Práce s grafikou v LaTeXu

How to work with graphics in LaTeX

bakalářská práce

Pavel Bouček

Vedoucí bakalářské práce: Mgr. Jiří Pech, Ph.D.
Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky
2009

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně, pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 18. dubna 2009

Anotace

Tato práce se zabývá prací s grafikou v LaTeXu a v METAFONTu. Jsou zde uvedeny grafické formáty obrázků, které jsou vhodné a jejich možná úprava pro vkládání do LaTeXu. Dále je zde seznámení s programem METAFONT a jeho následné nastavení v operačním systému Linux (distribuce Ubuntu 8.04 LTS). Poté je zde vysvětleno a ukázáno na příkladech, jakými způsoby lze vytvářet obrázky (čtverec, obdélník, kružnice, elipsa aj.).

Jako příloha této práce je sestaven z těchto jednotlivých obrázků jeden výsledný obrázek.

Abstract

This work deals with the work with graphics in LaTeX and METAFONT. Are given graphic image formats that are appropriate and possible change to be put in the LaTeX. There is a familiarity with METAFONT and its setting up in Linux (Ubuntu 8.04 LTS distribution). Then there is explained and shown examples of ways you can create images (square, rectangle, circle, ellipse, etc.).

In appendix of this work is composed of these individual files one resulting image.

Poděkování

Rád bych poděkoval v první řadě vedoucímu této bakalářské práce, v neposlední řadě své rodině, která mě od psaní práce nevyrušovala a respektovala můj volný čas, který jsem věnoval této práci a i své přítelkyni, která mi vypůjčila v knihovnách veškeré potřebné knihy.

Obsah

1 Úvod **7**

2 Cíl práce **8**

3 Metodika **9**

4 Výchozí stav problematiky **10**

5 Možnosti Práce s grafikou v LaTeXu **11**
5.1 Grafické formáty pro vkládání do LaTeXu 11
5.1.1 Vektorové obrázky 12
5.1.2 Rastrové/Bitmapové obrázky 15
5.2 Nástroje pro úpravu obrázků pro vložení do LaTeXu 17
5.2.1 bm2font 17
5.2.2 Autotrace 18
5.2.3 Gimp 18

6 METAFONT **19**
6.1 Nastavení METAFONTu v operačním systému Linux 20
6.2 Popis příkazů a následné vytvoření obrázku 21
6.2.1 Délkové jednotky 22
6.2.2 Záhlaví dokumentu 23
6.2.3 Kreslící nástroje 24
6.2.4 Kreslíme s METAFONTem 29

7 Závěr **37**

Literatura **38**

Přílohy **I**

A Doplňující obrázky v METAFONTu **II**
A.1 Ukázky běžného zápisu II
A.1.1 Rovnoramenný trojúhelník II
A.1.2 Rovnostranný trojúhelník III
A.1.3 Trojúhelník pomocí xpart a ypart IV
A.1.4 Čtverec IV
A.2 Ukázky zjednodušeného zápisu V
A.2.1 Příkaz `unitsquare` - Čtverec V
A.2.2 Příkaz `unitsquare` - Obdélník VI
A.2.3 Příkaz `fullcircle` - Kružnice VI
A.2.4 Příkaz `fullcircle` - Elipsa VII
A.2.5 Příkaz `halfcircle` - Půlkružnice VII

B Zdrojový kód výsledného obrázku **VIII**
B.1 Výsledný obrázek v METAFONTu XVI

Kapitola 1

Úvod

Na úvod bych nejprve představil a stručně sdělil něco z historie systému LaTeX, respektive systému TeX. Jedná se o systém na úpravu textových dokumentů a následného tisku v té nejlepší kvalitě jakou si může uživatel dopřát. Jistě každý z nás zná programy na úpravu textu od společnosti Microsoft a to konkrétně MS Office, za který musíte této společnosti pro jeho použití zaplatit nemalou finanční částku, která se pohybuje v řádu tisícikorun. Dalším programem na stejné bázi je program OpenOffice.org, který je zcela zdarma. Bohužel, ale ani tento program není dostačující. U některých oborů, jako je například matematika, je nepostradatelné zapisování vzorců do textu a tento zápis je v LaTeXu velice příjemně vyřešen. Málo kdo už zná, že existuje i program LaTeX na sázení textu, který vznikl daleko před těmito programy typu Microsoft Office atd.

