto odkazy musely být v rámci daného celku (zvlášť se vytvářel sborník, časopis a ostatní příspěvky) jedinečné. Takže i když autor použil okolí `thebibliography` a čísloval položky 1, 2 atd., bylo nutné změnit návěští (vkládal jsem navíc první tři písmena ze jména autora) často ve stovkách položek a odkazů na ně.

Pracovní prostředí

Aby bylo vůbec možné něco takového uskutečnit, bylo důležité mít vhodný nástroj. Naštěstí používám editor epm v operačním systému OS/2, který leccos dovede, zejména vyhledávání a výměny s využitím rozšířených regulárních výrazů. Jinou variantou by byl asi emacs. Domnívám se, že použití skriptů pro sed, awk nebo perl by bylo neúčinné — každý příspěvek byl trochu jiný a výjimek bylo příliš mnoho. Celou práci jsem, jak už bylo naznačeno, prováděl

v prostředí OS/2 s využitím nadstavby pmCSTeX nad editorem epm, jejímž autorem je P. Mikulík (`http://www.sci.muni.cz/~mikulik/os2.html`). Z editoru se přímo spouští překlad (a nemusí to být jen TEX, ale třeba i METAFONT nebo METAPOST), prohlížení dvi a PostScriptových souborů, dvips, index atd. Rovněž není problém mít otevřených několik desítek souborů, což jsem často potřeboval.

Další požadavky

Chtěl bych podotknout, že když jsem začal pracovat na přípravě CD ROMu, zdaleka mi nebylo dopředu jasné, co vše bude třeba udělat. To se postupně vyvíjelo, jak se objevovaly další požadavky a jak jsem zjišťoval, co vše je možné udělat. Soubory `equadiff.sty` a i `llncs.sty` se průběžně měnily a zdárně rozrůstaly.

Např. se objevil požadavek, aby v hlavičce každého příspěvku bylo uvedeno, od které strany po kterou sahá. Vyřešilo to několik vhodných maker, která jen potvrdila sílu TEXu.

Dalším požadavkem byly rejstříky. Chtěl jsem udělat dva — autorský rejstřík a rejstřík AMS Subject Classsification. Potíž byla, že balík `hyperref` nedělal hypertextové linky z příkazů `\glossary`, ale jen `\index`. Nakonec jsem se rozhodl dělat rejstříky nadvakrát. Při jednom překladu se jistý příkaz předefinoval na `\index` a ze souboru `.idx` se vygeneroval jeden rejstřík. Pak se předefinoval jiný příkaz a ze souboru `.idx` se vygeneroval druhý rejstřík. Nakonec se do souboru `.idx` ukládaly oba rejstříky, aby se správně vytvořily linky (soubor `.idx` už byl v tu chvíli nepotřebný).

Balík `hyperref` s volbou bookmarks vytváří pomocný soubor s příponou `out`, do nějž ukládá informace potřebné pro vytvoření tzv. záložek. Jde o něco podobného jako při tvorbě obsahu. Tyto údaje jsou zpracovány vlastně až při distilaci a musí jít o obyčejný text. Případné TEXovské příkazy v nadpisech způsobí katastrofu. Proto je nutné tento pomocný soubor editovat a příkazy nahradit vhodným textem. Navíc vznikají principiální problémy s diakritikou, které jsou, jak jsem se později dozvěděl od odborníků, neřešitelné. Některé znaky v použitých fontech prostě nejsou a navíc to závisí na operačním systému.

Použité fonty

Aby byla při prohlížení výsledného PDF souboru zajištěna vysoká kvalita i při velkém zvětšení (a i z jiných důvodů), je nutné použít vektorové fonty. Volil jsem Computer Modern PostScript Fonts, které uvolnila AMS. Některé chybějící velikosti jsem doplnil z obdobné sady BaKoMa (PostScriptové csr fonty se objevily až v době, kdy práce už končila). Jinou variantou byla volba fontů Times, ale ty

by se měly pro sazbu matematiky kombinovat s komerčními fonty MathTimes, jejichž použití na CD ROMu je finančně dost náročné.

Grafika

Obrázků sice nebylo v příspěvcích příliš mnoho, přesto se ale objevilo téměř vše z běžného TEXovského repertoáru — bitmapy PCX a BMP, Encapsulated PostScript, METAFONT (ten jsem předělal na METAPOST), LATEXovské okolí `picture` a balíky `texdraw` a `xy`. Kvalita obrázků byla značně různorodá.

Vytvoření PDF souborů

Z výsledných `dvi` souborů byl pomocí programu dvips vytvořen PostScript. Ukázalo se, že je nutné použít volbu `-j0`, která zajišťuje, že fonty budou zařazeny celé, ne jen použité znaky. V opačném případě při prohlížení PostScriptu i PDF chyběly některé znaky. Problémy zřejmě působí obrázky z METAPOSTu, v nichž je nějaký text.

Použití Distilleru je jednoduché. Při vhodné konfiguraci stačí PostScriptový soubor nakopírovat do jednoho adresáře a v jiném se najde výsledný PDF soubor. Takových souborů je 15 (od jednostránkových až po 338 stran) a jsou vzájemně provázány, což umí rovněž zařídit styl `hyperref`. Dále je třeba připočítat 77 fotografií, které byly kvalitně naskenovány, upraveny, uloženy jako Encapsulated PostScript a načteny do jednostránkových TEXovských souborů a po zpracování dvips distilovány. Vše je propojeno přes jeden hlavní soubor, z něhož je řada odskoků do ostatních dokumentů s možností návratu. V případě připojení na Internet některé linky spouštějí prohlížeč WWW s příslušnou adresou.

Pokud máme v úmyslu umístit PDF soubor na Internet, je žádoucí provést optimalizaci, což způsobí, že při prohlížení je vždy stahována jen aktuální stránka, ne celý soubor. To se provádí programem Acrobat Exchange, ale až od verze 3.0. Vůbec s verzí 2.1 se vyskytovaly potíže. Při použití stylu `hyperref` v novějších verzích (poslední je 6.40) nefungují některé linky, při načtení do Acrobat Exchange se ztrácely některé znaky a pod.

Dále je vhodné vytvořit thumbnails — miniaturní ukázky stran, které se zobrazují v Acrobat Readeru. Ve verzi 2.1 se to dělá při distilaci, ve verzi 3.0 pomocí Acrobat Exchange. Ukazuje se tedy, že i při použití pdftexu se (pokud chceme optimalizaci a miniatury) bez použití komerčního balíku Adobe Acrobat neobejdeme.

Závěrečné kroky

Vlastní PDF soubory s textem zabraly asi 15 MB, fotografie pak asi 50 MB. Z toho důvodu jsme se rozhodli na CD ROM umístit tytéž fotografie ještě v komprimovaném formátu TIFF. To představovalo dalších 400 MB.

Konečně z důvodů soběstačnosti jsme chtěli na CD ROM umístit anglické verze programu Acrobat Reader pro všechny dostupné operační systémy. Zde jsme narazili na problém dlouhých jmen u verzí pro unix. Nakonec mi pomohl jeden ze správců serverů na VA, který programem mkisofs pod linuxem vyrobil obraz CD ROMu se souborovým systémem ISO 9660 s Rock Ridge Extensions, což umožňuje dlouhá jména. Obraz se pak vypálil. Přes veškerou snahu se nepodařilo vyrobit CD ROM s oblastí se souborovým systémem ISO 9660 a současně s oblastí se souborovým systémem, který používá Mac OS a z něhož by bylo možné přímo spustit instalační program pro Acrobat Reader na Mac OS. Museli jsme ho tedy umístit na CD ROM zazipovaný.

Pak už zbývalo jen připravit průvodní brožuru (k tomu posloužil výše zmíněny hlavní soubor), inlay (text včetně hřbetů vkládaný na zadní stranu obalu) a potisk CD ROMu (pro psaní do kruhu se mi osvědčil balík `pstricks`). CD ROM vylisovali v Loděnicích v nákladu 600 kusů.

Na semináři SLT v Jevíčku se mne někdo zeptal, zda bych postupoval stejně, kdybych měl dělat něco podobného ještě jednou. Pokud bych se k tomu vůbec odhodlal, tak asi ano. Možná bych jen použil pdftex. Také bych mnohem lépe věděl, jak co dělat. Určitě by se ušetřilo spoustu práce, kdyby se podařilo autory donutit, aby použili připravené styly. Ale to nebývá tak jednoduché. Mohu se utěšovat tím, že získané zkušenosti zužitkuji např. při psaní skript. Pokud se vše předem důkladně rozmyslí a od začátku se důsledně značkuje, nepřináší vytvoření PDF souborů žádnou podstatnou práci navíc.

Kde co najít

Pro případné zájemce umístím na `ftp.fi.muni.cz` v adresáři `/pub/tex/local/cstug/kuben/equadiff` některé ukázky stylů, souborů pro přípravu formátu, článků a pod. Na adrese `http://www.math.muni.cz/Equadiff9CDROM/` bude časem možné prohlédnout si vše podstatné z CD ROM *Equadiff 9* (soubory s textem a fotografiemi v PDF).

Jaromír Kuben
`kuben@scova.vabo.cz`

LaTeX pro začátečníky – komentář nad novým vydáním knihy

Jiří Rybička

Jedním z hlavních cílů činnosti Československého sdružení uživatelů TeXu je podpora kvalitní počítačové typografie. Zásluhou tohoto sdružení je nyní těsně před dokončením již druhé vydání knihy LaTeX pro začátečníky, jejímž záměrem je oslovit zejména ty uživatele počítačů, kteří chtějí produkovat kvalitní dokumenty a doposud nejsou v této oblasti jednoznačně orientováni.

První vydání této knihy, které se na trhu objevilo již před třemi lety, bylo již zcela rozebráno. Tato skutečnost těší zejména proto, že kniha pravděpodobně našla své čtenáře a ti zase na oplátku našli kvalitní prostředek pro tvorbu svých dokumentů. Tato skutečnost ovšem na druhé straně zavazuje nezklamat další takové zájemce a připravit jim co nejužitečnějšího průvodce v oblasti typografie a počítačové sazby.

Bylo by však jistě chybou jen „oprášit“ původní text, napravit nalezené nedostatky a nechat vytisknout. Za uplynulou dobu se mnohé změnilo a na to bylo nutné reagovat. Proto bylo nutné text značně přepracovat a podstatně změnit charakter výkladu. V této souvislosti se vynořuje řada problémů, které souvisejí s možnostmi výuky (kvalitní) počítačové sazby.

V době, kdy je celý svět osobních počítačů typu PC zahlcen programovými systémy typu Word, je velmi obtížné prosazovat cokoliv jiného. Nedaří se to ani velkým firmám, jejichž produkty jsou postupně vytlačovány z operačních i diskových pamětí. Jediným prvkem, s nímž může jiný systém uspět, je snadné dosažení kvalitního výsledku. K tomu je však v prvé řadě potřebné, aby uživatel *byl schopen* kvalitní výsledek rozeznat od nekvalitního, a to může nastat jen tehdy, bude-li disponovat alespoň základními znalostmi z typografie.

Z mých zkušeností vyplývá, že během poměrně krátké doby (zhruba 10 vyučovacích hodin) lze většinu studentů-uživatelů základními typografickými znalostmi vybavit. V tom okamžiku nastupuje problém druhý – jak co nejracionálněji realizovat typograficky bezchybný dokument. Racionalita však nespočívá jen ve volbě vhodného programového vybavení, ale také v určité technologii návrhu formátovacích značek a zpracování dokumentů.

Kvalitní výsledek – počítačový dokument – je tedy dán třemi základními pilíři: typografie + technologie + programový nástroj. V tomto duchu je koncipováno i druhé vydání učebnice určené pro začátečníky. Těžištěm tedy není jen samotný systém LaTeX, ale také potřebné prvky aplikované počítačové typografie a poznatky o strukturním značkování dokumentů.

Určitým dluhem vůči začátečníkům je stále malá technická podpora TEXovských instalací. Většinu čtenářů tohoto časopisu by jistě mrzelo, jako mrzí mne, že zájemci při snaze nainstalovat TEX nebo přizpůsobit jeho instalaci často narážejí na značné obtíže vyžadující kvalifikované systémové zásahy. Fatálním důsledkem toho může být například návrat k Wordu, protože „... ten se mi nainstaloval bez problémů“. Bezproblémová instalace je vstupní branou, za níž je uživatel daleko ochotnější překonat počáteční těžkosti a osvojit si nástroj, který mu bude velmi dobře sloužit. Pokud bude navíc při tomto překonávání nástrah knížka LATEX pro začátečníky užitečná, bylo jejího záměru bezezbytku dosaženo.

Jiří Rybička
rybicka@mendelu.cz

Překlad: Stručný úvod do systému LATEX 2ε

Michal Kočer

V dobách, kdy jsem k TEXování užíval diskový operační systém, jsem si jednoho dne nainstaloval CSTEX a byl jsem mile překvapen, že tento obsahuje i český překlad Pavla Sýkory dokumentu *LATEX — stručný popis*. Měl jsem konečně LATEXovskou příručku v češtině a mohl jsem LATEXem kazit kolegy a známé tvrdošíjně užívající ChiWriter či T602.

Čas běží neúprosně dále, kolegové používají tu nejnovější verzi Wordu a netrpělivě vyhlížejí verzi 2000. LATEXperti přešli na užívání nejnovější verze systému LATEX 2ε a nedočkavě vyhlížejí verzi 3. Ačkoli o systému LATEX 2ε mohou najít poměrně mnoho cizojazyčné literatury, dosud (listopad 1998) v knihkupectvích či FTP archivech marně hledají příručku či knihu o novém LATEXu v češtině či slovenštině.

Zhruba před rokem jsem se tuto trhlinu rozhodl vyplnit překladem elektronického, volně šiřitelného, dokumentu Tobiáše Oetikera *The Not So Short Introduction to LATEX 2ε, version 2.2beta.* Ten lze v poslední verzi najít na každém zrcadle CTANu.

Mým cílem nebylo doslovně přeložit anglický originál. Vytvořil jsem spíše jakýsi kompilát anglické verze, textu pana Sýkory a svých zkušeností. První verze spatřila světlo obrazovky počátkem března. Neustále mi nezbýval volný čas na korektury textu, proto jsem o pomoc počátkem prázdnin požádal v elektronické konferenci csTeX@cs.felk.cvut.cz. Ochotně se přihlásili a korektury provedli: Tomáš Davídek <Tomas.Davidek@cern.ch>, Tomáš Kouba <kouba@fzu.cz> a Arnošt Štědrý <arnost@uivt.cas.cz>. Chtěl bych jim všem, manželce

Martině a panu P. Olšákovi touto cestou poděkovat za jejich pomoc a podporu.

Na mě pak už zbylo pouze všechny připomínky a korektury přenést do elektronické verze dokumentu a vše vystavit ve zdrojové formě, ve formě PDF, HTML, PS a DVI na URL `http://www.cstug.cz/pub/CTAN/info/lshort/czech/`.

Nyní, po zveřejnění, už jen čekám, až se ozve první nespokojený čtenář, který bude mít konstruktivní připomínky, které povedou k dalšímu, lepšímu, dokonalejšímu *Stručnému úvodu do systému LaTeX 2ε*.

Michal Kočer
`kocer@nime.cz`

Začínáme s METAPOSTem aneb Udělejte si vlastní logo

Ladislav Dobiáš

V tomto dokumentu si řekneme o tom, co je to METAPOST, co umožňuje, jak se dá nainstalovat, a ukážeme si několik jednoduchých příkladů, podle kterých si snadno vyrobíte i nějaké vlastní logo.

Možnosti METAPOSTu

METAPOST je program se silným jazykem pro kreslení obrázků a výstupem do PostScriptu – jazyku moderních tiskáren.

METAPOST přebral část zdrojového kódu z programu METAFONT (samozřejmě se svolením autora – D. E. Knutha) a tím i většinu příkazů METAFONTu. Navíc umožňuje plně využít možnosti PostScriptu.

