M E T A F O N T

METAFONT je program pro vytváření znaků nebo obrázků, které TEX sází na stránku.

Instalace METAFONTu

Na disku je nutné mít vlastní METAFONT `mf.exe`, případně jeho obdoby `mf186.exe`, `mf386.exe` nebo `sbmf.exe`. Dále je třeba vlastnit konvertor `gftopk.exe` a základní makro k METAFONTu `plain.bas`. Všechny důležité soubory jsou součástí programového balíku emTEX, ze kterého je také můžete instalovat.

Spouštění METAFONTu z menu

Příkazy důležité pro spuštění METAFONTu se v menu systému nacházejí pod heslem **METAFONT**.

- **Parameters..** Otevře se okénko, kde lze měnit parametry ovlivňující činnost METAFONTu a GFtoPK.

  `'%MF% file (without extension):'` Název METAFONTového souboru bez přípony.

  `'METAFONT'` Parametry předávané programu METAFONT. Nejčastěji používané podoba parametrů je `&plain \mode=laserjet; mag=1.0; input %MF%`. Při úpravách je vhodné měnit pouze `laserjet`, což určuje tiskárnu, pro kterou se generuje font, a `mag=1.0`, což znamená zvětšení oproti základní velikosti. Používané názvy pro tiskárny jsou `laserjet` (laserové a tryskové tiskárny v rozlišení 300dpi), `epsonfx` (9 jehličkové tiskárny v rozlišení 240dpi) nebo `lqlores` (24 jehličkové tiskárny v rozlišení 180dpi). Další lze nalézt v souboru `modes.mfj`.

  `'Convertor'` Výstup METAFONTu je vždy nutno zkonvertovat pomocí konvertoru GFtoPK, jež se volá nejčastěji jako `gftopk %MF%.300 %MF%.pk`. Mění se pouze přípona souboru (`300`), která označuje hustotu bodů v dpi u vygenerovaného fontu. Hustota je ovlivňována typem tiskárny a zvětšením.
- **Edit (mf)** Editování METAFONTového souboru, ve kterém lze vektorově popisovat znaky nebo obrázky.
- **Metafont** Spuštění METAFONTu, kontrola syntaxe. Vzniknou soubory s příponami `300gf`, `tfm` a `log`.
- **Convertor** Spuštění konvertoru GFtoPK. Program zkonvertuje soubor `300gf` do souboru `pk`, kterému je TEX schopen porozumět.
- **Edit (mp)** Editace souboru s příponou `mp`, ve kterém lze psát příkazy pro TEX. Nejčastěji se zde zkouší sazba vygenerovaného fontu na stránku.
- **Tex (mp)** Překlad souboru s příponou `mp`.
- **View (mp)** Prohlížení výsledku po překladu souboru s příponou `mp`.

Umístění obrázku do dokumentu

Potřebné obrázky jsme vytvořili v METAFONTu např. v souboru `obrazky`. Nový font (soubory s příponou `pk` a `tfm`) zavedeme (obvykle v záhlaví dokumentu) příkazem `\font\obr=obrazky`. Pokud je zvětšení dokumentu jiné než velikost fontu, je nutno použít např. `\font\obr scaled 1440`. V textu pak stačí napsat např. `{\obr\char1}` a TEX vysází znak s číslem 1 v ASCII z fontu `obrazky`. Umístění znaku v TEXu umožňuje makro `\unitlength=1mm \put(10,20){tex` které vloží `text` 10 mm vpravo a 20 mm nahoru od levého dolního rohu znaku, který se vysází.

