Zpravodaj Československého sdružení uživatelů TeXu

Robert Špalek

METAPOST

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 8 (1998), No. 3-4, 175–218

Persistent URL: http://dml.cz/dmlcz/149829

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.

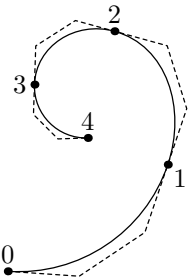
4. Ukázky

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

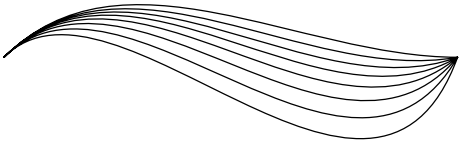
V této kapitole se nacházejí původní obrázky z METAPOST User Manual. Není na nich nic změněno, vylepšeno, či přidáno. Ukázky jsou cíleně vybrány pro demonstraci schopností METAPOSTu. Děkuji tímto autorovi METAPOSTu (*John Hobby*) za svolení k přetisknutí obrázků.

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



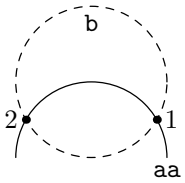

```
prologues:=1;

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
```

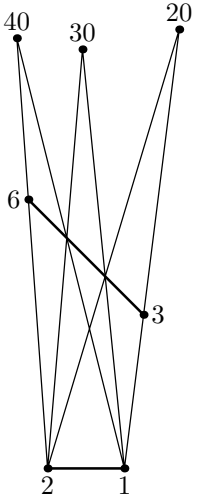
3. Demonstrace výpočtu průsečíků 2 kružnic, kreslení čárkovaných křivek.




```
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);
```

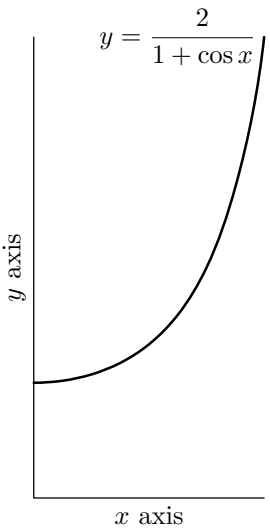
```
label.bot(btex \tt aa etex, point 0 of aa);
label.bot(btex \tt b etex, point 2 of b);
endfig;
end
```

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
```

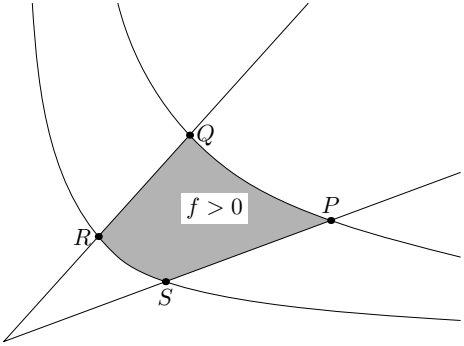
5. Matematickou funkci vykreslíme `for` cyklem, popisky k ní vysázíme TEXem. Na vysázené popisky můžeme aplikovat PostScriptovou transformaci, např. rotaci o 90°.




```
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(
```

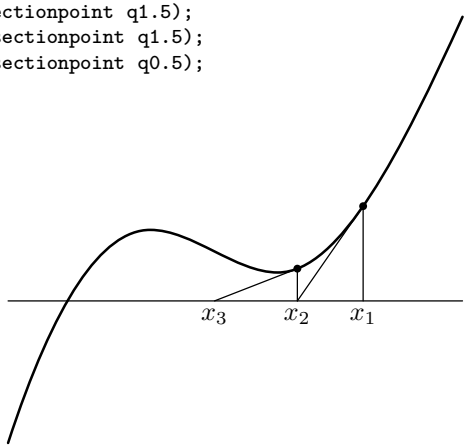
```
  btex $\displaystyle y={2\over1+\cos x}$ etex,
  (120ux, 4uy));
endfig;
end
```

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.

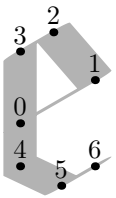


```
prologues:=1;