Protože METAPOST obsahuje část METAFONTu, nabízí i stejné základní grafické možnosti a práci s nimi. Tedy umožňuje řešit soustavy lineárních rovnic, obsahuje objekty pro křivky, obrázky (pictures), transformace, tvary pera. Navíc přibírá PostScriptové věci jako barvy, výřezy, stínování, čárkované čáry.

Z těchto základních věcí jde udělat snad vše, co dokáže papír snést. Ovšem někdy to může být dosti náročné. A protože je v METAPOSTu silný programovací jazyk, jsou na Internetu dostupná různá makra, která ulehčují práci s grafickými objekty.

Standardně jsou u METAPOSTu makra na grafy. Také existují například makra na animaci objektů, které mohou být i 3D (třírozměrné). V literatuře [5]

se udává, že „D. Knuth povedal, že nepoužíva nič iné na kreslenie diagramov, keď píše nejaké texty.“

Výstup z METAPOSTu lze jednoduše použít v sázecích systémech jako TEX, *troff* apod.

O „opačném toku dat“ – o možnosti vkládat kód TEXu do METAPOSTu se zmíním později.

Instalace METAPOSTu

Systém METAPOST je součástí distribuce TEX pro UNIX od Karla Berryho `web2c`. Pokud ji máte nainstalovanou, máte pravděpodobně nainstalován i METAPOST jako program `mpost`.

Balík `web2c` s podporou pro českou sazbu lze nalézt např. na `ftp://math.feld.cvut.cz/pub/cstex/unix`.

S instalací pod DOSem nemám zkušenosti, vím jen, že existuje již přeložený METAPOST na CTANu v adresáři `graphics/metapost/dos`.

Rozdíly mezi METAPOSTem a METAFONTem

METAFONTovým výstupem je bitmapa, zatímco z METAPOSTu vzniká PostScriptový kód. Proto v METAPOSTu odpadají tzv. ostré jednotky (sharp units).

METAFONT je primárně určen pro generováni fontů pro TEX, v METAPOSTu je tato funkce potlačena do pozadí. Proto například práce s `.tfm` soubory a podobné věci jsou v METAPOSTu dostupné až po vložení souboru `mfplain.mp`.

Dále odpadají možnosti METAFONTu zobrazit obrázky na terminálu.

Do METAPOSTu jde navíc vkládat text, který se někam umisťuje, proto tu jsou příkazy pro práci s fonty.

O využívání PostScriptových možností jsem se zmínil výše, některé z nich (práce s barvami a vyřezávání) si ukážeme v příkladech.

Přesný popis rozdílů lze nalézt v literatuře [3] s odkazy na základní knihu o METAFONTu [4].

Spuštění a jednoduchý příklad grafu

Předtím než si ukážeme, jak se METAPOST spouští, si musíme říci, jak vypadá program pro METAPOST, a jednoduchý program si ukážeme.

Program v METAPOSTu obecně obsahuje jednotlivé obrázky začínající `beginfig` (⟨ číslo ⟩)`;` a končící `endfig;`. Program končí většinou příkazem `end;`.

Standardně je u METAPOSTu soubor maker `graph.mp`, který umožňuje jednoduše vytvářet grafy.

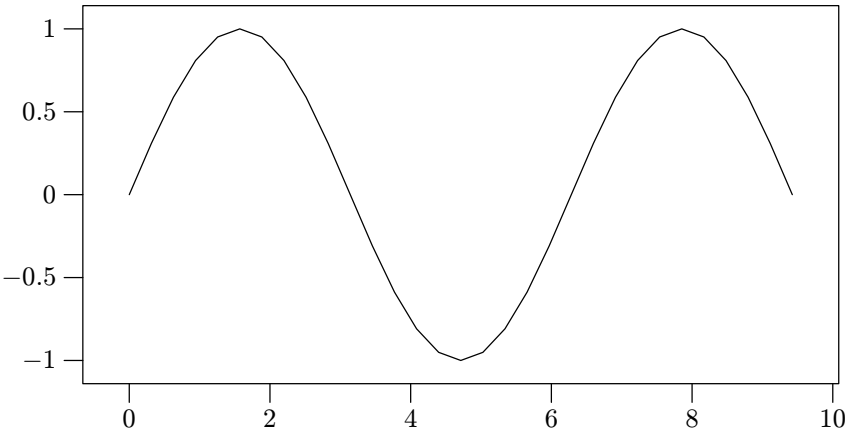
Jednoduchý graf vytvoříte programem `graf.mp`:

```
beginfig(1);
input graph;
draw begingraph(10cm,5cm);
  gdraw "graf1.d";
  endgraph;
endfig;
end;
```

METAPOST spustíme pomocí `mpost graf.mp` a vznikne PostScriptový soubor `graf.1`.

Pokud teď chcete obrázek vložit do svého TEXovského dokumentu, napište na začátek dokumentu `\input epsf` a pak někam vložte `$$\epsfbox{graf.1}$$` a máte graf vložen.

Graf vypadá takto:



Poznámky k programu:

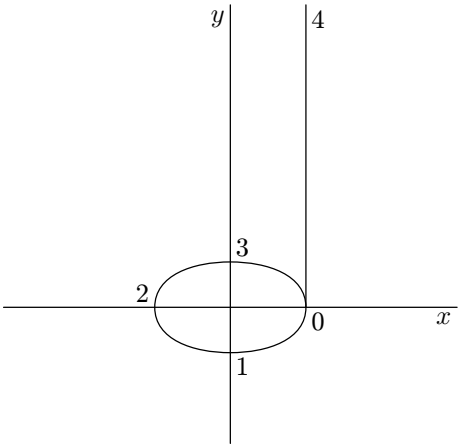
- `input` vkládá soubor maker `graph.mp`
- dále definujeme jen celkovou velikost grafu a soubor, z něhož se mají data pro graf číst
- soubor `graf1.d` obsahuje 2 sloupce čísel oddělených mezerami (souřadnice x a y)

Příklad 1 – nota

Bez další teorie si ukážeme jednoduchý příklad na spojování bodů.

```
beginfig(1);
width:=.5cm;
height:=.3cm;
stem:=1cm;

y0=x1=y2=x3=0;
-x2=x0=x4=width/2;
-y1=y3=height/2;
y4=stem;
draw z0..z1..z2..z3..cycle;
draw z0--z4;
endfig;

end;
```

Program vykreslí obrázek podobající se nějaké notě. Zde jsem ho trochu zvětšil a popsal body:



Poznámky k programu:

- `width`, `height`, `stem`, `x0`, `x1`, ... `y0`, `y1`, ... jsou proměnné typu souřadnice
- `z0`, `z1` ... je zkratka pro `(x0,y0)`, `(x1,y1)` ...
- `:=` je operace přiřazení
- `=` definuje závislosti (to jsou ty soustavy lineárních rovnic)
- `draw` spojuje body
- `..` spojuje body Bézierovými křivkami
- `cycle` uzavře křivku
- `--` spojuje body úsečkami

Pokud bychom místo prvního řádku s `draw` napsali `draw z0--z1--z2--z3--cycle;`, dostali bychom místo

„hranatou“ verzi noty

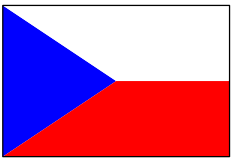
Snad jste viděli, že to je docela lehké.

Příklad 2 – vlajka

Nyní si ukážeme jednoduchou práci s barvami.

```
vyska:=2cm;
delka=3/2vyska;

beginfig(1);
y1=y2=x1=x4=0;
x2=x3=delka;
y4=y3=vyska;
z5=.5[z1,z3];
z6=.5[z2,z3];
fill z1--z4--z5--cycle withcolor blue;
fill z1--z2--z6--z5--cycle withcolor red;
fill z3--z4--z5--z6--cycle withcolor white;
draw z1--z2--z3--z4--cycle;
endfig;

end;
```

Tento program vykreslí naši vlajku



Poznámky k programu:

- `z5=.5[z1,z3]` znamená, že bod `z5` leží „na polovině cesty“ mezi body `z1` a `z3`
- barvy se dají volit pomocí předdefinovaných slov jako `blue`, `red`,... nebo pomocí složek červené, zelené, modré, jako trojice `(r,g,b)`, kde jednotlivé složky jsou v rozmezí od 0 do 1.

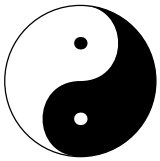
Příklad 3 – monáda

Teď si ukážeme práci s cestami, perem a jednoduchou transformaci.

Program

```
prumer:=1cm;

beginfig(1);
% Zakladni vlna a kolo
path p;
-x1=-y2=x3=y4=prumer;
x0=y0=y1=x2=y3=x4=0;
p = z1..z2..z3;
fill p{up}..z0{down}..{up}cycle;
draw p..z4..cycle;

% Tecky
pickup pencircle scaled (prumer/6);
-z5=z6=.5[z0,z3];
draw z5;
undraw z6;

% Prevratime a otocime "na bok"
currentpicture:= currentpicture xscaled -1 rotated 90;
endfig;

end;
```

vykreslí znak symbolizující věčné prolínání Jin a Jang.



Poznámky k programu:

- `path p` definuje `p` jako cestu (tu jsme použili již předtím u `draw`, aniž bychom o tom věděli)

- `fill` vyplňuje uzavřenou cestu
- `pickup pencircle scaled (prumer/6)` říká, že použijeme kruhové pero daného průměru
- `undraw` (a podobně `unfill`) maže cestu
- proměnná `currentpicture` obsahuje vše, co jsme nakreslili, a to převrátíme a otočíme o 90 stupňů

Příklad 4 – logo

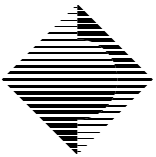
Nyní zkusíme udělat nějaké logo, např. pro firmu.

Je zde použito cyklu `for`, typu obrázek a cesta a nakonec tam jsou dvě oříznutí a spojení oříznutých části ve výsledné logo.

```
h:=2cm;       % celkova vyska (i sirka)
n:=22;        % pocet car je (n+1) !!!
wn:=(h/n)/20; % nejtenci cara (baj voko)
dd:=(h/n)/5;  % dalsi (pridavna) mezera mezi cary
h=d*(n+1);    % d je vzdalenost mezi stredy car
w0=d-wn-dd;   % nejsirsi = sirka - nejtenci - pridavna

beginfig(1);
% zakladni cary
  x0=0;
  y0=-h/2+d/2; % aby to bylo soumerne okolo 0
for i=0 upto n:
  w:=w0-(w0-wn)/n*i;  % zacneme nejtlustsi carou
  draw z0 withpen pensquare yscaled w xscaled h;
  y0:=y0+d;
endfor

% vytvorime obracenou kopii
picture c;
c = currentpicture yscaled -1;

% orizneme na spravny tvar
y1=x2=y3=x4=x5=x6=y7=0;
-x1=-y2=x3=y4;
x3-x1=h;
-y5=y6=x7=(17/32/2)*h; % "baj voko" zmereno pravitkem...

path p; % rozdelujici cara + vybouleni uprostred
p= z2--z5..z7..z6--z4;
clip currentpicture to z1--p--cycle;
clip c to z3--p--cycle;
% spojime obe casti dohromady
addto currentpicture also c;
```

```
endfig;

end;
```

Vyjde z toho takovéhle logo



Poznámky k programu:

- nejdříve nakreslíme základní čáry různé tloušťky, ale se stejnými vzdálenostmi mezi jejich osami (zde je to kresleno jako body různými pery)
- vytvoříme si otočenou kopii
- z originálu necháme (vyřízneme dle cesty) jen levou část loga
- z kopie necháme jen pravou část
- obě části spojíme

(Toto logo má jednu chybu – je to chráněná značka[1], proto ho nemůžete použít pro vlastní firmu :-)

Vkládaní TEXovského kódu do METAPOSTu

Jako součást programu pro METAPOST můžete mezi klíčová slova `btex` a `etex` vkládat příkazy pro TEX. Ty se zpracují preprocesorem, který spouští TEX na pozadí, vytvoří se soubor `.mpx`, kterému již METAPOST rozumí.

Pokud vás přesně zajímá, jak to dělá, stačí udělat mezi `btex` a `etex` nějakou chybu a v aktuálním adresáři zůstane soubor `mpxerr.tex`, ve kterém to uvidíte.

Například, pokud do souboru `chyba.mp` napíšeme `btex $x etex`, v souboru `mpxerr.tex` bude:

```
\shipout\hbox{\smash{\hbox{\hbox{% line 1 chyba.mp
$x}\vrule width1sp}}}
\end{document}
```

Také lze využít příkazů `verbatimtex` a `etex`, mezi které se pouze vloží preprocesorem do dočasného TEXovského souboru. To je vhodné např. pro vložení souboru vlastních TEXovských maker (pomocí `verbatimtex \input mamakra etex`), která se budou později využívat mezi `btex` a `etex`.

[1] Logo firmy CZ. TECH s.r.o. je použito se svolením jeho majitele.

Závěr

Doufám, že se mi podařilo ukázat, že METAPOST je poměrně jednoduše použitelný systém a že ho alespoň někdy vyzkoušíte (třeba pro vytvoření loga vaší firmy).

Pokud by vás METAPOST zaujal, tak přesnější popis a syntaxi příkazů najdete např. v literatuře [3].

Odkazy

[1] J. D. Hobby: *The METAPOST System.* Součást distribuce programu METAPOST.

[2] J. D. Hobby: *Drawing grafs with METAPOST.* Součást distribuce programu METAPOST.

[3] J. D. Hobby: *A User's Manual for METAPOST.* Součást distribuce programu METAPOST.

[4] D. E. Knuth: *The METAFONTbook.*

[5] *Často kladené otázky o TEXu a odpovědi na ně (CSTUG FAQ).* Zpravodaj Československého sdružení uživatelů TEXu, **6** (3), 129–211 (1996).

METAPOST

Robert Špalek

1. Co je to METAPOST

METAPOST je jazyk určený pro popis obrázků – technických ilustrací. Výstupem METAPOSTu je program v PostScriptu, který vykreslí na zařízení požadovaný obrázek. Oproti PostScriptu poskytuje mnoho užitečných funkcí, které kreslení zjednodušují a urychlují.

Mezi nejpoužívanější funkce bezesporu patří výpočet souřadnic bodů podle zadaných závislostí, automatické řešení soustav lineárních rovnic, výpočet průsečíků křivek, jejich plynulé navazování, inteligentní prokládání bodů křivkou, vkládání textových popisků (s možností formátování TEXem a TROFFem), kreslení kaligrafickým perem, práce s barvami, vestavěný makrojazyk,... Při popisu se používá snadno pochopitelný programovací jazyk procedurálního typu.

Vykreslené obrázky je možno použít všude, kde je možno pracovat s PostScriptem, tedy všude. Pomocí maker `epsf` je můžeme vkládat i do TEXovských

dokumentů. Dříve se na to sice používal METAFONT, ale teď už i jeho autor D. E. Knuth uznal, že už používá pouze METAPOST.

2. Jak METAPOST nainstalovat a spustit

Nejjednodušší je nainstalovat METAPOST spolu s některou distribucí TEXu, bezproblémová je např. instalace balíku teTEX (pod Linuxem ji obsahují distribuce Debian, RedHat,...). Program se spouští příkazem `mpost`, a protože je to překladač, je veškeré ovládání řízeno z příkazové řádky.

Např. obrázky `loga.mp` popíšete v libovolném textovém editoru, spustíte `mpost loga.mp`, METAPOST vykreslí jednotlivé obrázky do souborů `loga.1`, `loga.2`,...

Vygenerované obrázky je možno před předáním dál prohlédnout libovolným prohlížečem PostScriptu, např. programem `ghostview`. Pro korektní zobrazování fontů je nutno nastavit několik parametrů.

2.1. Nastavení cest k TEXovským fontům

Standardně `ghostscript` nezná Computer Modern fonty použité v TEXem sázených textech. Pokud máte nainstalovánu jejich PostScriptovou verzi, stačí k vyřešení tohoto problému pouze přidání cest do konfiguračního souboru. Přepněte se tedy do adresáře `/usr/lib/ghostscript/fonts`.