Generování chybějícího fontu

Nenajde-li ovladač při prohlížení nebo tisku font, oznámí uživateli následující:

```
Loading font csfib10 scaled 1440

Warning 1205: font csfib10 [csfib10<432>] not found
Enter new font name/size:
```

Program nás vybízí, abychom vložili jméno jiného fontu a jeho rozlišení (oddělovačem je mezera). To je pouze jednorázové řešení a pokud nemáme konfigurovaný program MFJOB, který dokáže spustit METAFONT, musíme font vygenerovat sami.
V menu METAFONT/Parameters zapíšeme jméno fontu (csfib10), typ tiskárny (laserjet), zvětšení (1.44) a příponu (432). Potom spustíme METAFONT a konvertor.
Soubory `pk` se obvykle nacházejí v adresáři `\EMTEX\FONTS\PKzzz\DPIxxx`, kde `xxx` jsou číslice označující rozlišení fontu v DPI a `zzz` odpovídá rozlišovací schopnosti tiskárny. Soubor `tfm` se již na disku nejspíše vyskytuje v adresáři `\EMTEX\TFM`. Tvar tohoto souboru totiž nezávisí na rozlišení. Soubory s příponami `log` a `xxx` můžeme smazat.

Skupiny příkazů METAFONTu

Operace s čísly: $+$, $-$, $*$, $/$, $**$, $++$, $+-+$, abs, cosd, dir, sind, sqrt
Předdefinované hodnoty: down, left, origin, right, up, whatever
Kreslení: draw, drawdot, fill, filldraw, erase, pickup
Cesty: $--$, $---$, .., …, controls, cycle, fullcircle, unitsquare, tension
Deklarace: boolean, numeric, pair, path, pen, picture, transform

Operace s cestami: &, direction, directiontime, intersectionpoint, intersectiontimes, length, reverse, subpath
Operace s body (vektory): +, −, ∗, abs, angle, dotprod, length, xpart, ypart
Transformace: identity, reflectedabout, rotated, rotatedaround, scaled, shifted, slanted, xscaled, yscaled

Index METAFONT**u**

`em#:=1/3pt#;` jednotka `em` nezávisí na rozlišení
& `draw p1 & p2;` volné navázání cest `p1`, `p2`
“ ” `s="";` prázdný řetězec
[ ] `z3=.3[z1,z2];` lineární kombinace bodů, bod `z3` leží v 1/3 mezi body `z1`, `z2`
+ `c=a+b;` sčítání
++ `c=a++b;` $\sim \sqrt{a*a+b*b}$ Pythagorejské sčítání
+ − + `c=a+-+b;` $\sim \sqrt{a*a-b*b}$ Pythagorejské odečítání
− `c=a-b;` odečítání
−− `z1--z2;` rovná čára
− − − $\sim$ `..tension infity..` křivka s maximálním napětím, hladký přechod úsečky na křivku
* `c=a*b;` násobení
** `c=a**b;` umocňování
/ `c=a/b;` dělení
:= `a:=b;` přiřazení
= `a=b;` porovnání, lineární rovnice
.. `z1{dir40}..z2..z3;` volná křivka, z bodu `z1` vychází pod úhlem 40°
... $\sim$ `..tension atleast 1..` křivka s napětím větším než 1
abs `abs z;` absolutní hodnota vektoru
addto `addto currentpicture also V;` k aktuálnímu obrázku přičte obrázek `V`
also viz addto
and logický součin
angle `angle(z1-z2);` úhel vektoru v intervalu $(-180°, 180°)$
beginchar `beginchar("A",11pt#,11pt#,0);` začátek definice znaku, umístění v ASCII (`"A"` nebo `65`), šířka, výška nad účařím, hloubka pod účařím (uloží se do proměnných `w`, `h`, `d`)
begingroup `begingroup` začátek skupiny
boolean `boolean b;` logická proměnná
bot `bot z1=(0,0);` zarovnání na dolní okraj podle tloušťky pera
ceiling `ceiling x;` zaokrouhlování nahoru
clearit `clearit;` $\sim$ `currentpicture:=nullpicture;` vymazání obrázku
clear_pen_memory `clear_pen_memory`; smazání dříve uchovaných per
controls `draw z1..controls z3 and z4..z2;` křivka s pomocnými body `z3`, `z4`
cosd `cosd40;` $\cos 40°$
cull $\sim$ `cullit`, viz `keeping`
cullit `cullit;` záporné hodnoty bodu nahradí 0, kladné hodnoty 1
currentpicture `V:=currentpicture;` proměnná uchovávající aktuální obrázek
cutdraw `cutdraw p;` $\sim$ `draw p; cutoff(point 0 of p, 180+angle direction 0 of p); cutoff(point infinity of p, angle direction infinity of p);`
cutoff `cutoff(z,Φ);` v bodě `z` smaže polovinu bodů pera `currentpen` se směrem v intervalu $(\Phi - 90°, \Phi + 90°)$
cycle `fill z1..z2..z3..cycle;` uzavření cesty, návrat do prvního bodu
def `def arrow(expr x,y,uhel)=` začátek definice makra
define_pixels `define_pixels(u);` přepočet jednotek na pixely podle parametru `\mode=`, `u:=u#*hppp`, kde `hppp` je počet pixelů na bod v horizontálním směru (`define_blacker_pixels(u);` a další viz soubor `plain.mf`
dir `dir40;` jednotkový vektor svírající úhel 40° s osou x;
direction `direction t of p;` směr cesty `p` v čase `t`
directiontime `directiontime z of p;` čas `t`, pro který má tečna v bodě křivky směr vektoru `z`
dotprod `z1 dotprod z2;` skalární součin
down `z1:=down;` bod $(0, -1)$
draw `draw p;` nakreslení cesty `p`
drawdot `drawdot z`k`;` vykreslení tečky
end `end;` konec souboru, stisknout ⟨Enter⟩
endchar `endchar;` konec definice znaku
enddef `enddef;` konec definice
endfor `endfor` konec for cyklu
endgroup `endgroup;` konec skupiny
epsilon $epsilon = \frac{1}{65536}$ nejmenší číslo větší než nula
erase `erase draw p;` $\sim$ `cullit; undraw p; cullit;`
exitif `exitif b=true;` podmínka pro ukončení for cyklu