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
```

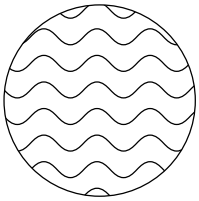
9. Při návrhu reklamní grafiky, firemních log a pro tvorbu výrazných efektů oceníme kaligrafické pero, čímž může být např. šikmo seříznutý husí brk, který držíme po dobu kreslení pod daným úhlem. Pokud bychom chtěli úhel a poloměr pera po dobu kreslení měnit, i tady nám METAPOST pomůže. Makrem `penstroke`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



Zde je demonstrováno obyčejné kreslení křivek kaligrafickým perem.

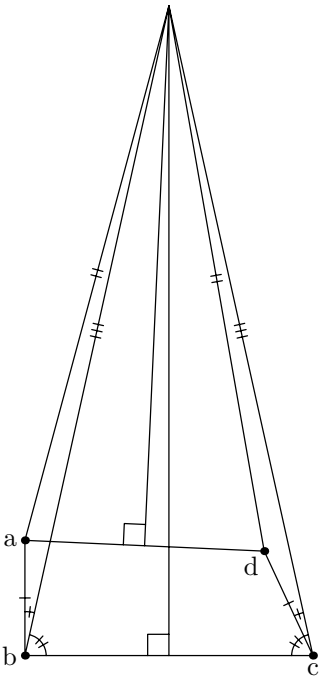
```
prologues:=1;

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.

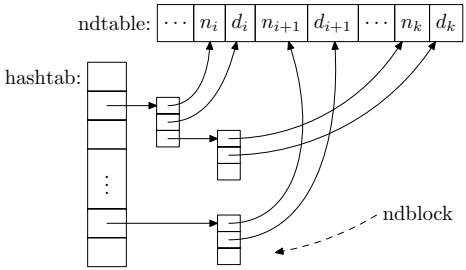



```
prologues:=1;

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
```

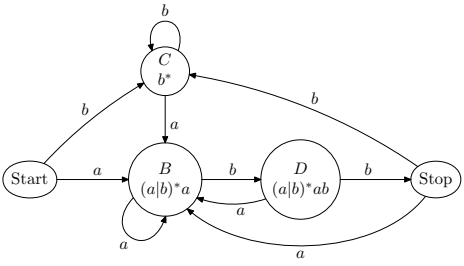
12. Při sazbě technických článků je často potřeba vykreslit komplikovaný graf nejrůznějších závislostí plný rámečků, šipek a popisků. Pomocí maker ze souboru `boxes.mp` popíšeme souřadnice a METAPOST nám pomůže nakreslit co nejpřehlednější šipky.




```
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.



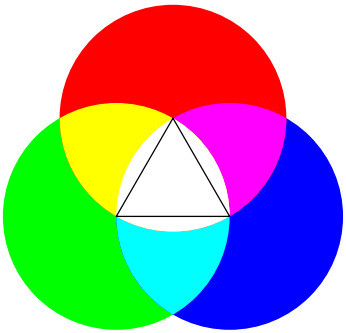

```
prologues:=1;

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.

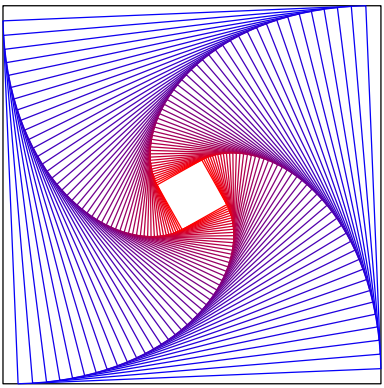
1. Aditivní mísení barev nakreslíme postupným překrýváním oblastí. Nejprve vyplníme do kruhů 3 základní barvy, pak vykreslíme po dvou jejich průniky a výsledek překryjeme bílým středem s vyznačeným trojúhelníkem.



```
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…

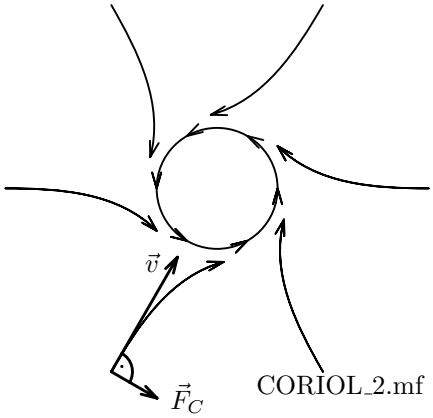

```
prologues:=1;

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.