Do souboru `Fontmap` je nutno za každý potenciálně použití font přidat jeden řádek s odkazem. Nechť jsou tyto fonty uloženy v adresářích

```
/usr/lib/texmf/fonts/type1/bluesky/cm      TEX
/usr/lib/texmf/fonts/type1/bluesky/ams     AMS-TEX.
```

Pak přidejte do souboru `Fontmap` tyto řádky:

```
/cmb10 (/usr/lib/texmf/fonts/type1/bluesky/cm/cmb10.pfb) ;
/cmbsy10 (/usr/lib/texmf/fonts/type1/bluesky/cm/cmbsy10.pfb) ;
...
/lcmssi8 (/usr/lib/texmf/fonts/type1/bluesky/cm/lcmssi8.pfb) ;
/line10 (/usr/lib/texmf/fonts/type1/bluesky/cm/line10.pfb) ;
...
```

Není vhodné psát tyto řádky ručně, neboť fontů je mnoho. Nejlépe je použít nějaký Unixový filtr, např. `sed`. Od tohoto okamžiku bude `ghostscript` v pořádku zobrazovat tyto fonty ve všech velikostech. Je možné, že na jiných operačních systémech se způsob nastavování bude lišit. Uvedené cesty platí pro OS Linux/Debian.

Pokud tyto fonty nevlastníte, musíte si je nainstalovat. V PostScriptu není možné použít standardní `GF` či `PK` fonty.

2.2. Vygenerování Encapsulated PostScript

V METAPOST User Manual je to sice zmíněno, ale poněkud nevýrazně. Pokud chceme, aby byl obrázek oříznut na správnou velikost a aby se správně nadeklarovaly fonty, musí se přidat před kreslicí příkazy nějaké deklarace. METAPOST to zařídí, pokud nastavíme `prologues:=1;` V opačném případě se nám žádné textové popisky nezobrazí (místo nich nám prohlížeč ohlásí chybu) a obrázek zabírá celou stránku.

3. Schopnosti METAPOSTu

Tento seznam má sloužit pouze jako výpis možností, které poskytuje METAPOST. Pro podrobnější nastudování je nejlépe přečíst originální dokumentaci METAPOST User Manual.

3.1. Definice obrázků

Obrázek se uvozuje mezi příkazy `beginfig(1)` a `endfig;`. Jednička znamená číslo obrázku. Těchto obrázků lze v jednom souboru popsat více. Na konci souboru se píše jedno `end` navíc. Na počátku souboru se obvykle uvádějí deklarace maker, nastavení speciálních parametrů,...

3.2. Definice bodů

Veškeré kreslící operace je možno provádět přímo s danými souřadnicemi, ale daleko flexibilnější je nadefinovat si nejprve pozice bodů, pojmenovat je a poté je symbolickým zápisem ve zbytku kódu používat. Bod se umístí např. příkazem `z1=(3cm,4cm);`. Při zápisu lze využít všech obvyklých délkových jednotek, ale také pozic už nadefinovaných objektů. Souřadnice bodů lze sčítat, odčítat, násobit reálným číslem, ale také lineárně interpolovat (resp. extrapolovat). Např. bod na jedné třetině cesty mezi body `z1, z2` se popíše výrazem `1/3[z1,z2]`.

Často používanou funkcí je automatické řešení soustav lineárních rovnic. Operátor rovnítko neplní totiž funkci přiřazení, místo toho vyjadřuje přání, aby se dvě strany rovnaly. Pokud v lineárních výrazech použijeme proměnné, jejichž hodnota je zatím neznámá, METAPOST si zapamatuje jejich vzájemný vztah a postupně při výpočtu dalších přiřazení eliminuje neznámé. Pokud se ale pokusíme použít neznámou souřadnici při použití některého kreslícího příkazu, ohlásí chybu.

Nejčastější chybou při výpočtu obrázku je buď příliš málo podmínek nebo naopak jejich přemíra (pak rovnice nemají řešení).

- `z1-z2=z3-z4=z5-z6=(0,2cm)` — `z1` má být 2cm nad `z2`,...
- `z5=z1+whatever[z1,z2]=z3+whatever[z3,z4]` — `z5` je průnikem přímek

`z1--z2` a `z3--z4`, `whatever` značí neznámou, jejíž hodnotu nepotřebujeme znát. Zde je to potřeba kvůli popsání náležení přímce.
- `z3-z1=(z2-z1) rotated 60` — body `z1,z2,z3` jsou vrcholy rovnostranného trojúhelníka

Při umisťování objektů lze využít afinních transformací. Tyto transformace můžeme ukládat do proměnných (typu `transform`) nebo je přímo používat. METAPOST poskytuje tyto elementární transformace: `identity`, `scaled` (resp. `xscaled, yscaled, zscaled`), `shifted`, `slanted`, `rotated`. Při použití objektu (např. jeho vykreslením) pak připojíme `transformed t`, nebo přímo `xscaled -1`.

3.3. Kreslení polygonů a křivek

Polygony se kreslí příkazem `draw z1--z2--z3;`. Uzavřené polygony pak příkazem `draw z1--z2--z3--cycle;`. METAPOST vykreslí čáru nastaveným stylem a perem.

Zajímavější je kreslení křivek. METAPOST používá Bézierovy kubické křivky, stejné jako používá PostScript. Na rozdíl od mnoha konkurenčních programů, v METAPOSTu není nutno zadávat jejich kontrolní body. V nejjednodušším případě se pouze zadá seznam bodů, jimiž má křivka procházet, a METAPOST sám podle své heuristiky zvolí body tak, aby byla křivka co nejhezčí. Nemusíme se ale bát, že by nám METAPOST vnucoval svůj názor. Jeho chování můžeme ovlivnit nastavením mnoha lidsky srozumitelnými parametry:

1. Syntaxe základního kreslícího příkazu je `draw z1..z2..z3..z4--z5..z6;`.
2. Pokud potřebujeme v daném bodě předepsat sklon křivky, píšeme `draw z1..z2{dir 45}..{dir 30}z3{dir 10}..z5;` Lze použít předdefinované směry `up,down,left,right`.
3. Pokud se nám zdá, že křivka mezi dvěma body příliš „plandá“, můžeme upravit její napětí výrazem `z1..tension 3..z2`. Pokud chceme křivku napnout, jak to jenom jde, napíšeme `z2...z3`, což je zkratka za `z2..tension infinity..z3`.
4. Jestliže chceme na konci křivky udělat větší „kudrlinku“, píšeme `z1{curl 2}..z2..{curl 1}z3`.
5. Pokud si přesto nevybereme, zůstane nám poslední možnost zadat kontrolní body ručně, což je však málokdy potřeba. Dosáhneme toho výrazem `z1..controls z2 and z3..z4`.

Většinu těchto parametrů využijeme málokdy, METAPOSTová heuristika je skutečně vynikající. Zadávat body ručně se mi nyní jeví zbytečné. Cesty nemusíme hned kreslit, můžeme si je uložit do proměnných typu `path`.

METAPOST poskytuje spoustu funkcí pro práci s cestami:

1. Pozici jednotlivých bodů na křivce zjistíme výrazem `point 0.7 of p`, kde `p` je cesta. METAPOST použije tuto parametrizaci: zadané body jsou očíslovány přirozenými čísly, body mezi nimi jsou spojitě rozloženy podél křivky.

2. Délku (počet zadaných bodů) křivky zjistíme výrazem `length p`.
3. Část cesty vybranou jako podinterval mezi dvěma parametry získáme výrazem `subpath(1.5,length p) of p`.
4. Cestu vzniklou oříznutím po posledním protnutí s jinou cestou popíšeme výrazem `p cutafter q`.
5. Směrový vektor v daném bodě zjistíme výrazem `direction 2 of p`.
6. Parametr průsečíku dvou cest zjistíme výrazem `p intersectiontimes q` (výsledkem je dvojice parametrů pro obě křivky). Pro běžné použití se hodí makro `intersectionpoint`, které vrací přímo souřadnice bodu.
7. Parametr bodu, ve kterém křivka směřuje daným směrem, zjistíme pomocí výrazu `directiontime (1,1) of p`. Pro běžné použití oceníme zejména funkci `directionpoint (1,1) of p`, která vrátí souřadnice tohoto bodu.
8. Skutečnou délku křivky zjistíme výrazem `arclength p`.
9. Parametr bodu, do kterého je křivka dlouhá danou délkou, zjistíme výrazem `arctime 6 of p`.

Zjištěné hodnoty můžeme samozřejmě využít i při definici dalších bodů. Tato schopnost je asi největším přínosem METAPOSTu oproti PostScriptu.

3.4. Vkládání popisků do obrázku

Popisek vysázený obyčejným fontem bez dalšího zpracování vložíme příkazem `label.bot("X-axis",0.5[z1,z2]);` příp. `label("hello",(2cm,1cm));`. Uvedené prefixy specifikují, jak bude popisek zarovnán. Pokud bychom si přáli udělat v daném bodě ještě puntík, použijeme příkaz `dotlabel`.

Popisek se vysází fontem, jehož jméno je uloženo v proměnné `defaultfont`, což snadno změníme např. příkazem `defaultfont:="Times-Roman";`. Dalším důležitým parametrem je `defaultscale`.

Pokud chceme vysázet popisek pořádně některým sázecím systémem (např. TEX nebo TROFF), pak místo textu uzavřeného do uvozovek vepíšeme text mezi klíčová slova `btex,etex`. Např.

```
label.lrt(btex $\int_0^a x\;{\rm d}x={1\over2}a^2$ etex
rotated 20 scaled 1.4142, (3cm,2cm));
```

METAPOST se postará o vytvoření zdrojového souboru pro sázecí systém, jeho kompilaci, analýzu výstupního (DVI) souboru a převedení na posloupnost PostScriptových příkazů. Díky tomu není problém s výsledkem dále pracovat, např. jej natáčet, zvětšovat nebo měnit barvu.

Pokud potřebujeme TEX nějak inicializovat (vložit soubor s makry), uvedeme tyto příkazy na začátku souboru ve tvaru `verbatimtex \input mymac etex`.

Nezapomeňte na správné (nenulové) nastavení proměnné `prologues`, jinak žádné popisky neuvidíte.

3.5. Složitější grafické příkazy

Uzavřenou cestu (ukončenou klíčovým slovem `cycle`) můžeme vyplnit příkazem `fill p;` resp. `fill p withcolor red;`. Pokud potřebujeme vyplnit oblast mezi dvěma křivkami, které se nějak protínají, využijeme primitivu `buildcycle`. Např. oblast omezenou 4 křivkami vyplníme šedou barvou pomocí příkazu `fill buildcycle(p0, q0, p1, q1) withcolor 0.7white;`. Další příkaz `filldraw` je pouhým spojením příkazů `fill` a `draw`. Vedle všech těchto příkazů existují varianty `unfill`,... které oblasti mažou místo kreslení.

Díky schopnostem PostScriptu nakreslíme snadno i přerušovanou křivku. Připojíme pouze specifikaci, jak má být přerušována. Existuje více variant:

1. `dashed withdots scaled p` vyprodukuje tečkovanou čáru.
2. `dashed evenly scaled p` naopak čáru čárkovanou.
3. METAPOST umožňuje specifikovat i vlastní styl čáry. Dělá se to poněkud těžkopádně dočasným vykreslením přerušovaného vzorku do obrazové proměnné příkazem `dashpattern` a zaregistrováním vzorku. Pak už můžeme svůj styl používat stejně jako standardní.

Veškerá tato nastavení je možno nastavit jako standardní pomocí příkazu `drawoptions(dashed evenly withcolor 0.7[red,blue]);`

Tvar konců čar je možno modifikovat nastavením proměnné `linecap:=butt;` (resp. `squared, rounded`). Analogicky tvar napojení čar se mění nastavením `linejoin:= rounded;` (resp. `beveled, mitered`). Šipky se dají kreslit na rozdíl od METAFONTu přímo primitivem `drawarrow`, který bere v úvahu i zakřivení křivky.

Cesty je možno kreslit i kaligrafickým perem. Příkazem `pickup pencircle xscaled 3mm yscaled 0.5mm rotated 60;` nadefinujeme pero se sklonem 60 stupňů. Místo `pencircle` můžeme kreslit libovolným konvexním polygonem, syntaxe je `makepen((-.5,-.5)--(-.5,.5)--(.5,.5)--(.5,-.5)--cycle)`. METAPOST definuje tato pera: `pencircle, pensquare, penrazor`. Nejsem si jist, jak řeší METAPOST kolizi mezi použitím kaligrafického pera a aplikací PostScriptových parametrů na vzhled čar.

Oříznutí obrázku oblastí ohraničenou uzavřenou cestou `p` se dá docílit příkazem `clip currentpicture to p;`

K Plain METAPOSTu existuje balík maker nazvaný `boxes.mp`, který umožňuje pohodlným způsobem kreslit rámečky okolo obrázků, jednoduše je spojovat šipkami,... Balík je ideální pro kreslení grafů, schémat. Dalším používaným souborem maker je `mpgraph` pro kreslení grafů.

3.6. Datové typy a makrojazyk

Všechny objekty v METAPOSTu mají svůj datový typ, takže je možné je ukládat pro pozdější zpracování. Datové typy jsou tyto:

1. `numeric` pro uložení čísla ve fixed-point aritmetice

2. `pair` pro uložení souřadnic bodu
3. `color` pro uložení RGB barvy
4. `transform` pro lineární transformace
5. `path` pro cesty
6. `pen` pro definici pera
7. `picture` pro uložení vykresleného obrázku

V METAPOSTu nechybí ani makrojazyk. Je možno v něm programovat `for` cykly a podmíněné `if` příkazy. **Naprosto nedocenitelná** je skutečnost, že tyto jazykové konstrukce nejsou příkazy, ale makra preprocesoru, což znamená, že není problém použít `for` cyklus pro generování seznamu bodů např. při vyvolávání příkazu `draw`. Matematickou funkci pak vykreslíme takto:

```
draw (0,0) for i=1 upto 20:
  ..(i/20*w,sin(i/20*360)*h)
endfor;
```

Hustěji body funkce počítat nemusíme, neboť Bézierovy křivky aproximují většinu funkcí velice pěkně.

Makra se dají definovat s parametry i bez pomocí příkazu `def`. Např.:

```
pero(expr t)=
  pickup pencircle scaled t;
enddef;
```

Makra se dají nastavit, aby se chovala jako unární či binární operátory. Dokonce i taková základní zvyklost, jako že `z5` je zkratka za `(x5,y5)`, je pouhé makro PlainMETAPOSTu.

O makrech je toho možné říci **velice** mnoho. Zde musím čtenáře bohužel odkázat na METAPOST User Manual.

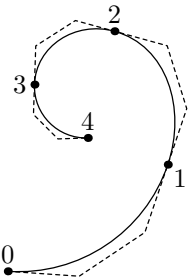
4. Ukázky

Těžištěm tohoto příspěvku jsou hlavně ukázky. Některé jsou převzaty z METAPOST User Manual, další jsou odněkud z Internetu, většina z nich pochází z mých vlastních sbírek. Původně jsem použil METAFONT k rýsování ilustrací pro matematicko-fyzikální seminář, nyní jsem je pro účely článku přepsal do METAPOSTu.

4.1. METAPOST User Manual

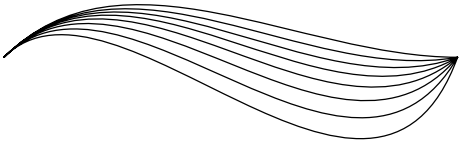
V této kapitole se nacházejí původní obrázky z METAPOST User Manual. Není na nich nic změněno, vylepšeno, či přidáno. Ukázky jsou cíleně vybrány pro demonstraci schopností METAPOSTu. Děkuji tímto autorovi METAPOSTu (*John Hobby*) za svolení k přetisknutí obrázků.