expr `def(expr x)=` parametr libovolného typu
fill `fill p;` vyplnění oblasti ohraničené cestou `p`
filldraw `filldraw p;` ∼ `draw p; fill p;`
flex `flex(z1,z2,...,zn)` ∼ `z1..z2{zn-z1}..···..zn`
floor `floor x;` celá část `x`
fontdimen `fontdimen 3: 2.5,6.5,0,4x` v TEXu se tyto hodnoty volají příkazy `\fontdimen3` až `\fontdimen6`, v plainu je `\fontdimen1` až 7 rezervováno
for `for x=x1 upto x2: text(x); endfor;` nebo `for x=x1 step x2 until x3: text(x); endfor;` nebo `for k=1,2,3: text(x[k]); endfor;` cyklus for
forever `forever: text; exitif b=true; endfor;` cyklus repeat until
fullcircle `draw fullcircle;` kružnice o jednotkovém průměru, střed `(0,0)`
halfcircle `draw halfcircle;` půlkružnice o jednotkovém průměru, střed `(0,0)`
identity `t=identity;` identické zobrazení
if:elseif:else:fi `if` $boolean_1$`:`$text_1$ `elseif` $boolean_2$`:`$text_2$ `else:`$text_3$ `fi;` podmínka
infinity `z1=point infinity of p;` konečný bod cesty `p`;
input `input makra.mf;` vložení souboru `*.mf`
interim `interim x:=0.4;` po `endgroup` (součást `endchar`) bude mít `x` hodnotu jako před `interim`
intersectionpoint `(x,y):=p1 intersectionpoint p2;` bod `(x,y)`, kde se cesty `p1,p2` protínají
intersectiontimes `(t1,t2)=p1 intersectiontimes p2;` časy `t1,t2`, kdy se cesty `p1,p2` protínají
keeping `cull currentpicture keeping(2,4);` vymaže body, které nejsou nakresleny $2\times, 3\times$ nebo $4\times$
known `known x;` logický výraz, true pokud je `x` definováno
label `label(1,2,3);` popis bodů `z1,z2,z3` ve výstupu programu GFtoDVI (viz dodatek GFtoDVI)
left `draw z1{left}..z2;` bod $(-1, 0)$
length `length(z2-z1); length s; length p;` délka vektoru, počet znaků, počet bodů na cestě zmenšený o 1
lft `lft z1=(0,y);` zarovnání na levý okraj podle tloušťky pera
ligtable `ligtable "A":"V" kern -2pt;` kerning, ligatury, vyskytne-li se "V" po "A", posune se o `2pt` doleva
makepen `makepen p;` vytvoření pera podle cesty `p`
makelabel `makelabel(z1,"bod 1");` popis bodu `z1` ve výstupu programu GFtoDVI, automatické umístění, `makelabel.top.nodot();` umístění nahoře, bez tečky (viz dodatek GFtoDVI)
max `d:=max(a,b,c);` maximum z množiny
message `message "text";` výpis textu na obrazovku při překladu (např. spolu s `show`)
mexp `mexp x;` ∼ $e^{x/256}$
min `d:=min(a,b,c);` minimum z množiny
mlog `mlog x;` ∼ $256 \ln x$
mode `mode=proof;` nastavení módu (přehled viz soubor `local.mf`)
mode_def `mode_def laserjet=` definice módu
mode_setup `mode_setup;` nastaví hodnoty jednotek podle parametrů `\mode=`, `\mag=`
normaldeviate `x:=normaldeviate;` náhodné číslo s normálním rozdělením
nullpen `pickup nullpen;` pero neviditelný bod, `beginchar` inicializuje `currentpen:=nullpen`
numeric `numeric a[];` nepovinná deklarace proměnné typu číslo
or logický součet
origin `z1=origin;` bod `(0,0)`
pair `pair z;` nepovinná deklarace bodu
path `path p[];` deklarace proměnné typu cesta, řada bodů
pencircle `pickup pencircle;` pero jednotková kružnice
penlabels `penlabels(1,2);` popis bodů `z1,z2,z1r,z2r,z1l,z2l` ve výstupu programu GFtoDVI (viz dodatek GFtoDVI)
penpos `penpos1(a,b)` ∼ `z1=.5[z1l,z1r]; z1r=z1l+(a,0) rotated b;` deklarace `z1l`, `z1r`, viz `penstroke`
penrazor `pickup penrazor;` pero nejtenčí jednotková úsečka v ose x se středem v počátku
pensquare `pickup pensquare;` pero jednotkový čtverec
penstroke `penstroke z1e..z2e;` ∼ `fill z1l..z2l--z2r..z1r--cycle;` viz `penpos`
pickup `pickup pencircle xscaled 2pt yscaled 1pt rotated 30;` nastavení aktuálního pera
picture `picture V[];` deklarace proměnné typu obrázek
point `point t of p;` bod na cestě `p` v čase `t`
postcontrol `z1:=postcontrol t of p;` kontrolní bod následující po bodu s časem `t`
precontrol `z1:=precontrol t of p;` kontrolní bod předcházející bod s časem `t`
quartercircle `draw quartercircle;` čtvrtkružnice o jednotkovém průměru, střed `(0,0)`
randomseed `randomseed:=2;` posloupnost náhodných čísel bude reprodukovatelná
reflectedabout `(x,y) reflectedabout((a,b),(c,d));` osová souměrnost podle vektoru `(a,b)-(c,d)`
reverse `reverse p;` body cesty v opačném pořadí
right `draw z1{right}..z2;` bod $(1, 0)$