LaTeX je jedna z mnoha nadstaveb TeXu, které jsou zcela zdarma. TeX vznikl v sedmdesátých letech. Jeho tvůrce Donald Ervin Knuth pochází ze Stanfordské univerzity, který na základě špatného tisku vytvořil program TeX. Autor typografického systému vytvořil také program METAFONT, který původně vznikl na tvorbu fontů respektive písma a též na tvorbu vektorových obrázků, tzv. pérovek, o kterých bude převážně tato bakalářská práce. TeX však nebyl pro začátečníky a příležitostné uživatele. Práce v něm byla značně obtížná a velice zdlouhavá. Proto vznikla nadstavba LaTeX, ve které se formátuje text stejně jako u TeXu pomocí příkazů, avšak zde se uživatel vyhne zdlouhavé tvorbě maker, kde se dá nastavit i ten nejmenší detail k vlastnímu výstupnímu vzhledu dokumentu. LaTeX má předdefinovaná makra. V tomto případě je makro znázorněno jako nový příkaz a nemusí je uživatel zdlouhavým způsobem nastavovat a vytvářet. Jsou zde automaticky nastavena a drží se správné typografie.

Kapitola 2

Cíl práce

Ve své práci se budu zabývat typografickým systémem LaTeX a prací s grafikou v něm. Tuto bakalářskou práci jsem si vybral za účelem poznání nepoznaného a pochopení tvorby obrázků pomocí programu METAFONT, kde bych chtěl jako úplný začátečník s tímto programem krok po kroku seznámit čtenáře s jednoduchou tvorbou pomocí geometrických útvarů (trojúhelník, čtverec, kružnice a jiné) samozřejmě pro začínající uživatele METAFONTu. V závěru této Bakalářské práce bych chtěl všechna pravidla, se kterými čtenáře krok po kroku budu seznamovat, shrnout a vytvořit odpovídající obrázek.

Cílem práce je tedy seznámení naučnou formou s tímto programem na takové úrovni, aby i úplný začátečník mohl pracovat s METAFONTem a na základě této práce mohl též vytvářet různé obrázky, schémata a podobně, podle své kreativity.

Byl jsem seznámen s několika materiály, které mě osobně měly naučit v tomto programu pracovat, ale z mého pohledu úplného začátečníka, byly tyto materiály později po seznámení s METAFONTem spíše jen výpisky z knihy *The METAFONT book* od Donalda Knutha. Bohužel tuto knihu jsem nesehnal, avšak jsem o ní četl při výběru literatury pro tuto práci. Proto bych se chtěl pokusit seznámit s METAFONTem začínající uživatele z jiné stránky a to ze stránky méně náročné pro úplného začátečníka, aby i on mohl vytvářet lehké a později i podle naučených dovedností složitější obrázky.

Kapitola 3

Metodika

Na začátku práce seznámím uživatele s LaTeXem a prací s grafikou v něm. Jak vkládat a upravovat obrázky do systému LaTeX. Poté čtenáře seznámím s programem METAFONT. Zde se pokusím popsat instalaci a veškeré potřebné nastavení v operačním systému Linux, distribuce Ubuntu 8.04 LTS, pro správný chod tohoto programu. Nadále bude začínající uživatel pokračovat se seznamování s METAFONTem, jaké delkové jednotky může použít, jak by mělo vypadat záhlaví obrázku, ve kterém mohou uživatelé předem definovat hodnoty.

Dále vysvětlím jak a pomocí čeho kreslíme obrázky v METAFONTu. Vytvoření jednoduchých geometrických útvarů, mezi které patří přímka, čtverec, obdélník, kružnice, půlkružnice, čtvrtkružnice, elipsa a obrázek vytvořený pomocí křivky, konkrétně sinusoida.