1. První ukázka slouží k demonstraci kreslení příkazem `draw`. Kromě toho jsou kolem cesty vykresleny její kontrolní body.



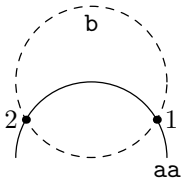

```
prologues:=1;

beginfig(1);
z0 = (0,0);    z1 = (60,40);
z2 = (40,90);  z3 = (10,70);
z4 = (30,50);
path p; p = z0..z1..z2..z3..z4;
draw p;
for t=0 upto 3:
  draw point t of p--postcontrol t of p
    --precontrol t+1 of p--point t+1 of p
    dashed (evenly scaled .5);
endfor
dotlabels.top(0,2,4);
dotlabels.lft(3);
dotlabels.lrt(1);
endfig;

end
```

2. V další ukázce pomocí `for` cyklu kreslíme křivky, ve kterých měníme počáteční úhel.



```
prologues:=1;

beginfig(1)
for a=0 upto 7:
  draw (0,0){dir 45}..{dir 10a}(6cm,0);
endfor
endfig;
end
```

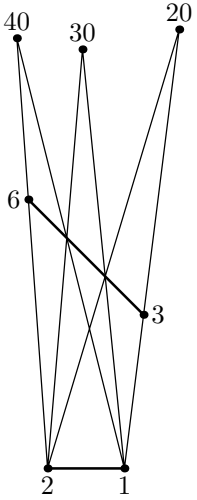
3. Demonstrace výpočtu průsečíků 2 kružnic, kreslení čárkovaných křivek.




```
prologues:=1;

beginfig(1);
path a, aa, b;
a = fullcircle scaled 2cm;
b = a shifted (0,1cm);
aa = halfcircle scaled 2cm;
draw aa;
draw b dashed evenly;
z1 = aa intersectionpoint reverse b;
z2 = reverse aa intersectionpoint b;
dotlabel.rt(btex 1 etex, z1);
dotlabel.lft(btex 2 etex, z2);
```

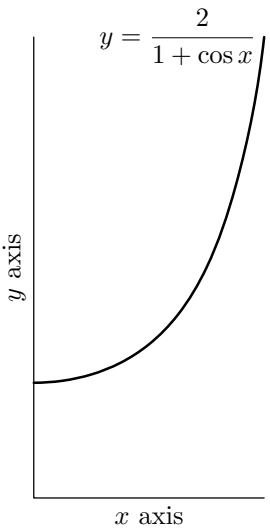
```
label.bot(btex \tt aa etex, point 0 of aa);
label.bot(btex \tt b etex, point 2 of b);
endfig;
end
```

4. Průsečíky přímek snadno zjistíme použitím lineární extrapolace. Jako parametr uvedeme `whatever`, METAPOST vyřeší lineární rovnice a dopočítá polohu bodů.




```
prologues:=1;

beginfig(1);
z1=-z2=(.2in,0);
x3=-x6=.3in;
x3+y3=x6+y6=1.1in;
z4=1/3[z3,z6];
z5=2/3[z3,z6];
z20=whatever[z1,z3]=whatever[z2,z4];
z30=whatever[z1,z4]=whatever[z2,z5];
z40=whatever[z1,z5]=whatever[z2,z6];
draw z1--z20--z2--z30--z1--z40--z2;
pickup pencircle scaled 1pt;
draw z1--z2;
draw z3--z6;
%
dotlabels.bot(1,2);
dotlabels.rt(3);
dotlabels.lft(6);
dotlabels.top(20,30,40);
endfig;
end
```

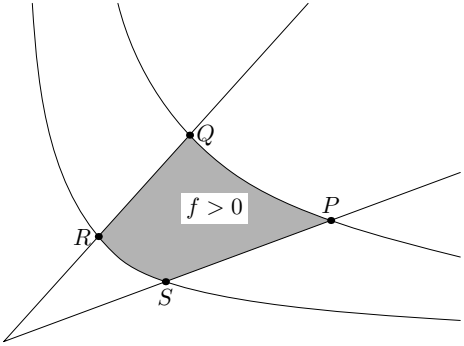
5. Matematickou funkci vykreslíme `for` cyklem, popisky k ní vysázíme TEXem. Na vysázené popisky můžeme aplikovat PostScriptovou transformaci, např. rotaci o 90°.




```
prologues:=1;

beginfig(1);
numeric ux, uy;
120ux=1.2in; 4uy=2.4in;
draw (0,4uy)--(0,0)--(120ux,0);
pickup pencircle scaled 1pt;
draw (0,uy){right}
  for ix=1 upto 8:
    ..(15ix*ux, uy*2/(1+cosd 15ix))
  endfor;
label.bot(btex $x$ axis etex, (60ux,0));
label.lft(btex $y$ axis etex rotated 90,
          (0,2uy));
label.lft(
```

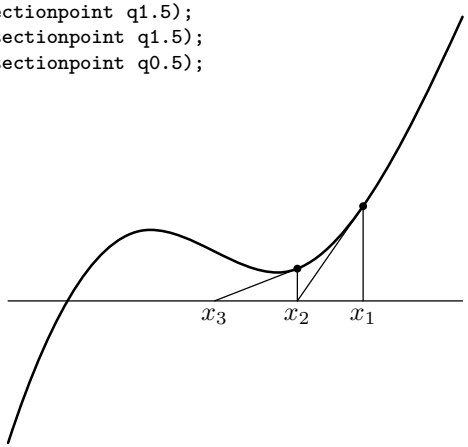
```
  btex $\displaystyle y={2\over1+\cos x}$ etex,
  (120ux, 4uy));
endfig;
end
```

6. Plochu omezenou čtyřmi funkcemi vykreslíme tímto postupem: Primitivem `buildcycle` si uložíme uzavřenou cestu, která plochu omezuje. Posléze pomocí makra `intersectionpoint` zjistíme polohy 4 vrcholů. Pak už jenom vyplníme uzavřenou oblast šedou barvou a zvýrazníme puntíky jednotlivé vrcholy.




```
prologues:=1;

beginfig(1);
h=2in; w=2.7in;
path p[], q[], pp;
for i=2 upto 4: ii:=i**2;
  p[i] = (w/ii,h){1,-ii}...(w/i,h/i)...(w,h/ii){ii,-1};
endfor
q0.5 = (0,0)--(w,0.5h);
q1.5 = (0,0)--(w/1.5,h);
pp = buildcycle(q0.5, p2, q1.5, p4);
fill pp withcolor .7white;
z0=center pp;
picture lab; lab=thelabel(btex $f>0$ etex, z0);
unfill bbox lab; draw lab;
draw q0.5; draw p2; draw q1.5; draw p4;
dotlabel.top(btex $P$ etex, p2 intersectionpoint q0.5);
dotlabel.rt(btex $Q$ etex, p2 intersectionpoint q1.5);
dotlabel.lft(btex $R$ etex, p4 intersectionpoint q1.5);
dotlabel.bot(btex $S$ etex, p4 intersectionpoint q0.5);
endfig;
end
```

7. Použitím maker snadno ilustrujeme i Newtonovu metodu tečen. Nakreslíme si křivku a primitivem `point of` zjistíme polohu bodu na křivce, primitivem `direction of` směrový vektor křivky a primitivem `intersectiontimes` polohu průsečíku. Nic nám nebrání měnit funkci, počet iterací a počáteční podmínky, abychom demonstrovali požadované vlastnosti metody.

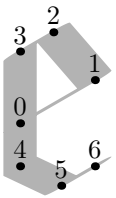


```
prologues:=1;

beginfig(1);
numeric scf, #, t[];
3.2scf = 2.4in;
path fun;
# = .1;  % Keep the function single-valued
fun = ((0,-1#)..(1,.5#){right}..(1.9,.2#){right}..{curl .1}(3.2,2#))
  yscaled(1/#) scaled scf;
x1 = 2.5scf;
for i=1 upto 2:
  (t[i],whatever) =
    fun intersectiontimes ((x[i],-infinity)--(x[i],infinity));
  z[i] = point t[i] of fun;
  z[i]-(x[i+1],0) = whatever*direction t[i] of fun;
  draw (x[i],0)--z[i]--(x[i+1],0);
  fill fullcircle scaled 3bp shifted z[i];
endfor
label.bot(btex $x_1$ etex, (x1,0));
label.bot(btex $x_2$ etex, (x2,0));
label.bot(btex $x_3$ etex, (x3,0));
draw (0,0)--(3.2scf,0);
pickup pencircle scaled 1pt;
draw fun;
endfig;
end
```

8. Zde je příklad na definici vlastního stylu čar. Nejprve vykreslíme daný vzorek, pak ho uložíme do obrazové proměnné a po smazání hlavního obrázku jej můžeme využít při kreslení elipsy.

```
prologues:=1;

beginfig(1);
draw dashpattern(on 15bp off 15bp) dashed evenly;
picture p;
p=currentpicture;
currentpicture:=nullpicture;
draw fullcircle scaled 1cm xscaled 3 dashed p;
endfig;
end
```

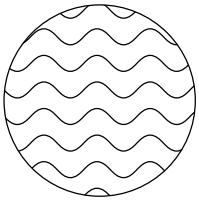
9. Při návrhu reklamní grafiky, firemních log a pro tvorbu výrazných efektů oceníme kaligrafické pero, čímž může být např. šikmo seříznutý husí brk, který držíme po dobu kreslení pod daným úhlem. Pokud bychom chtěli úhel a poloměr pera po dobu kreslení měnit, i tady nám METAPOST pomůže. Makrem `penstroke` nadefinujeme v každém kontrolním bodě potřebné parametry a vzniklou uzavřenou plochu vykreslíme příkazem `draw`.



Zde je demonstrováno obyčejné kreslení křivek kaligrafickým perem.

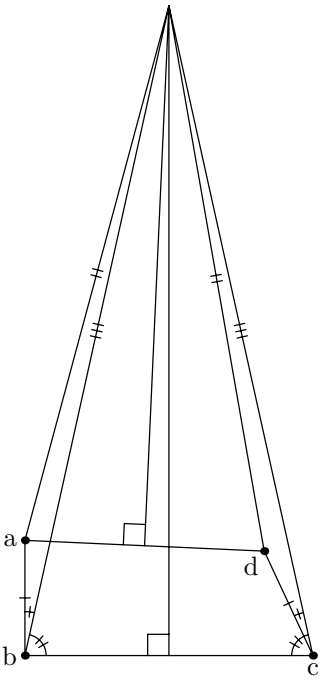
```
prologues:=1;

beginfig(1);
pickup pencircle scaled .2in yscaled .08 rotated 30;
x0=x3=x4;
z1-z0 = .45in*dir 30;
z2-z3 = whatever*(z1-z0);
z6-z5 = whatever*(z1-z0);
z1-z6 = 1.2*(z3-z0);
rt x3 = lft x2;
x5 = .55[x4,x6];
y4 = y6;
lft x3 = bot y5 = 0;
top y2 = .9in;
draw z0--z1--z2--z3--z4--z5--z6 withcolor .7white;
dotlabels.top(0,1,2,3,4,5,6);
endfig;
end
```

10. Již nakreslený vzorek můžeme oříznout uzavřenou křivkou příkazem `clip`.



```
prologues:=1;

beginfig(1);
path p[];
p1 = (0,0){curl 0}..(5pt,-3pt)..{curl 0}(10pt,0);
p2 = p1..(p1 yscaled-1 shifted(10pt,0));
p0 = p2;
for i=1 upto 3:  p0:=p0.. p2 shifted (i*20pt,0);
  endfor
for j=0 upto 8:  draw p0 shifted (0,j*10pt);
  endfor
p3 = fullcircle shifted (.5,.5) scaled 72pt;
clip currentpicture to p3;
draw p3;
endfig;
end
```

11. Pokud si nadefinujeme makra pro zvýrazňování úhlů a úseček, můžeme vykreslit i tuto konstrukci.

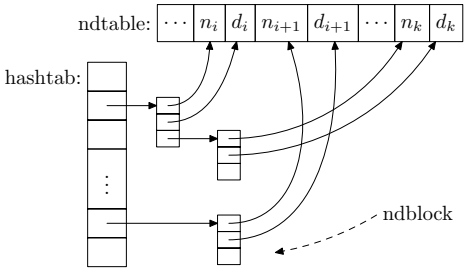



```
prologues:=1;

marksize=4pt;
angle_radius=8pt;

def draw_mark(expr p, a) =
  begingroup
  save t, dm; pair dm;
  t = arctime a of p;
  dm = marksize*unitvector direction t of p
```

```
    rotated 90;
  draw (-.5dm.. .5dm) shifted point t of p;
  endgroup
enddef;

def draw_marked(expr p, n) =
  begingroup
  save amid;
  amid = .5*arclength p;
  for i=-(n-1)/2 upto (n-1)/2:
    draw_mark(p, amid+.6marksize*i);
  endfor
  draw p;
  endgroup
enddef;

def mark_angle(expr a, b, c, n) =
  begingroup
  save s, p; path p;
  p = unitvector(a-b){(a-b)rotated 90}..unitvector(c-b);
  s = .9marksize/length(point 1 of p - point 0 of p);
  if s<angle_radius: s:=angle_radius; fi
  draw_marked(p scaled s shifted b, n);
  endgroup
enddef;

def mark_rt_angle(expr a, b, c) =
  draw ((1,0)--(1,1)--(0,1))
       zscaled (angle_radius*unitvector(a-b)) shifted b
enddef;

beginfig(1);
pair a,b,c,d;
b=(0,0); c=(1.5in,0); a=(0,.6in);
d-c = (a-b) rotated 25;
dotlabel.lft("a",a);
dotlabel.lft("b",b);
dotlabel.bot("c",c);
dotlabel.llft("d",d);
z0=.5[a,d];
z1=.5[b,c];
(z.p-z0) dotprod (d-a) = 0;
(z.p-z1) dotprod (c-b) = 0;
draw a--d;
draw b--c;
draw z0--z.p--z1;
draw_marked(a--b, 1);
draw_marked(c--d, 1);
draw_marked(a--z.p, 2);
draw_marked(d--z.p, 2);
draw_marked(b--z.p, 3);
```

```
draw_marked(c--z.p, 3);
mark_angle(z.p, b, a, 1);
mark_angle(z.p, c, d, 1);
mark_angle(z.p, c, b, 2);
mark_angle(c, b, z.p, 2);
mark_rt_angle(z.p, z0, a);
mark_rt_angle(z.p, z1, b);
endfig;
end
```