rotated `(x,y) rotated Φ;` $\sim (x\cos\Phi - y\sin\Phi, x\sin\Phi + y\cos\Phi)$ otočení okolo `(0,0)`
rotatedaround `(x,y) rotatedaround((a,b),Φ);` rotace `(x,y)` okolo `(a,b)` o úhel Φ
round `round x;` zaokrouhlování
rt `rt z1=(x,y);` zarovnání na pravý okraj podle tloušťky pera
sind `sind40;` $\sin 40°$
save `save x;` po `endgroup` bude mít `x` hodnotu jako před `begingroup`
savepen `p:=savepen;` uchování aktuálního pera
scaled `(x,y) scaled a;` $\sim (ax, ay)$ násobení v obou složkách
screenchars `screenchars; showit` po `endchar`
screenstrokes `screenstrokes; showit` po `draw`, `fill`
shifted `(x,y) shifted (a,b);` $\sim (x + a, y + b)$ posunutí
show `show a,b;` výpis proměnných na obrazovku
showdependencies `showdependencies;` výpis proměnných s neznámou hodnotou, na kterých jiné proměnné závisí
showit `showit;` vykreslení aktuálního obrázku
showvariable `showvariable s;` výpis pole proměnných `s1`, `s2` ...
slanted `(x,y) slanted a;` $\sim (x + sy, y)$ zešikmení
special `special "labelfontat" numspecial 20;` nastavení velikosti popisu (viz dodatek GFtoDVI)
sqrt `b:=sqrt(a);` odmocnina
subpath `subpath (t1,t2) of p;` část cesty `p` mezi časy `t1`, `t2`
superellipse `superellipse(right_point,top_point,left_point,bottom_point,superness);` ovál
tension `draw z1..tension 1 and 2.5..z2;` napětí křivky u levého a pravého bodu
top `top z1=(0,y);` zarovnání na horní okraj podle tloušťky pera
transform `transform t;` proměnná typu transformace
transformed `currentpicture transformed t;` transformace
undraw `undraw p;` inverze `draw`, sníží počet nakreslení bodů
undrawdot `undrawdot z1;` inverze `drawdot`
unfill `unfill p;` inverze `fill`
unfilldraw `unfilldraw p;` inverze `filldraw`
uniformdeviate `uniformdeviate x;` náhodné číslo od 0 do `x`
unitsquare `draw unitsquare;` jednotkový čtverec, levý dolní roh `(0,0)`
unitvector `unitvector(x,y);` jednotkový vektor daného směru
unknown `if unknown mag: mag:=1; fi;` logický výraz, false pokud je `mag` definováno
up `z1:=up;` bod $(0, 1)$
whatever `z1=z2+whatever*(z3-z4);` proměnná nabývající hodnoty při použití, body `z1`, `z2` leží na přímce se směrnicí `z3-z4`
xpart `xpart z;` x-ová část
xscaled `(x,y) xscaled a;` $\sim (ax, y)$ násobení číslem x-ové souřadnici
ypart `ypart z;` y-ová část
yscaled `(x,y) yscaled a;` $\sim (x, ay)$ násobení číslem y-ové souřadnici
zscaled `(x,y) zscaled (a,b);` $\sim (xa - yb, xb + ya)$ násobení vektorů jako komplexních čísel