Na závěr z těchto útvarů vytvořím obrázek, který bude obsahovat všechny tyto útvary kromě sinusoidy, ta bude nahrazena jinou křivkou. V tomto případě se bude jednat o konečný obrázek bokorysu domu, vedle domu nakreslen strom, na obloze usmívající se slunce a mrak, který je nahrazen sinusoidou.

Kapitola 4

Výchozí stav problematiky

Základní publikací, se kterou jsem pracoval byla publikace *LaTeX pro začátečníky* [1]. V této publikaci se snaží autor seznámit čtenáře s typografickým programem LaTeX v co největší míře, od instalace systému, prvního spuštění, po samou tvorbu celého dokumentu. Tato kniha je též vhodná pro zkušené uživatele LaTeXu, kteří občas potřebují při tvorbě dokumentů nápovědu, respektive manuál. Bohužel je zde obsažen jen okrajově METAFONT.

Další knihou, která byla nepostradatelná při práci s METAFONTem byla kniha *Typografický systém TeX* [2], kde se nejedná jako v předešlé literutuře *LaTeX pro začátečníky* o postupné seznamování s LaTeXem, ale spíše o řešení daných problémů.

Velikým přínosem pro mou práci byly stránky sdružení CSTUG, odkud jsem čerpal z elektoronické knihy *Kreslíme METAFONTem* [3]. Tato elektronická kniha seznamuje uživatele s možnostmi METAFONTu.

Nesmím též zapomenout na elektronickou knihu, která byla vydaná prostřednictvím sdružení CSTUGu a to na knihu *Můj zápas s METAFONTem aneb pérovky a jiná zvěrstva* [4], kde autor popisuje a seznamuje čtenáře s prací v METAFONTu.

Dalším přínosem byla elektronická publikace, kde je vysvětlena tvorba obrázků v METAFONTu tzv. rychlokurzem. Dle mého názoru je to velice přínosná publikace, ale nikoliv pro úplného začátečníka. Jedná se o internetovou publikaci *METAFONT - Jak kreslit obrázky* [5].

Kapitola 5

Možnosti Práce s grafikou v LaTeXu

LaTeX byl vytvořen pro sazbu textu a vytváření dokumentu pro tisk v nejlepší kvalitě. Občas se však uživatel neobejde bez vložení, úpravy či celé tvorby obrázku v dokumentu. V LaTeXu můžeme vkládat, upravovat a dokonce i vytvářet obrázky.

Pro tvorbu vektorové grafiky v LaTeXu existuje prostředí picture, ve kterém obdobně jako v METAFONTu uživatel definuje pomocí souřadnicových bodů x a y výsledný obrázek. Prostředí picture má však omezené možnosti, ale většině uživatelům bude stačit. Pokud uživatel chce či potřebuje kreslit složitější obrázky nebo si vytvořit svůj vlastní font, je zde program METAFONT, o kterém je převážně tato práce. Tímto bych rád odkázal začínající uživatele na knihu LaTeX pro začátečníky [1], kde je velice dobře popsaná situace s tvorbou obrázků v LaTeXu.

5.1 Grafické formáty pro vkládání do LaTeXu

V LaTeXu je možno vytvářet vektorové či vkládat bitmapové obrázky, ale jelikož LaTeX pracuje s textovým vstupem a tím bitmapové obrázky nejsou, je potřeba mít na paměti, že LaTeX je defaultně nastaven na výstup do dokumentu PostScript. Je to pomocí ovladače dvips. Musíme tedy obrázky v nějakém grafickém programu předem převézt do souboru *.eps* či *.ps* a poté bez problému vložit do LaTeXovského dokumentu. Jsou zde pochopitelně i další možnosti, kterými vložíme obrázek i v jiném formátu. Například vložení běžného bitmapového obrázku .bmp, .jpg, .png a jiných formátů pomocí ovladače pdfLaTeX. Tímto způsobem nebude výstup dokumentu v PostScriptu, ale v pdf. Výstup může být i v PostScriptu, ale vložené obrázky by

se ve výstupním dokumentu nezobrazovaly. Vhodným nástrojem pro změnu formátu obrázku je například grafický program Gimp, který si uživatel může nainstalovat, jak v operačním systému Linux, tak i v MS Windows. Je zcela na každém uživateli, se kterými obrázky bude pracovat a za jakým účelem bude vytvářet dokument.