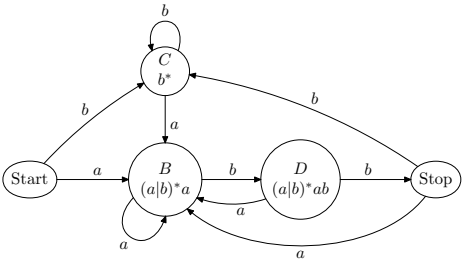
12. Při sazbě technických článků je často potřeba vykreslit komplikovaný graf nejrůznějších závislostí plný rámečků, šipek a popisků. Pomocí maker ze souboru `boxes.mp` popíšeme souřadnice a METAPOST nám pomůže nakreslit co nejpřehlednější šipky.




```
prologues:=1;

input boxes


beginfig(1);
boxjoin(a.se=b.sw; a.ne=b.nw);
boxit.a(btex\strut$\cdots$ etex);    boxit.ni(btex\strut$n_i$ etex);
boxit.di(btex\strut$d_i$ etex);      boxit.ni1(btex\strut$n_{i+1}$ etex);
boxit.di1(btex\strut$d_{i+1}$ etex); boxit.aa(btex\strut$\cdots$ etex);
boxit.nk(btex\strut$n_k$ etex);      boxit.dk(btex\strut$d_k$ etex);
drawboxed(di,a,ni,ni1,di1,aa,nk,dk); label.lft("ndtable:", a.w);
interim defaultdy:=7bp;
boxjoin(a.sw=b.nw; a.se=b.ne);
boxit.ba(); boxit.bb(); boxit.bc();
boxit.bd(btex $\vdots$ etex); boxit.be(); boxit.bf();
bd.dx=8bp; ba.ne=a.sw-(15bp,10bp);
drawboxed(ba,bb,bc,bd,be,bf); label.lft("hashtab:",ba.w);
vardef ndblock suffix $ =
  boxjoin(a.sw=b.nw; a.se=b.ne);
  forsuffixes $$=$1,$2,$3: boxit$$(); ($$dx,$$dy)=(5.5bp,4bp);
  endfor; enddef;
ndblock nda;  ndblock ndb;  ndblock ndc;
nda1.c-bb.c = ndb1.c-nda3.c = (whatever,0);
xpart ndb3.se = xpart ndc1.ne = xpart di.c;
ndc1.c - be.c = (whatever,0);
drawboxed(nda1,nda2,nda3, ndb1,ndb2,ndb3, ndc1,ndc2,ndc3);
drawarrow bb.c -- nda1.w;
drawarrow be.c -- ndc1.w;
drawarrow nda3.c -- ndb1.w;
drawarrow nda1.c{right}..{curl0}ni.c cutafter bpath ni;
drawarrow nda2.c{right}..{curl0}di.c cutafter bpath di;
```

```
drawarrow ndc1.c{right}..{curl0}ni1.c cutafter bpath ni1;
drawarrow ndc2.c{right}..{curl0}di1.c cutafter bpath di1;
drawarrow ndb1.c{right}..nk.c cutafter bpath nk;
drawarrow ndb2.c{right}..dk.c cutafter bpath dk;
x.ptr=xpart aa.c;   y.ptr=ypart ndc1.ne;
drawarrow subpath (0,.7) of (z.ptr..{left}ndc3.c) dashed evenly;
label.rt(btex \strut ndblock etex, z.ptr); endfig;
end
```

13. Konečné automaty pomocí těchto maker nakreslíme velice snadno. Výsledek je efektní.



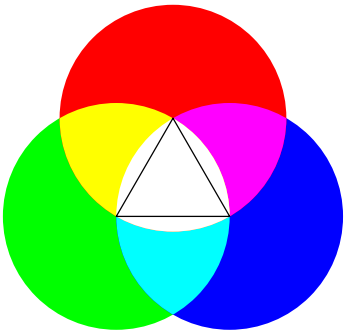

```
prologues:=1;

input boxes

vardef drawshadowed(text t) =
  fixsize(t);
  forsuffixes s=t:
    fill bpath.s shifted (1pt,-1pt);
    unfill bpath.s;
    drawboxed(s);
  endfor
enddef;

vardef cuta(suffix a,b) expr p =
  drawarrow p cutbefore bpath.a cutafter bpath.b;
  point .5*length p of p
enddef;

vardef self@# expr p =
  cuta(@#,@#) @#.c{curl0}..@#.c+p..{curl0}@#.c  enddef;

beginfig(1);
verbatimtex \def\stk#1#2{$\displaystyle{\matrix{#1\cr#2\cr}}$} etex
circleit.aa(btex\strut Start etex); aa.dx=aa.dy;
circleit.bb(btex \stk B{(a|b)^*a} etex);
circleit.cc(btex \stk C{b^*} etex);
circleit.dd(btex \stk D{(a|b)^*ab} etex);
circleit.ee(btex\strut Stop etex); ee.dx=ee.dy;
numeric hsep;
bb.c-aa.c = dd.c-bb.c = ee.c-dd.c = (hsep,0);
cc.c-bb.c = (0,.8hsep);
xpart(ee.e - aa.w) = 3.8in;
drawboxed(aa,bb,cc,dd,ee);
label.ulft(btex$b$etex, cuta(aa,cc) aa.c{dir50}..cc.c);
label.top(btex$b$etex, self.cc(0,30pt));
label.rt(btex$a$etex, cuta(cc,bb) cc.c..bb.c);
label.top(btex$a$etex, cuta(aa,bb) aa.c..bb.c);
label.llft(btex$a$etex, self.bb(-20pt,-35pt));
label.top(btex$b$etex, cuta(bb,dd) bb.c..dd.c);
```

```
label.top(btex$b$etex, cuta(dd,ee) dd.c..ee.c);
label.lrt(btex$a$etex, cuta(dd,bb) dd.c..{dir140}bb.c);
label.bot(btex$a$etex, cuta(ee,bb) ee.c..tension1.3 ..{dir115}bb.c);
label.urt(btex$b$etex, cuta(ee,cc) ee.c{(cc.c-ee.c)rotated-15}..cc.c);
endfig;
end
```

4.2. Barevné kreslení

Následující obrázky jsou v originálu plné barev. Na černobílém tisku si můžete vychutnat maximálně odstíny šedi.

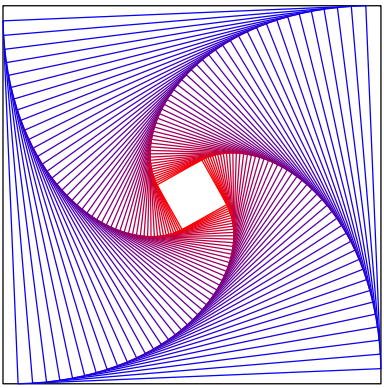
1. Aditivní mísení barev nakreslíme postupným překrýváním oblastí. Nejprve vyplníme do kruhů 3 základní barvy, pak vykreslíme po dvou jejich průniky a výsledek překryjeme bílým středem s vyznačeným trojúhelníkem.



```
prologues:=1;

beginfig(1);
 path a[],aa[];
 color cl[];
 picture p;
 cl1=green;
 cl2=blue;
 cl3=red;
 w=3cm;
 r=1.5cm;
 for i=1 upto 3:
  a[i]=fullcircle scaled (r*2);
 endfor
 x.a1+x.a2=w;
 x.a3=w/2;
 x.a2-x.a1=r;
 z.a3-z.a1=r * (cosd 60, sind 60);
 z.a2-z.a3=r * (cosd -60, sind -60);
 y.a1+y.a3=w;

 for i=1 upto 3:
  aa[i]=a[i] shifted z.a[i];
  fill aa[i] withcolor cl[i];
 endfor

 for i=1 upto 3:
  j:=i+1; if j=4: j:=1 fi;
  p := image(fill aa[i] withcolor cl[i]+cl[j]);
  clip p to aa[j];
  draw p ;
 endfor
```

```
 clip p to aa2;
 draw p withcolor cl1+cl2+cl3;
 draw z.a1--z.a2--z.a3--cycle;
endfig;

end
```

2. V druhém příkladě kreslíme do sebe zanořené čtverce s použitím afinních transformací. Každý další čtverec vychází z předchozího, pouze se zmenší a pootočí. Kromě toho plynule měníme barvu z modré do červené.



```
prologues:=1;

beginfig(1);
 path squares[];
 numeric side;
 side=5cm;
 squares0=unitsquare scaled side;
 draw squares0;
 y1=0;x2=side;
 x1=side/25;y2=x1;
 d=angle(z2-z1);
 r=abs(z2-z1)/side;
 for i:=1 upto 50:
  squares[i]:=squares[i-1] shifted (-side/2,-side/2)
  rotated d scaled r shifted (side/2,side/2);
  draw squares[i] withcolor (i/50)[blue,red];
 endfor
endfig;

end
```

3. Následující obrázek není žádná užitečná ilustrace, je to „splácanina“ všeho možného, co mě napadlo demonstrovat na METAPOSTu. Např. kaligrafické pero, více barev, sázení matematických vzorečků, rotace nápisů, TEXovské akcenty…

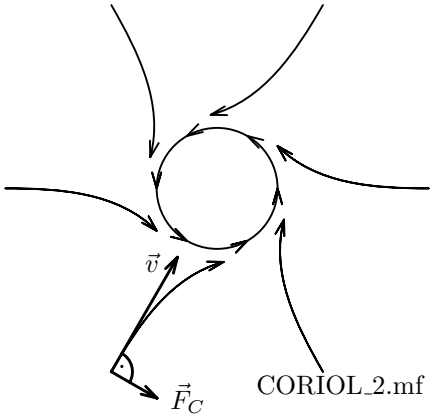

```
prologues:=1;

cmm:=cm/2;

beginfig(1)
fill fullcircle scaled 3cmm
shifted (3cmm,3cmm) withcolor (red+green);
```

```
pickup penrazor scaled 3mm;
z1=(0,0);
z2=(3,2)*cmm;
z3=(0.7,1.1)*cmm;
draw z1..z2..z3{up}..cycle withcolor red;
defaultfont:="cmss10";
label(btex $\int_0^ax\;{\rm d}x={a^2\over2}$ etex
rotated 20 scaled 4, (1,1)*cmm);
label("ahoj" infont defaultfont scaled 4,
(1.5,3)*cmm) withcolor green;
label(btex aho\v\j etex scaled 4,(5,0)*cmm);
endfig;

end;
```

4.3. Skutečně použité obrázky

Tyto obrázky jsou doprovodné ilustrace k příspěvkům dopisovatelů korespondenčního matematicko-fyzikálního semináře M&M. Kreslil jsem je poslední dva roky. Mezitím jsem si vyrobil menší soubor maker pro kreslení úhlů atp. Původně byly kresleny v METAFONTu, pro účely článku jsem je přepsal do METAPOSTu. Tyto dva jazyky jsou v podstatě stejné, liší se jenom syntaxe hlaviček obrázků.

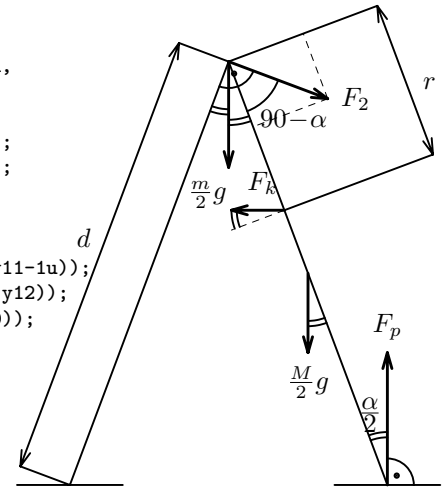
1. Rotačně symetrický obrázek plný zakřivených vektorů nakreslíme např. takto: nadefinujeme si makro, které vykreslí jednu z šipek, a pak `for` cyklem vyvoláme makro podél obvodu kruhu.



CORIOL_2.mf

```
prologues:=1;

input makra

u:=0.7mm;

%vykresleni uhlu sevreneho bodu a,b,c;
% polomer oblouku je delka
def dejuhel(expr a,b,c,delka) =
draw (b+dir(angle(a-b))*delka){dir(angle(a-b)+90)}
 ..{dir(angle(c-b)+90)}(b+dir(angle(c-b))*delka);
enddef;

def kruznice(expr S,r) =
 draw fullcircle scaled 2r shifted S;
enddef;

def min(expr x,y) =
if x<y: x else: y fi
enddef;
```

```
def KrivaSipka(expr bod,uhel,delka)=
begingroup
 save x,y;
 z1=(0,0); z2=(r'-r,-8u);
 draw ((z1{dir(0)}..{dir(-45)}z2)
  rotated uhel shifted bod);
 draw (sipka(delka,-45) shifted z2
  rotated uhel shifted bod);
endgroup;
enddef;

beginfig(1)
 w:=80u;
 h:=80u;
 v:=25u;
 Fc:=10u;
 r':=min(w,h)/2;
 r:=min(w,h)/7;
 pocet:=6; delka:=3u;
 z1=(w/2,h/2);
 pero(.4u);
 kruznice(z1,r);
 for i:=(floor(pocet/2)-pocet) upto pocet:
  draw (sipka(delka,(90+i/pocet*360))
   shifted (z1+r*dir(i/pocet*360)));
  KrivaSipka(z1+r'*dir(i/pocet*360),
   i/pocet*360+180,delka);
 endfor;
 pero(.6u);
 vektor(z1+r'*dir(4/pocet*360),v,delka,
  4/pocet*360+180);
 vektor(z1+r'*dir(4/pocet*360),Fc,delka,
  4/pocet*360+90);
 z10=z1+r'*dir(4/pocet*360);
 z11=z10+(v,0)rotated (4/pocet*360+180);
 z12=z10+(Fc,0)rotated (4/pocet*360+90);
 dejuhel(z12,z10,z11,4u);
 drawdot z10+(2u,1u);

 label.lft(btex $\vec v$ etex,(x11-2u,y11-1u));
 label.rt(btex $\vec F_C$ etex,(x12+1u,y12));
 label.ulft(btex CORIOL\_2.mf etex,(w,0));
endfig;

end
```



2. Při kreslení štaflí je potřeba popsat spoustu délek, úhlů,... Pro tyto účely vytvoříme makra `oblouk`, `vektor`, `XPopisek`. Celý další kód

STAFLE_1.mf

je v podstatě jenom kreslení těchto maker na příslušných místech.

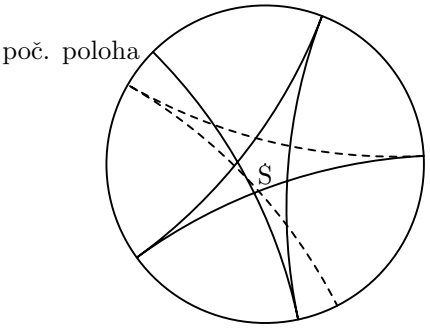
```
verbatimtex \def\ds{\displaystyle}
\def\Over{\over\ds}etex;
prologues:=1;

input makra
input mkmakra

u:=0.7mm;

beginfig(1)
 w:=90u;
 h:=100u;
 path p[];
 s:=w/90;
 v:=h/100;
 hust=.05;
 polomer:=10u;
 polomer':=9u;
 r:=30u;
 z1=(10s,10v); z2=z1+60s*right;
 x3=1/2[x1,x2]; y3=90v;
 z4=1/2[z3,z2];
 z5=z3-r*dir(angle(z3-z2));
 alfa=angle(z3-z1);
 mg:=20u; Mg:=15u;
 Fk:=10u; Fp:=mg/2+Mg;
 F:=20u;
 z6=z5+whatever*dir(angle(z3-z2)+90);
 x6=x5-Fk;
 z7=z3+F*dir(angle(z1-z3)+90);
 z8=z2+whatever*(z3-z2);
 z8=z7+whatever*(z5-z6);
 z9=z3+whatever*(z5-z6);
 z9=z7+whatever*(z3-z2);
 delka:=3u;
 pero(.2u);
 draw z5--z6 dashed evenly scaled 0.3mm;
 draw z7--z8 dashed evenly scaled 0.3mm;
 draw z7--z9 dashed evenly scaled 0.3mm;
 pero(.4u);
 draw z1--z3--z2;
 draw((-10u,0)--(10u,0)) shifted z1);
 draw((-10u,0)--(10u,0)) shifted z2);
 XPopisek(z1,z3,delka,0,10u);
 XPopisek(z3,z5,delka,0,30u);
 PUhel(z2,0,polomer);
 PUhel(z3,angle(z1-z3),polomer);
 oblouk(z2,polomer,90,angle(z3-z2));
 oblouk(z3,polomer,angle(z1-z3),-90);
 oblouk(z3,polomer+2u,-90,angle(z2-z3));
```

```
 oblouk(z3,polomer,angle(z2-z3),
  (angle(z1-z3)+90));
 oblouk(z4,polomer,-90,angle(z4-z3));
 oblouk(z5,polomer,180,(angle(z3-z2)+90));
 oblouk(z2,polomer',90,angle(z3-z2));
 oblouk(z3,polomer',angle(z1-z3),-90);
 oblouk(z3,polomer'+2u,-90,angle(z2-z3));
 oblouk(z4,polomer',-90,angle(z4-z3));
 oblouk(z5,polomer',180,(angle(z3-z2)+90));
 pero(.6u);
 vektor(z2,Fp,delka,90);
 vektor(z4,Mg,delka,-90);
 vektor(z5,Fk,delka,180);
 vektor(z3,mg,delka,-90);
 vektor(z3,F,delka,angle(z1-z3)+90);

 label.ulft(btex $d$ etex,
  (1/2[x1,x3]-10u,1/2[y1,y3]+4u));
 label.rt(btex $r$ etex,
  (1/2[x3,x5]+30u,1/2[y3,y5]+10u));
 label.rt(btex $F_2$ etex,(x7+1u,y7));
 label.bot(btex ${m\over2}g$ etex,
  (x3-4u,y3-mg-1u));
 label.urt(btex $F_k$ etex,
  (x5-Fk+2u,y5+2u));
 label.bot(btex ${M\over2}g$ etex,
  (x4,y4-Mg-1u));
 label.top(btex $F_p$ etex,(x2,y2+Fp+1u));
 label.ulft(btex $\ds\alpha\Over2$ etex,
  (x2,y2+9u));
 label.llft(btex $90\!-\!\alpha$ etex,
  (x7+1u,y7-1u));
 label(btex STAFLE\_1.mf etex,(w/2,0));
endfig;

end
```

3. Foucaltovo kyvadlo je vykresleno přesně velice elegantním trikem. Zvolíme si úhlovou odchylku od 180° (zde 45°) a definujeme body rotací okolo středu. Posléze vykreslíme přibližnou dráhu kyvadla pomocí 2 následujících bodů obratu, jako směrnici křivky v těchto bodech zvolíme vektor směrem do středu rotace.