Dodatky

Konvence značení v tomto dokumentu:
`x,y,a,b,c,d,u` čísla, `z` vektor, bod, `p` cesta, pero, `s` řetězec, `t` transformace, čas (parametr křivky), `V` obrázek, `b` logická hodnota

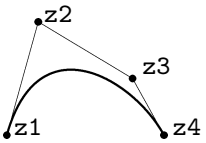
GFtoDVI
Program GFtoDVI slouží k prohlížení fontů vytvořených METAFONTem. Nejprve musíme vygenerovat font ve vyšším rozlišení (např. `\mode=proof; mag=1`). Poté spustíme `gftodvi <gf_file> [<dvi_file>]` (např. `gftodvi cscsc10.260`). Program GFtoDVI potřebuje k vytvoření souboru `dvi` metriky `cmr8.tfm`, `cmtt10.tfm`, `gray.tfm` a `logo8.tfm`. Soubor `dvi` (např `cscsc10.dvi`) prohlédneme z menu TEXu. Obrázek písmene bude obsahovat popisy bodů, které jsou zadány v souboru `mf` pomocí maker label, makelabel, penlabels, special (viz Index METAFONTu). Pro prohlížení souboru `dvi` jsou potřeba fonty `cmr8`, `cmtt10`, `gray` (zdrojový soubor `grayf.mf`), `logo8`