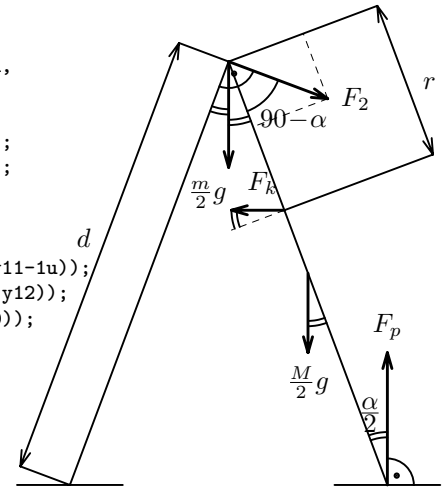
1. Rotačně symetrický obrázek plný zakřivených vektorů nakreslíme např. takto: nadefinujeme si makro, které vykreslí jednu z šipek, a pak `for` cyklem vyvoláme makro podél obvodu kruhu.



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.

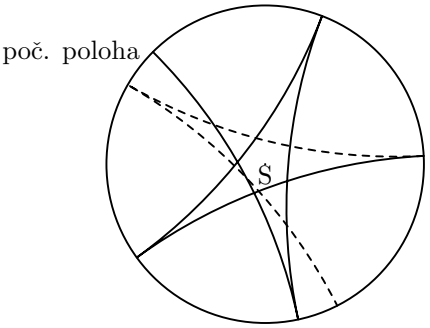
```
verbatimtex \def\ds{\displaystyle}
\def\Over{\over\ds}etex;
prologues:=1;

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.

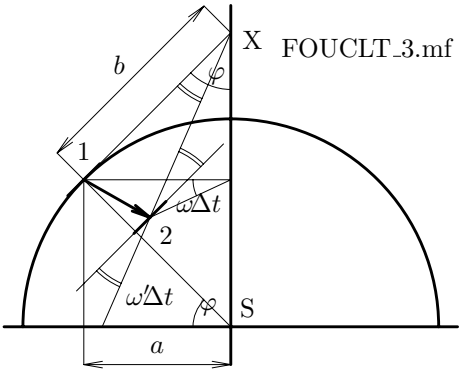
```
prologues:=1;
```

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
```

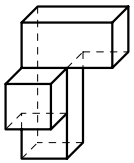
4. Detail působení sil na Foucaltovo kyvadlo. Práce s kreslením úhlů mezi body je ulehčena makrem `dejuhel`, které převezme trojici bodů a poloměr úhlu a samo vykreslí křivku na správném místě.




```
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;
```

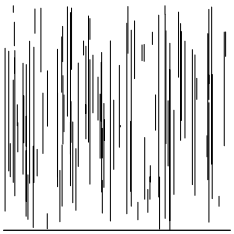
```
draw z12--z13;
dejuhel(z6,z8,z11,delkauhlu,+1);
dejuhel(z13,z11,z10,2*delkauhlu,+1);
dejuhel(z12,z11,z9,2*delkauhlu,+1);
dejuhel(z13,z11,z10,1.9*delkauhlu,+1);
dejuhel(z12,z11,z9,1.9*delkauhlu,+1);

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.

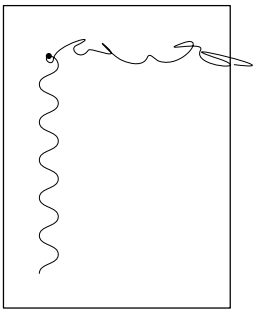


zvětšený papír

```
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.