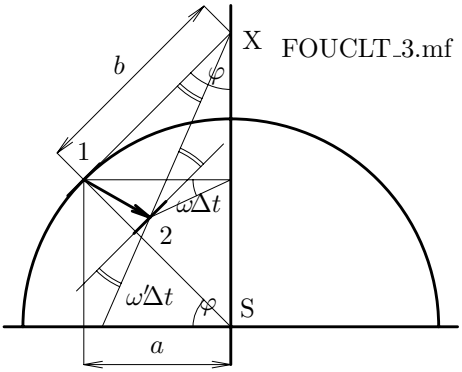
```
prologues:=1;
```

FOUCLT_1.mf



```
input makra

u:=0.7mm;

beginfig(1)
 w:=80u;
 h:=80u;
 polomer:=30u;
 z1=(40u,40u);
 z2=(polomer,0) rotated 135;
 for i:=3 upto 8:
  z[i]=z[i-1] rotated 147;
 endfor;
 pickup pencircle scaled 0.4u;
 drawdot z1;
 draw fullcircle scaled 2polomer shifted z1;
 for i:=3 upto 6:
  draw (z1+z[i-1]){-z[i-1]}..{z[i]}(z1+z[i]);
 endfor;
 draw (z1+z[6]){-z[6]}..{z[7]}(z1+z[7])
  dashed evenly scaled 0.3mm;
 draw (z1+z[7]){-z[7]}..{z[8]}(z1+z[8])
  dashed evenly scaled 0.3mm;

 label(btex S etex,(x1,y1-2u));
 label.lft(btex po\v c. poloha etex,
  (x1+x2-1u,y1+y2));
 label.bot(btex FOUCLT\_1.mf etex,(w/2,h));
endfig;

end
```

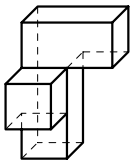
4. Detail působení sil na Foucaltovo kyvadlo. Práce s kreslením úhlů mezi body je ulehčena makrem `dejuhel`, které převezme trojici bodů a poloměr úhlu a samo vykreslí křivku na správném místě.




```
prologues:=1;

jedn=5mm;
u=1/10jedn;
cara=0.4mm;
popis=0.1mm;

uhelsipky=10;
delkasipky=0.5jedn;
```

```
delkauhlu=1jedn;

def sipka(expr odkud,kam,delka) =
draw (kam-dir(angle(kam-odkud)+uhelsipky)*delka)--kam
   --(kam-dir(angle(kam-odkud)-uhelsipky)*delka);
enddef;

def sipkaa(expr kam,uhel,delka) =
draw (kam-dir(uhel+uhelsipky)*delka)--kam
   --(kam-dir(uhel-uhelsipky)*delka);
enddef;

def dejuhel(expr a,b,c,delka,znamenko) =
draw (b+dir(angle(a-b))*delka){dir(angle(a-b)+90)}
   ..{dir(angle(c-b)+90)}(b+dir(angle(c-b))*delka);
enddef;

beginfig(1)
w:=11jedn;
h:=9jedn;
RR=5.5jedn;
z1=(6jedn,0);
z2=(0,0); z3=(12jedn,0);
z4=(6jedn,-1jedn); z5=(6jedn,8.5jedn);

pickup pencircle scaled cara;
draw halfcircle scaled 2RR shifted z1;
draw z2--z3; draw z4--z5;

pickup pencircle scaled popis;
z6=(RR,0) rotated 135 shifted z1;
z7=(x6,y4);
draw z1--z6--z7--z4;
sipka(z4,z7,delkasipky);
sipka(z7,z4,delkasipky);
z8=(x1,y6);
z9=(7/8[x1,x7],y1);
z10=(x1,whatever)=z6+whatever*dir(angle(z6-z1)+90);
draw z9--z10;
draw z10--z6--z8;
dejuhel(z6,z1,z2,delkauhlu,+1);
dejuhel(z6,z10,z9,2*delkauhlu,+1);
dejuhel(z6,z10,z9,1.9*delkauhlu,+1);
dejuhel(z6,z10,z1,1.5*delkauhlu,+1);

uhel=25;
z11=whatever[z10,z9]=z8+whatever*dir uhel;
draw z8--z11--z6;
uhel:=angle(z10-z6);
z13=z11+(RR/2)*dir uhel;
z12=z11-(RR/2)*dir uhel;
```

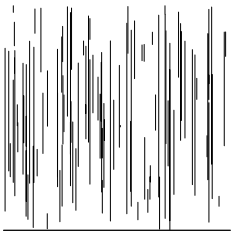
```
draw z12--z13;
dejuhel(z6,z8,z11,delkauhlu,+1);
dejuhel(z13,z11,z10,2*delkauhlu,+1);
dejuhel(z12,z11,z9,2*delkauhlu,+1);
dejuhel(z13,z11,z10,1.9*delkauhlu,+1);
dejuhel(z12,z11,z9,1.9*delkauhlu,+1);

z14=z6+(1jedn,0) rotated 135;
z15=z14+z10-z6;
draw z6--z14--z15--z10;
sipka(z14,z15,delkasipky);
sipka(z15,z14,delkasipky);

pickup pencircle scaled cara;
draw z6--z11;
sipka(z6,z11,delkasipky);
z16=z6-(0.6jedn,0) rotated 225; z17=2*z6-z16;
z18=z11+z16-z6; z19=z11+z17-z6;
draw z16--z17;
draw z18--z19;

label.urt(btex S etex,(x1+1u,y1+1u));
label.top(btex 1 etex,(x6,y6+3u));
label.lrt(btex 2 etex,(x11+1u,y11-1u));
label.rt(btex X etex,(x10+1u,y10-2u));
label.top(btex $a$ etex,((x7+x4)/2,y7+1u));
label.ulft(btex $b$ etex,((x14+x15)/2,(y14+y15)/2));

label.ulft(btex $\varphi$ etex,(x10,y10-1.5jedn));
label.top(btex $\varphi$ etex,(x1-0.6jedn,y1+1u));
label(btex $\omega'\!\!\Delta t$ etex,(x11,y12-1u));
label.ulft(btex $\omega\!\Delta t$ etex,(x8-1u,y11));
label.llft(btex FOUCLT\_3.mf etex,(x3,y10));
endfig;

end;
```

5. Prostorová tetrisová kostička.

Kromě prostorových kostiček jsme publikovali také rovinné kostičky až do řádu 7. Celkem jich je více než 100. Samozřejmě jsme je nekreslili ručně. Kód byl vygenerován speciálním programem ve vyšším progr. jazyce vyrobeném za tímto účelem. Vygenerovaný kód není nijak úsporný ani zajímavý, takže ho neuvádím, chtěl jsem jen napsat, že takto to jde taky.

```
prologues:=1;

input makra
input mkmakra

u=.6mm;
```

```
beginfig(1)
 w:=40u;
 h:=40u;
 numeric s,v,uhel,zpart,hust;
 s:=w/4;
 v:=h/4;
 uhel:=45;
 zpart:=s*cosd(uhel)++v*sind(uhel);
 z1=(1s,0v);
 z2=z1+s*right;
 z3=z2 ZMove(zpart,uhel,1/2);
 z4=z3+s*left;
 z7=z2+v*up;
 z8=z7+s*left;
 z6=z7 ZMove(zpart,uhel,-1/2);
 z5=z6+s*left;
 z10=z6+v*up;
 z9=z10+s*left;
 z11=z7+v*up;
 z15=z8+v*up;
 z14=z3+2v*up;
 z12=z11+s*right;
 z13=z14+s*right;
 z16=z15+v*up;
 z17=z12+v*up;
 z18=z13+v*up;
 z19=z4+3v*up;
 x20=x3; y20=y11;
 x21=x1; y21=y5;
 pero(.5u);
 draw z21--z1--z2--z3--z20;
 draw z2--z11;
 draw z7--z6--z5--z9--z10--z6;
 draw z10--z11;
 draw z9--z15--z12--z17--z16--z15;
 draw z12--z13--z18--z19--z16;
 draw z17--z18;
 pero(.15u);
 draw z1--z4 dashed evenly scaled 0.3mm;
 draw z3--z4 dashed evenly scaled 0.3mm;
 draw z19--z4 dashed evenly scaled 0.3mm;
 draw z8--z5 dashed evenly scaled 0.3mm;
 draw z21--z8 dashed evenly scaled 0.3mm;
 draw z8--z7 dashed evenly scaled 0.3mm;
 draw z14--z13 dashed evenly scaled 0.3mm;
 draw z14--z20 dashed evenly scaled 0.3mm;
 draw z11--z14 dashed evenly scaled 0.3mm;
endfig;

end;
```

6. Ukázka interního generátoru náhodných čísel. METAPOST umí generovat náhodná čísla, a to hned ve dvou často používaných rozděleních – rovnoměrném a normálním. Detail struktury papíru byl přiblížen pomocí struktury náhodných rovnoběžných čar.

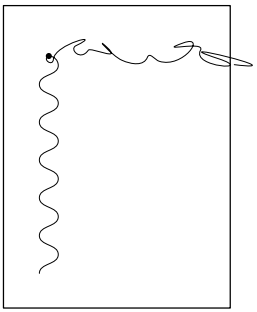


zvětšený papír

```
prologues:=1;

u:=1cm;
l:=0.2mm;
t:=0.1mm;

beginfig(1)
  w:=3u;
  h:=3u;
  z1=(0,0); z2=(w,0);
  pickup pencircle scaled l;
  draw z1--z2;
  pickup pencircle scaled t;
  for i=1 upto 100:
    x3:=uniformdeviate w;
    x4:=x3;
    y3:=uniformdeviate h;
    y4:=uniformdeviate h;
    draw z3--z4;
  endfor;
  label.bot(btex zv\v et\v sen\'y pap\'\i r etex,
   (w/2,-4l));
endfig;

end
```

7. Ukázka stopy, jak jehla trhá papír ve dvou kolmých směrech. Obrázek není samozřejmě dokonalý, chtěli jsme jen ukázat, že v jednom směru se stopa přibližuje sinusovce, zatímco v druhém náhodně trhá papír. Použil jsem normální rozdělení.

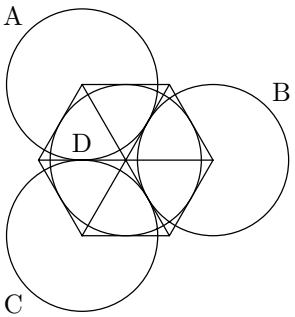
jehla



```
prologues:=1;

u:=1cm;
l:=0.2mm;
t:=0.1mm;

beginfig(1)
  w:=3u;
  h:=4u;
  z1=(1/5w,5/6h);
```

```
  pickup pencircle scaled 4l;
  drawdot z1;
  pickup pencircle scaled l;
  draw (0,0)--(w,0)--(w,h)--(0,h)--cycle;
  pickup pencircle scaled t;
  draw z1
    for i=1 upto 6:
      ..(z1+(u/8,(3/4-i)*u/2)){down}
      ..(z1+(-u/8,(1/4-i)*u/2)){down}
    endfor
    {down};
  draw z1
    for i=0 upto 10:
      ..(z1+(i*u/4+normaldeviate*(u/20),
                  normaldeviate*(u/10)))
        {dir uniformdeviate 360}
    endfor
    ;
  label.urt(btex jehla etex,(x1,h+2l));
endfig;

end
```

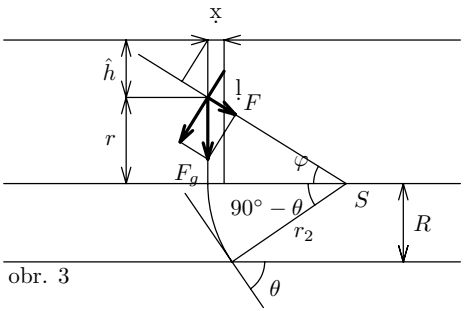
8. Geometrické uspořádání kuliček hrachu nasypaných pravidelně do krabice. Kód by šel vylepšit využitím symetrie, ale ručně spočítat souřadnice funguje taky.




```
prologues:=1;

u=1mm;
v=1mm;

%def sipky
def sipka(expr cil,delka,smer)=
draw ((-1,0.3)--(0,0)--(-1,-0.3)) scaled delka rotated
smer shifted cil; enddef;

def kolo(expr stred,polomer)=
draw (fullcircle scaled 2polomer shifted stred);enddef;


beginfig(1)
w:=40u;
h:=40u;
pickup pencircle scaled 0.2v;
z100=(0,0);z101=(0,h);z102=(w,h);z103=(w,0);
%draw z100--z101--z102--z103--z100;

RR=10u;
z1=(RR,RR);
```

```
kolo(z1,RR);
z2=(RR,3RR);
kolo(z2,RR);
z3=(RR+(sqrt(3))*RR,2RR);
kolo(z3,RR);

z4=(RR+((sqrt(3))/(3))*RR,2RR);
kolo(z4,RR);

x=(2/(sqrt(3)))*RR;
z5=(RR+((sqrt(3))/3)*RR-x,2RR);
z6=(RR+x,3RR);
z7=(RR+x,RR);
draw z1--z5--z2--z6--z3--z7--z1;
draw z5--z3;
draw z6--z1;
draw z2--z7;

z50=(RR,0) rotated 135;
z51=(RR,0) rotated 45;
z52=(z5+z2+z4)/3;
label.ulft("A",(x2+x50,y2+y50));
label.urt("B",(x3+x51,y3+y51));
label.llft("C",(x1-x51,y1-x51));
label("D",(x52,y52-1u));
endfig;

end
```

9. Zahradní sprcha – zakreslení sil působících na molekuly vody. Zajímavé je, že tento obrázek je kompletně zkonstruován, tzn. jsou zadány přesně polohy pouze několika málo výchozích bodů, ostatní jsou dopočítány. Změnou několika málo parametrů můžeme měnit parametry kresby (úhly,...)