Literatura

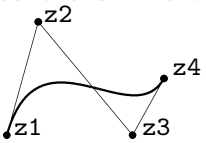
The METAFONTbook, Donald E. Knuth
TEX bulletin, Horák, K.: Můj zápas s METAFONTem

Miroslav Brož `miroslav.broz@usa.net`, `http://www.kolej.mff.cuni.cz/~broz/`
Přemysl Šedivý

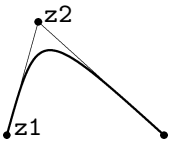
Bezierovy křivky



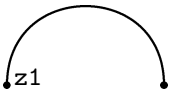

```
draw z1..controls z2 and z3..z4;
```



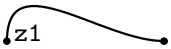

```
draw z1..controls z2 and z3..z4;
```




```
draw z1..controls z2..z4;
```

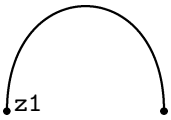



```
draw z1{dir90}..z2;
```



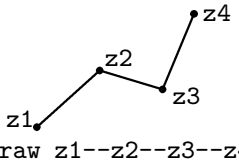

```
draw z1{dir90}..{dir0}z2;
```


```
draw z1{dir90}..tension 3..z2;
```

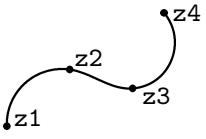



```
draw z1{dir90}..tension 0.75..z2;
```


```
draw z1{dir90}..tension 2 and 1..z2;
```



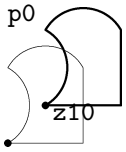

```
draw z1--z2--z3--z4;
```



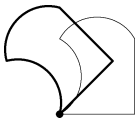

```
draw z1..z2..z3..z4;
```

Transformace

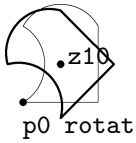
```
p0
```



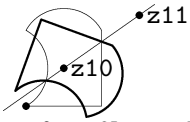

```
p0 shifted z10;
```



```
p0 rotated 45;
```

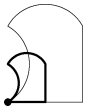



```
p0 rotatedaround
(z10,45);
```

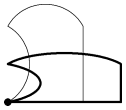



```
p0 reflectedabout
(z10,z11);
```

```
p0 slanted 1/3;
```

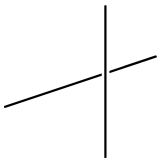


```
p0 scaled 0.5;
```

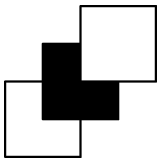


```
p0 xscaled 1.5
yscaled 0.5;
```

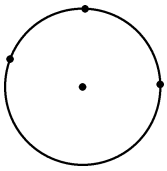
Jednoduché příklady



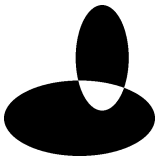
```
z1=(2/3w,0);
z2=(2/3w,h);
z3=(0,1/3h);
z4=(w,2/3h);
pickup pencircle scaled .3u;
draw z3--z4;
pickup pencircle scaled u;
erase draw z1--z2;
pickup pencircle scaled .3u;
draw z1--z2;
```



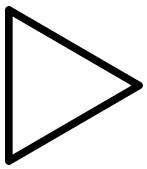
```
pickup pencircle scaled .3u;
draw unitsquare scaled 10u;
filldraw unitsquare
  scaled 10u shifted (5u,5u);
erase fill unitsquare
  scaled 10u shifted (10u,10u);
draw unitsquare
  scaled 10u shifted (10u,10u);
```



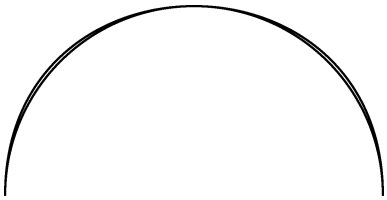
```
z1=(w,h/2);
z2=(1/2w,h);
z3=(0,2/3h);
z12=1/2[z1,z2];
z23=1/2[z2,z3];
z0-z12=whatever*((z1-z2)
  rotated 90);
z0-z23=whatever*((z3-z2)
  rotated 90);
R=length(z1-z0);
pickup pencircle scaled .3u;
draw fullcircle scaled 2R
  shifted z0;
pickup pencircle scaled u;
for i=0,1,2,3: drawdot z[i];
endfor;
```