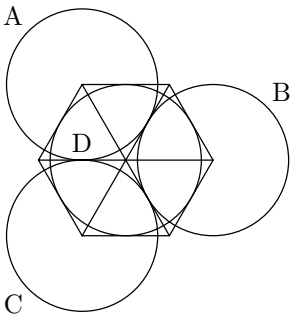
jehla



```
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
```

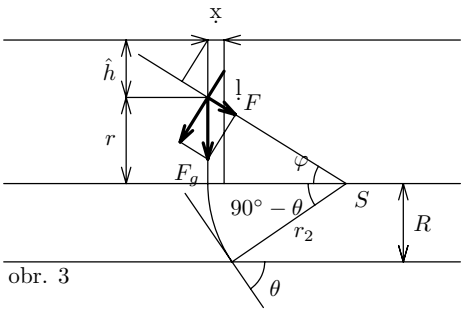
8. Geometrické uspořádání kuliček hrachu nasypaných pravidelně do krabice. Kód by šel vylepšit využitím symetrie, ale ručně spočítat souřadnice funguje taky.




```
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,...)



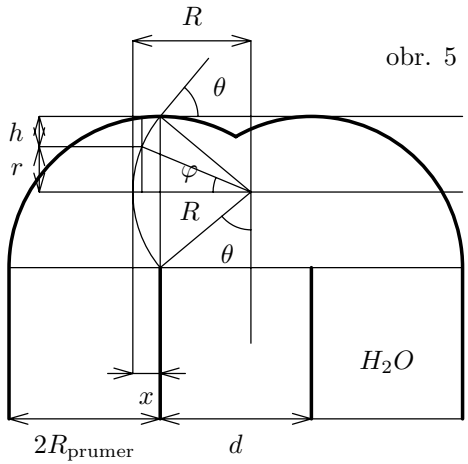
obr. 3

```
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



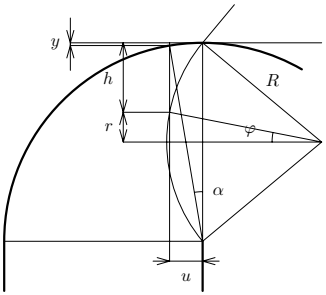
obr. 5

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.



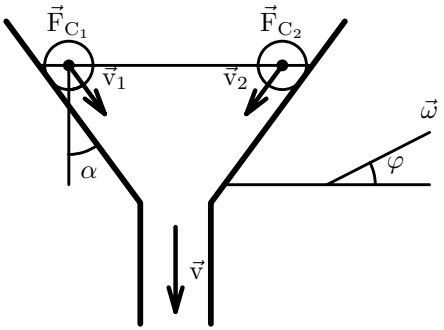
obr. 6
(detail obrázku 5)

```
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.



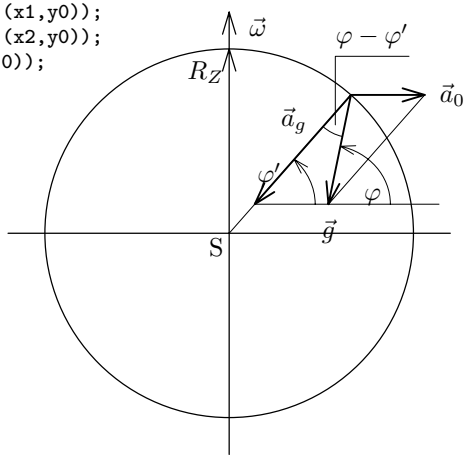
Obr. IV.1.6

```
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.

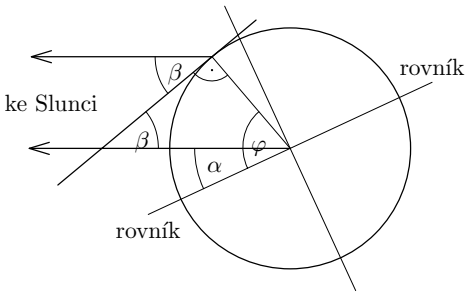



```
prologues:=1;

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.



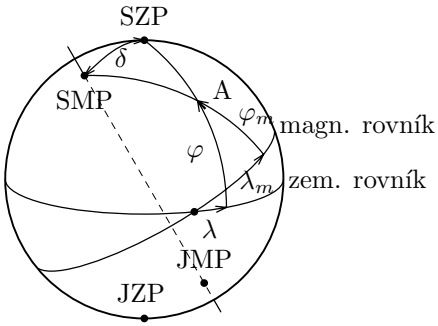

```
prologues:=1;

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