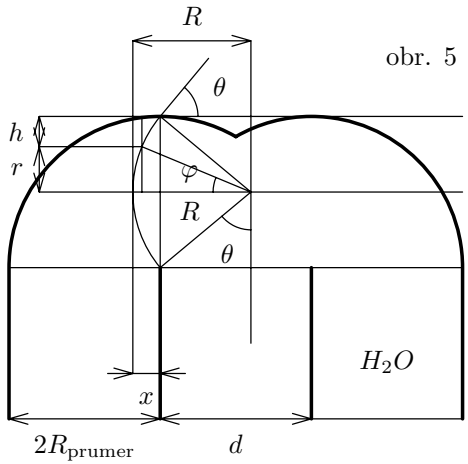
obr. 3

```
prologues:=1;

input makra

u=1cm;
```

```
l=0.5mm;
t=0.2mm;

beginfig(1)
  w:=7u;
  h:=4u;
  z1=(w/2,1/10h);
  z2=(7/8w,y1);
  z3=(x2,2/5h);
  z4=(2/3[x1,x2],y3);
  z5=(x4-length(z4-z1),y4);
  z6=(x5,19/20h);
  z7=(x6+1/4u,y6);
  z8=(x5,3/5[y5,y6]);
  x9=x7; z9=z8+whatever*(z8-z4) rotated 90;
  z10=1/5[z8,z4];
  x12=x5; z12=z10+whatever*(z9-z8);
  z11=whatever[z8,z9]=z12+whatever*(z8-z4);
  z13=whatever[z4,z8]=z6+whatever*(z9-z8);
  z14=(3/5x8,y8);
  z15=(x14,y3);
  z16=(x14,y6);
  z20=z1+2/5(z4-z1)rotated -90;
  z21=(-3/2)[z1,z20];
  z22=3/2[z4,z8];
  delobl:=u/2;
  z51=z4+delobl*dir(angle(z8-z4));
  z52=(x4-delobl,y4);
  z53=(x4-7/6delobl,y4);
  z54=z4+7/6delobl*dir(angle(z1-z4));
  z55=(x1+delobl,y1);
  z56=z1+delobl*dir(angle(z20-z1));
  pickup pencircle scaled t;
  draw (0,y1)--(w,y1);
  draw (0,y3)--(w,y3);
  draw (0,y6)--(w,y6);
  draw z2--z3;
  draw z1--z4--z22;
  draw z6--z5..{z20-z1}z1;
  draw z21--z20;
  draw z7--(x7,y5);
  draw z6--z13;
  draw z15--z16; draw z8--z14;
  draw z11--z12--z10;
  delsip:=u/4;
  draw sipka(delsip,90) shifted z16;
  draw sipka(delsip,-90) shifted z14;
  draw sipka(delsip,90) shifted z14;
  draw sipka(delsip,-90) shifted z15;
  draw sipka(delsip,90) shifted z3;
  draw sipka(delsip,-90) shifted z2;
```

```
  draw sipka(delsip,-90) shifted z2;
  draw sipka(delsip,0) shifted z6;
  draw sipka(delsip,180) shifted z7;
  oblouk(z4,delobl,angle(z8-z4),180);
  oblouk(z4,7/6delobl,180,angle(z1-z4)+360);
  oblouk(z1,delobl,angle(z20-z1),0);
  pickup pencircle scaled l;
  draw z9--z11;
  draw z10--z8--z12;
  draw sipka(delsip,angle(z10-z8))
shifted z10;
  draw sipka(delsip,angle(z12-z8))
shifted z12;
  draw sipka(delsip,angle(z11-z8))
shifted z11;

  label.rt(btex $R$ etex,(x2+l,1/2[y2,y3]));
  label.lrt(btex $S$ etex,(x4+l,y4-l));
  label.lrt(btex $\theta$ etex,
   (x55,1/2[y55,y56]));
  label.ulft(btex $\varphi$ etex,
   (x52,1/2[y51,y52]));
  label.lrt(btex $r_2$ etex,
   (1/2[x1,x4],1/2[y1,y4]));
  label.llft(btex $90^\circ-\theta$ etex,
   (x53,1/2[y53,y54]));
  label.llft(btex $F_g$ etex,(x12-l,y12));
  label.urt(btex $F$ etex,(x10+l,y10));
  label.lft(btex $r$ etex,
   (x14-l,1/2[y14,y15]));
  label.lft(btex $\hat h$ etex,
   (x14-l,1/2[y14,y16]));
  label.top(btex $\d x$ etex,
   (1/2[x6,x7],y7+3l));
  label.lrt(btex $\d l$ etex,(x9+2l,y9-l));
  label.urt(btex obr. 3 etex,(0,0));
endfig;

end
```

10. Zde je vykreslena výtoková trubice a geometrické vztahy v zahradní sprše. Také zde většinu kódu zabírá definice, jakou šipku/úhel nakreslit, samotný výpočet pozic bodů probíhá pouze na začátku zdrojového kódu. Kód by šel ještě zkrátit vyrobením maker pro



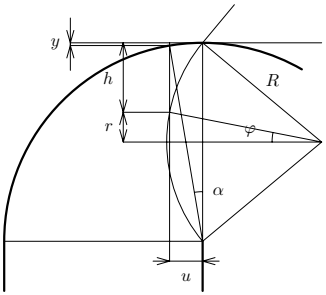
obr. 5

kreslení šipek,...

```
prologues:=1;

input makra

u=1cm;
l=0.5mm;
t=0.2mm;

delobl:=u/2;
delsip:=u/4;

beginfig(1)
  w:=6u;
  h:=5u;
  z1=(0,0); z2=(w/3,y1);
  z3=(w-x2,y2); z4=(w-x1,y1);
  z5=(x2,x2-x1); z6=(w-x5,y5);
  z8=(x5,y5+x5-x1); z9=(3/5[x5,x6],1/2[y5,y8]);
  z10=(x1,y5); z11=(w-x10,y10);
  path p[];
  p1=z10{up}..z8{right}..{down}z6;
  p2=p1 shifted (x5,0);
  z7=p1 intersectionpoint(p2);
  pickup pencircle scaled l;
  draw z2--z5; draw z3--z6;
  draw z1--z10..z8{right}
    ..{dir(angle(z7-z5)-90)}z7;
  draw z4--z11..(w-x8,y8){left}
    ..{dir(angle(z7-z6)+90)}z7;

  z12=(x9-length(z9-z5),y9);
  z13=(x12,1/5[y2,y12]);
  z14=(x12,h); z15=(x9,y14);
  z16=(1/5[x10,x5],y12); z17=(x16,y8);
  z18=3/5[z16,z17];
  z21=z8+u*dir(angle(z8-z9)-90);
  p3=z5{dir(angle(z9-z5)+90)}..z12{up}
    ..z8--z21;
  z19=(z18--(w,y18))intersectionpoint p3;
  z20=(x15,1/2[y2,y5]);
  pickup pencircle scaled t;
  draw p3; draw z5--z8; draw z5--z9--z8;
  draw z10--z11;
  draw z13--z14--z15--z20;
  draw (x13-u/3,y13)--(x2+u/3,y13);
  draw (w,y16)--z16--z17--(w,y17);
  draw z18--z19--z9;
  draw (x19,y12)--(x19,y8);
  draw sipka(delsip,180) shifted (x2,y13);
  draw sipka(delsip,0) shifted z13;
```

```
  draw sipka(delsip,0) shifted z15;
  draw sipka(delsip,180) shifted z14;
  draw sipka(delsip,90) shifted z17;
  draw sipka(delsip,-90) shifted z16;
  draw sipka(delsip,90) shifted z18;
  draw sipka(delsip,-90) shifted z18;
  oblouk(z8,delobl,0,angle(z21-z8));
  oblouk(z9,delobl,angle(z5-z9),-90);
  oblouk(z9,delobl,angle(z19-z9),180);
  draw z1--z4;
  draw sipka(delsip,0) shifted z2;
  draw sipka(delsip,180) shifted z1;
  draw sipka(delsip,0) shifted z3;
  draw sipka(delsip,180) shifted z2;

  label.bot(btex $2R_{\rm prumer}$ etex,
   (1/2[x1,x2],y1-2l));
  label.bot(btex $d$ etex,(1/2[x2,x3],y1-2l));
  label.bot(btex $H_2O$ etex,(1/2[x3,x4],y1+u));
  label.bot(btex $x$ etex,(1/2[x13,x2],y13-3l));
  label.llft(btex $\theta$ etex,
   (x9-3l,y9-5/4delobl));
  label.ulft(btex $\varphi$ etex,
   (x9-5/4delobl,y9+l));
  label.urt(btex $\theta$ etex,
   (x8+5/4delobl,y8+4l));
  label.lft(btex $r$ etex,(x16-2l,1/2[y16,y18]));
  label.lft(btex $h$ etex,(x16-2l,1/2[y17,y18]));
  label.top(btex $R$ etex,(1/2[x14,x15],y14+2l));
  label.llft(btex obr. 5 etex,(w,h));
  label.ulft(btex $R$ etex,
   (1/2[x5,x9],1/2[y5,y9]+l));
endfig;

end
```

11. Na tomto detailu je vidět důležitost přesného výpočtu průsečíků. Kdybychom obrázek nepočítali, ale kreslili od ruky, pak sebemenší změna pozic vstupních bodů nám velmi zkreslí přesné detaily.



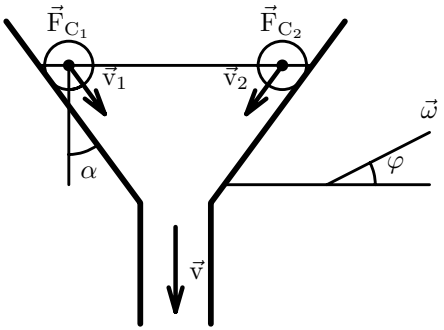
obr. 6
(detail obrázku 5)

```
prologues:=1;

input makra

u=1cm;
l=0.5mm;
t=0.2mm;
```

```
delobl:=u/2;
delsip:=u/4;

beginfig(1)
  w:=6u;
  h:=5u;
  z1=(0,0); z2=(2/3w,y1);
  z3=(2w-x2,y2); z4=(2w-x1,y1);
  z5=(x2,u); z6=(2w-x5,y5);
  z8=(x5,y5+x5-x1); z9=(3/5[x5,x6],1/2[y5,y8]);
  z10=(x1,y5); z11=(2w-x10,y10);
  path p[];
  p1=z10{up}..z8{right}..{down}z6;
  p2=p1 shifted (x5,0);
  z7=p1 intersectionpoint(p2);
  pickup pencircle scaled l;
  draw z2--z5;
  draw z1--z10..z8{right}
    ..{dir(angle(z7-z5)-90)}z7;

  z12=(x9-length(z9-z5),y9);
  z13=(x12,1/5[y2,y12]);
  z14=(x12,h); z15=(x9,y14);
  z16=(3/5[x10,x5],y12); z17=(x16,y8);
  z18=1.5/5[z16,z17];
  z21=z8+u*dir(angle(z8-z9)-90);
  p3=z5{dir(angle(z9-z5)+90)}
    ..z12{up}..z8--z21;
  z19=(z18--(w,y18))intersectionpoint p3;
  z20=(x15,1/2[y2,y5]);
  x13:=x19; x14:=x13;
  z50=(z13--z14) intersectionpoint p1;
  z51=(2/5[x1,x50],y50); z52=(x51,y17);
  pickup pencircle scaled t;
  draw p3; draw z5--z8; draw z5--z9--z8;
  draw z10--z5;
  draw (x13-u/3,y13)--(x2+u/3,y13);
  draw z9--z16--z17--(x9,y17);
  draw z13--z14; draw z18--z19--z9;
  draw z5--z50--z51--z52--z17;
  draw (x51,y51-u/2)--(x52,y52+u/2);
  draw sipka(delsip,180) shifted (x2,y13);
  draw sipka(delsip,0) shifted z13;
  draw sipka(delsip,90) shifted z17;
  draw sipka(delsip,-90) shifted z16;
  draw sipka(delsip,90) shifted z18;
  draw sipka(delsip,-90) shifted z18;
  draw sipka(delsip,90) shifted z51;
  draw sipka(delsip,-90) shifted z52;
  oblouk(z9,2delobl,angle(z19-z9),180);
```

```
  oblouk(z5,2delobl,90,angle(z50-z5));

  label.bot(btex $u$ etex,
   (1/2[x13,x2],y13-3l));
  label.ulft(btex $\varphi$ etex,
   (x9-5/2delobl,y9+l));
  label.lft(btex $r$ etex,(x16-2l,1/2[y16,y18]));
  label.lft(btex $h$ etex,(x16-2l,1/2[y17,y18]));
  label.urt(btex $R$ etex,
   (1/2[x8,x9],1/2[y8,y9]+l));
  label.lft(btex $y$ etex,(x51-2l,1/2[y51,y52]));
  label.rt(btex $\alpha$ etex,(x5+2l,y5+u));
  label.llft(btex obr. 6 etex,(0,h));
  label.llft(btex (detail obr\'azku 5) etex,
   (0,h-15pt));
endfig;

end.
```

12. Trychtýř. Pro jeho snažší vykreslení si nadefinujeme makro. Poněvadž je trychtýř symetrický podél svislé osy, vykreslíme jen jeho jednu půlku a pak ji překlopíme.



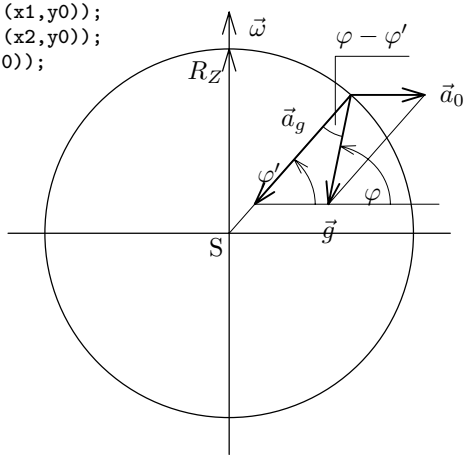
Obr. IV.1.6

```
prologues:=1;

input makra
input mkmakra

u:=0.8mm;

%                    __
% \  <---dn--->    / ^
%  \              /  |
%   \            /   |hn
%    \          /    |
%     \        /     |
%   ___\      /______V
%   ^  |     |
%   |  |     |
% ht|  |<dt>|
%   |  |     |
%   V__|     |
poc:=10u;
def trychtyr(expr poloha_pr_hor_rohu,
             dn,dt,hn,ht)=
 begingroup
  save x,y,p;
  path p;
  z1=(0,0);
  x2=x1+(dn-dt)/2;
```

```
  y2=y1-hn;
  z3=z2+ht*down;
  z4=z1+dn/2*right;
  z5=z3+dt/2*right;
  p:=(z1--z2--z3)
   shifted poloha_pr_hor_rohu;
  draw p;
  draw p reflectedabout (z4,z5);
 endgroup;
enddef;

beginfig(1)
 w:=70u;
 h:=60u;
 numeric v[];
 p:=u;
 delka:=4u;
 dn:=.8w;
 dt:=w/6;
 hn:=h/2;
 ht:=h-hn-poc;
 alfa:=angle((hn,(dn-dt)/2));
 phi:=27;
 r:=h/15;
 r_cast:=h/15;
 v0:=.7ht;
 v1=v2=h/6;
 z100=(dn/2,0);
 z101=(dn/2,h);
 z0=(0,h);
 z1=z0+3delka*dir(270+alfa)
  +r_cast*dir(alfa);
 z2=z1 reflectedabout(z100,z101);
 x3=x1;
 z3=z0+whatever*dir(270+alfa);
 z4=z0+whatever*dir(270+alfa);
 y4=y1;
 z5=z4 reflectedabout(z100,z101);
 z6=z5+whatever*dir(270-alfa);
 y6=y8=h-.9hn;
 x9=x8=w;
 z7=1/2[z6,z8];
 z9=z7+whatever*dir(phi);
 x11=x1;
 y11=h-.9hn;
 z10=(dn/2,.8ht+poc);
 pero(p);
 odot(z1,p);
 odot(z2,p);
 trychtyr(z0,dn,dt,hn,ht);
 pero(.75u);
```

```
 vektor(z10,v0,delka,270);
 vektor(z1,v1,delka,270+alfa);
 z21=z1+(v1,delka) rotated (270+alfa);
 vektor(z2,v2,delka,270-alfa);
 z22=z1+(v1,delka) rotated (270-alfa);
 pero(.5u);
 kruznice(z1,r_cast);
 kruznice(z2,r_cast);
 oblouk(z1,r,270,270+alfa);
 oblouk(z3,2r,270,270+alfa);
 oblouk(z7,2r,0,phi);
 draw z4--z5;
 draw z6--z8;
 draw z7--z9;
 draw z1--z11;

label.top(btex ${\rm\vec\omega}$ etex,
 (x9,y9+1u));
label.urt(btex $\varphi$ etex,(x7+9u,y7+1u));
label.rt(btex ${\rm\vec{v}}$ etex,
 (x10+1u,y10-v0/2));
label.urt(btex ${\rm\vec{v}_1}$ etex,
 (1/2[x1,x21],1/2[y1,y21]-2u));
label.ulft(btex ${\rm\vec{v}_2}$ etex,
 (2*x10-1/2[x1,x21],1/2[y1,y21]-2u));
label.lrt(btex $\alpha$ etex,(x3+1u,y3-9u));
label(btex ${\rm\vec{F}_{C_1}}$ etex,(x1,y0));
label(btex ${\rm\vec{F}_{C_2}}$ etex,(x2,y0));
label.top(btex Obr. IV.1.6 etex,(x10,0));
endfig;

end;
```