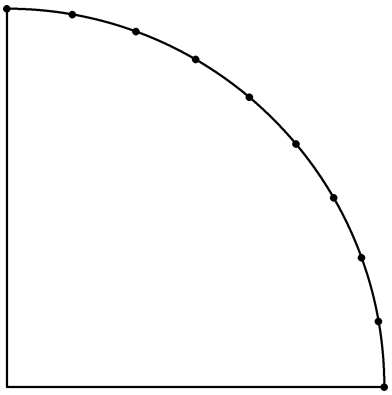
```
beginchar(6,20u#,20u#,0);
filldraw fullcircle xscaled 20u yscaled 10u shifted (10u,5u);
filldraw fullcircle xscaled 7u yscaled 14u shifted (13u,13u);
cull currentpicture keeping (1,1);
endchar;
```



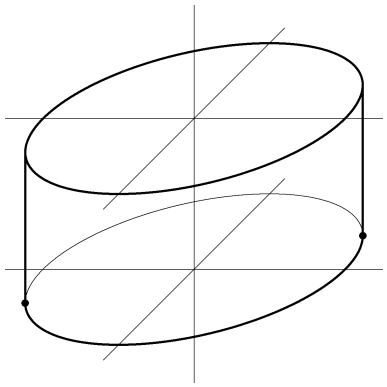
```
mode_setup; u#:=1mm#; define_pixels(u);

beginchar(1,20u#,20u#,0);
 z1=origin;
 z2=(0,h);
 z3=z2 rotated -60;
 pickup pencircle scaled 1u;
 draw z1--z2--z3--cycle;
endchar;
```



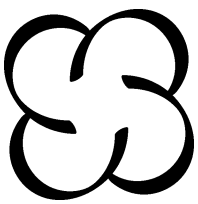
```
beginchar(50,50u#,25u#,0);
pickup pencircle scaled .3u;
draw halfcircle scaled w shifted (w/2,0);
draw origin{dir90}..(w,0);
endchar;
```



```
beginchar(51,50u#,50u#,0);
pickup pencircle scaled .3u;
draw (0,h)--origin--(w,0);
draw (0,h){dir0}..(w,0);
pickup pencircle scaled u;
for i=0 upto 9: drawdot (50u*cosd(10i),50u*sind(10i)); endfor;
endchar;
```



```
beginchar(52,50u#,50u#,0);
path p[];
p1=fullcircle xscaled 40u yscaled 20u slanted 1
  shifted (25u,15u);       p2=p1 shifted (20u*up);
pickup pencircle scaled 0.1u;
draw (0,15u)--(w,15u); draw (0,35u)--(w,35u);
draw (13u,3u)--(37u,27u); draw (13u,23u)--(37u,47u);
draw (25u,0)--(25u,50u);
t1:=directiontime up of p1; t2:=directiontime down of p1;
draw subpath (t1,t2) of p1;
z1= point t1 of p1; z2= point t2 of p1;
pickup pencircle scaled 0.3u;
draw p2;
draw subpath (0,t1) of p1; draw subpath (t2,8) of p1;
draw z1--(z1+20u*up); draw z2--(z2+20u*up);
pickup pencircle scaled u;
drawdot z1; drawdot z2;
endchar;
```



```
beginchar(4,30u#,30u#,0);
pickup pencircle scaled 2u yscaled 1/3 rotated 30;
transform t;
t=identity rotatedaround((.5w,.5h),-90);
x2=.35w; x3=.6w;
y2=.1h; top y3=.4h;
path p; p=z2{right}...{up}z3;
top z1=point .5 of p transformed t;
draw z1...p;
addto currentpicture also currentpicture transformed t;
addto currentpicture also currentpicture transformed (t transformed t);
endchar;
end;
```