13. V METAPOSTu snadno vykreslíme čáry a vektory různých tlouštěk, stejně jako snadno vyznačíme úhly a jejich popisky.

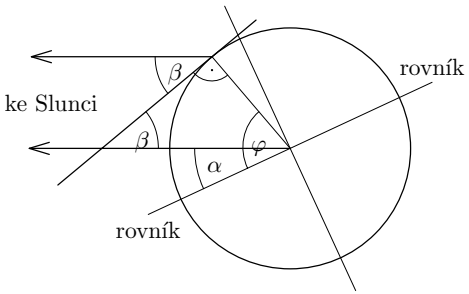



```
prologues:=1;

u=1mm;
v=1mm;

%def sipky
def sipka(expr cil,delka,smer)=
draw ((-1,0.3)--(0,0)--(-1,-0.3)) scaled delka rotated
smer shifted cil; enddef;

def kolo(expr stred,polomer)=
draw (fullcircle scaled 2polomer shifted stred);enddef;
```

```
beginfig(1);
w:=60u;
h:=60u;

pickup pencircle scaled 0.2v;
z100=(0,0);z101=(0,h);z102=(w,0);z103=(w,h);

RR=25u;
z1=(w/2,h/2);
path p[];
p1=fullcircle scaled 2RR shifted z1;
draw p1;


z2=(0,h/2);
z3=(w,h/2);
z4=(w/2,0);
z5=(w/2,h);
draw z2--z3;
draw z5--z4;
sipka(z5,3u,90);
z99=(w/2,h/2+RR);
sipka(z99,3u,90);

z6=point 1.08 of p1;
pickup pencircle scaled 0.15v;
draw z1--z6;

z7=0.21[z1,z6];
pickup pencircle scaled 0.3v;
draw z7--z6;
sipka(z7,3u,angle(z7-z6));

z8=z7+0.4RR*right;
draw z6--z8;
sipka(z8,3u,angle(z8-z6));

z9=z6+0.4RR*right;
draw z6--z9;
sipka(z9,3u,0);

z10=z8+0.6RR*right;
pickup pencircle scaled 0.15v;
draw z7--z8--z10; draw z8--z9;

pickup pencircle scaled 0.15v;
z21=0.82[z7,z8];
z22=0.41[z7,z6];
draw z22{dir(angle(z6-z7)-90)}
 ..{dir -90}z21;
sipka(z22,2u,angle(z6-z7)+85);
```

```
z23=0.53[z8,z6];
z24=0.56[z8,z10];
draw z23{dir(angle(z6-z8)-90)}
 ..{dir -90}z24;
sipka(z23,2u,angle(z6-z8)+87);


z31=0.29[z6,z7];
z32=0.38[z6,z8];
draw z31{dir(angle(z6-z7)-90)}
 ..{dir(angle(z6-z8)-90)}z32;

pickup pencircle scaled 0.1v;
z40=(w/2+14.4u,h/2+14.7u);
z41=(w/2+14.4u,h/2+24u);
z45=z41+0.4RR*right;
draw z40--z41;
draw z41--z45;

label.llft(btex S etex,(x1,y1));
label.bot(btex $\vec g$ etex,(x8,y8-1u));
label(btex $\varphi'$ etex,(x7+2u,y7+4u));
label(btex $\varphi$ etex,(x8+6u,y8+1u));
label.ulft(btex $\vec a_g$ etex,
 (2/5[x6,x7],2/5[y6,y7]));
label.top(btex $\varphi-\varphi'$ etex,
 (1/2[x41,x45],y41));
label.rt(btex $\vec a_0$ etex,(x9+1u,y9));
label.lrt(btex $\vec\omega$ etex,
 (x5+2u,y5));
label.llft(btex $R_Z$ etex,(x99,y99-1u));

endfig;

end
```

14. I na tomto obrázku je ihned zřejmé, který úhel je s kterým shodný.



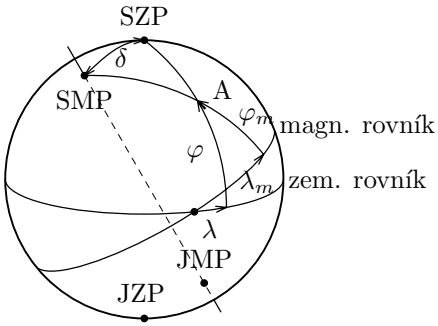

```
prologues:=1;

u=1mm;
v=1mm;

%def sipky
def sipka(expr cil,delka,smer)=
```

```
draw ((-1,0.3)--(0,0)--(-1,-0.3))
 scaled delka rotated smer shifted cil;
enddef;

def kolo(expr stred,polomer)=
draw (fullcircle scaled 2polomer
 shifted stred);
enddef;

beginfig(1)
w:=70u;
h:=60u;

pickup pencircle scaled 0.2v;
z100=(0,0);z101=(0,h);
z102=(w,0);z103=(w,h);

RR=18u;
z1=(3w/5,h/2);
path p[];
p1=fullcircle scaled 2RR shifted z1;
draw p1;


z2=point 2.9 of p1;
pickup pencircle scaled 0.15v;
draw z1--z2;

pickup pencircle scaled 0.2v;
z3=(0,h/2);
z6=z3+3u*right;
draw z1--z6;

z4=z2+27u*left;
draw z2--z4;

z10=0.67[z1,z3];
z11=1.4[z2,z10]; z12=1.4[z10,z2];
draw z11--z12;

pickup pencircle scaled 0.1v;
z61=0.4[z10,z2]; z62=0.3[z10,z1];
draw z62{dir 90}
 ..{dir(angle(z2-z10)+90)}z61;
z64=0.4[z2,z10]; z65=z2+8.5u*left;
draw z64{dir(angle(z2-z10)+90)}
 ..{dir 90}z65;
z68=0.17[z2,z10]; z69=0.2[z2,z1];
draw z68{dir(angle(z2-z10)-90)}
 ..{dir(angle(z1-z2)+90)}z69;
```

```
pickup pencircle scaled 0.2v;
sipka(z6,3u,180);
sipka(z4,3u,180);

pickup pencircle scaled 0.1v;
z34=point 2.55 of p1;
z35=2.3[z34,z1];
z36=1.3[z1,z34];
draw z35--z36;

z44=point 0.555 of p1;
z45=2.3[z44,z1];
z46=1.3[z1,z44];
draw z45--z46;

z50=0.61[z1,z45];
z51=z1+1.3*0.61*RR*left;
draw z50{dir(angle(z1-z45)+90)}
 ..{dir 90}z51;

z52=0.3[z1,z45];
z53=0.3999999[z1,z2];
draw z52{dir(angle(z1-z45)+90)}
 ..{dir(angle(z1-z2)+90)}z53;

pickup pencircle scaled 1/3v;
drawdot z2+(0,-2u);

label.bot(btex $\beta$ etex,(x65+3u,y65));
label(btex $\beta$ etex,(x10+6u,y10+1u));
label.bot(btex $\alpha$ etex,
 (x51+3u,y51-1u));
label(btex $\varphi$ etex,(x53,y1));
label.bot(btex rovn\'\i k etex,(x45,y45));
label.top(btex rovn\'\i k etex,(x46,y46));
label.rt(btex ke Slunci etex,
 (x4-5u,(y4+y6)/2));
endfig;

end
```

15. Někdy se nevyhneme ani složitým definicím. Sférická trigonometrie je složitá, takže jsem použil velice hrubé aproximace. Obvodové kružnice jsou ručně odhadnuty.

```
prologues:=1;
```



Obr. IV.1.8

```
input makra

u:=0.46mm;
p:=.6u;

def oblouk_elipsy(expr stred,polomer_x,
 splosteni,sklon,poc_uhel,konc_uhel)=
 ((dir(poc_uhel){dir(poc_uhel+90)}
 ..dir((konc_uhel+poc_uhel)/2)
 ..{dir(konc_uhel+90)}dir(konc_uhel))
  xscaled polomer_x
  yscaled (polomer_x*splosteni)
  rotated sklon shifted stred);
enddef;

beginfig(1)
  w:=100u;
  h:=100u;
 pair SZP,JZP,SMP,JMP,JMP',
      SMP'',JMP'',A;
 delka:=3u;
 path P[];
 s:=w/100;
 v:=h/100;
 const:=1/4;
 r:=40s;
 hust:=0.03;
 odklon:=30;
 sklon_mag:=8; %maly
 z0=(60s,50v);
 SZP=z0+r*up;
 JZP=SZP rotatedabout(z0,180);
 SMP=z0+r*(up scaled (1-sind(sklon_mag))
  rotated (odklon));
 JMP=SMP rotatedabout(z0,180);
 JMP'=z0+r*dir(angle(JMP-SMP));
 SMP''=z0+1.1r*dir(angle(SMP-JMP));
 JMP''=SMP'' rotatedabout(z0,180);
 pero(.75p);
 P1=oblouk_elipsy(z0,r,const,0,180,360);
 P2=oblouk_elipsy(z0+r*sind(sklon_mag)
  *dir(270+odklon),r*cosd(sklon_mag),
  const,odklon,180+sklon_mag,360-sklon_mag);
 draw P1;
 draw P2;
 draw SMP--SMP'';
 draw JMP'--JMP'';
 z1=point 1.4 of P1;
 z2=point 1.7 of P2;
 P3=SZP..{down}z1;
 draw P3;
```

```
A=point .42 of P3;
draw z2{dir(90+sklon_mag+30)}..A..SMP;
draw SMP{dir(sklon_mag+35)}..{right}SZP;
draw sipka(delka,10) shifted z1;
draw sipka(delka,48) shifted z2;
draw sipka(delka,115) shifted A;
draw sipka(delka,153) shifted A;
draw sipka(delka,3) shifted SZP;
draw sipka(delka,215+sklon_mag) shifted SMP;
pero(.5p);
draw SMP--JMP--JMP' dashed evenly
 scaled 0.3mm;
z10=(P1)intersectionpoint(P2);
pero(p);
draw (fullcircle scaled 2r shifted z0);
q:=2p;
pero(.9q);
odot(SZP,q);
odot(SMP,q);
odot(JZP,q);
odot(JMP,q);
odot(z10,q);

label(btex Obr. IV.1.8 etex,(w/2,-2.8u));
label.llft(btex $\lambda$ etex,
 (x1-1u,y1-2u));
label.llft(btex $\lambda_m$ etex,
 (x2+5u,y2-3u));
label.rt(btex A etex,((xpart A)+2u,
 (ypart A)+3u));
label.bot(btex SMP etex,(xpart SMP,
 (ypart SMP)-2u));
label.top(btex SZP etex,(xpart SZP,
 (ypart SZP)+2u));
label(btex $\delta$ etex,
 (0.5*((xpart SZP)+(xpart SMP))+2u,
  0.5*((ypart SMP)+(ypart SMP))+5u));
label.top(btex JMP etex,
 ((xpart JMP),(ypart JMP)+2u));
label.top(btex JZP etex,
 ((xpart JZP)-1u,(ypart JZP)+2.5u));

label(btex magn. rovn\'\i k etex,
 (x2+27u,y2+8u));
label(btex zem. rovn\'\i k etex,
 (x2+27u,y2-7u));

label.lft(btex $\varphi$ etex,
 (((xpart A)+x1)/2,((ypart A)+y1)/2));
label.rt(btex $\varphi_m$ etex,
 (((xpart A)+x2)/2,((ypart A)+y2)/2+3u));
```

```
endfig;

end;
```

5. Souvislosti s METAFONTem

METAPOST není původní produkt. Již okolo roku 1977 vymyslel D. E. Knuth společně s typografickým systémem TEX také program na návrh fontů METAFONT. Program má v názvu slovo META, protože neslouží pouze k návrhu jednoho fontu. Fonty se v něm popisují parametricky (jen CM fonty jsou závislé na asi 56 parametrech). Mnoho řezů daného fontu je pak možno vygenerovat ze stejných zdrojových textů pouhou editací několika málo parametrů. Kdo by věřil, že např. `typewriter` a `roman` fonty jsou generovány stejným programem.

Ačkoliv byl METAFONT určen pro generování fontů, jeho možnosti byly natolik široké, že si získal oblibu i jako editor vysoce kvalitních technických ilustrací. Bohužel program má dosud několik závažných omezení, které se musejí ošklivě obcházet (maximální velikost obrázku, bitmapový výstup pouze v černé a bílé barvě, nesnadné sázení popisků,...). Nejzávažnější je asi jeho bitmapovost. U fontů to je možná vhodné, protože požadujeme rychlé zpracování při mnohonásobném použití. U obrázku, který jednou použijeme, je tato vlastnost spíše na závadu. Kvůli obyčejné změně měřítka je nutno celý obrázek rekompilovat (přece ho nebudeme zvětšovat lineární interpolací!).

Proto se objevil METAPOST. Autorem METAPOSTu je John Hobby z Bellových laboratoří. Pokud jsme schopni vytisknout PostScriptový výsledek (což není problém), myslím si, že výhody tohoto programu vysoce převažují nad nevýhodami. Pro kreslení obrázků je METAPOST daleko pohodlnější.

Zajímavé je, že METAPOST vznikl z METAFONTu pouhou modifikací zdrojového textu (výpočetních a výstupních rutin). Zůstal zachován syntaktický parser, makrojazyk,...

METAPOST obsahuje mnoho rozšíření jazyka oproti METAFONTu. Dlužno podotknout, že některé prvky jazyka nebyly v METAPOSTu implementovány: např. veškeré bitmapové manipulace (protože PostScript je vektorový jazyk), popis desítek parametrů pro matematické fonty a kerningových tabulek (protože program není již určen pro generování fontů),...

6. Literatura

Nejaktuálnější informace naleznete samozřejmě na homepage METAPOSTu. Její adresa je `http://cm.bell-labs.com/who/hobby/MetaPost.html`. Nejlepší (a

pravděpodobně jedinou) učebnicí je METAPOST User Manual např. na adrese `http://www.cstug.cz/documentation/index.html`, kde najdete také spoustu jiné zajímavé dokumentace.

Robert Špalek

TUGboat 18(1), March 1997

TUGboat 18(2), June 1997

TUGboat 18(3), September 1997

TUGboat 18(4), December 1997

TUGboat 19(1), March 1998

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	710
Uzávěrka:	25. prosince 1998
Odpovědný redaktor:	Zdeněk Wagner
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. 05-740233
Adresa:	CSTUG, c/o FI MU, Botanická 68a, 602 00 Brno
fax:	05–412 125 68
e-mail:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
korespondence předsedovi sdružení

`cstug-members@cstug.cz`
korespondence členům sdružení

`cstug-faq@cstug.cz`
řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`secretary@cstug.cz`, `orders@cstug.cz`
korespondence administrativní síle sdružení, objednávky CD-ROM

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

ftp server sdružení:
`ftp://ftp.cstug.cz/`

www server sdružení:
`http://www.cstug.cz/`

Podávání novinových zásilek povoleno Českou poštou, s. p. OZJM Ředitelství v Brně č. j. P/2–1183/97 ze dne 11. 3. 1997.