Vkládání obrázků - úvod

LaTeX byl vytvořen pro sazbu textu, proto pokud potřebujeme vložit nějaký obrázek do dokumentu, který je tvořen LaTeXem, budeme muset počítat s tím, že vložení obrázku je až na druhém místě, tedy po vkládání textu. Ve většině DTP programů je to naopak. Když sázíme obrázek v jiném DTP programu, tak se text, do kterého je vkládán obrázek, sám přizpůsobí obrázku. V LaTeXu je to obráceně. Byl vytvořen hlavně na text, kde ale též je možné vkládání obrázků. Plovoucí obrázky, které se nám automaticky přemisťují z místa na místo dle textu, se do LaTeXu zadávají pomocí prostředí figure. Toto prostředí má též různou možnost číslování obrázků a jeho popis.

5.1.1 Vektorové obrázky

Vektorovým obrázkem je chápán obrázek, který je založen na matematických výpočtech. Pomocí čar a křivek nám zobrazuje konkrétní obrazec. Vektorový obrázek je vytvořen pomocí souřadnic bodů x a y, kde jejich body, které jsou následně spojeny vytváří námi definovaný obrázek. Obrázek, který je takto definovaný můžeme libovolně zmenšovat či zvětšovat, aniž bychom ztratili kvalitu obrázku. Tento obrázek je též vhodný pro uživatele, kteří k tisku obrázků nepotřebují barevné obrázky či jejichž tiskárny barevný tisk ani nepodporují. Tvorba vektorového obrázku je pro uživatele, kteří například upravují fotografie zcela zbytečnou. Pro některé uživatele, kteří pomocí souřadnic bodů mají nakreslit obrázek, zase věcí nezbytnou.

Výhodou vektorových obrázků je možnost libovolného bezestrátového zvětšování a velikost na paměťovém médiu. Tento způsob má uplatnění v mnoha oborech. Jedná se hlavně o vytvoření obrázků, které neodmyslitelně patří k oborům jako je fyzika, matematika, chemie, elektortechnika a jiné. U těchto oborů též existují novější a jednodušší programy na tvorbu obrázků. Tyto obrázky jsou ale většinou v programech ukládány do bitmapových obrázků, kde se například po zvětšení obrázku zhorší jeho kvalita. I přes stáří LaTeXu, je tento program vhodným východiskem.

picture - prostředí

Pokud chceme vytvořit obrázek přímo v LaTeXu, nakreslíme ho pomocí prostředí picture. Jedná se o prostředí, kde definujeme pomocí příkazů jednotlivé body obrázku, které poté spojíme. Je to obdobné vytváření obrázku jako v METAFONTu, o kterém se budu v této práci ve větší míře zmiňovat v dalších kapitolách.

Jedná se o vytvoření obrázku, který se chová stejně jako jedno písmeno v LaTeXu. Pomocí prostředí picture můžeme vytvářet jednoduché vektorové obrázky, které zadáváme do LaTeXu podobným způsobem jako samotné příkazy, kterými formátujeme text v dokumentu.

Vektorové formáty

Formáty, které se nejčastěji používají při tisku ve vektorové grafice:

Tyto formáty jsou nejvhodnější pro vkládání vektorových obrázků do LaTeXu:

- **.eps** – zapouzdřený postscriptový soubor - nejvhodnější formát pro vkládání do LaTeXu společně s formátem .ps
- **.ps** – stránkový popisový jazyk, který je určen jako primární pro elektronickou a počítačovou sazbu
- **.ai** – jde o vektorový formát pro aplikaci Adobe Illustrator ArtWork

Tyto formáty se moc nepoužívají, nejsou ani vhodné pro vkládáni do LaTeXu:

- **.wmf** – grafický formát souboru pro Microsoft Windows (povoluje i rastrovou grafiku).
- **.cgm** – používáno v leteckém průmyslu. Jde o Metasoubor počítačové grafiky.
- **.dxf** – grafický formát CAD systémy.
- **.cdr** – formát pro Corel Draw.
- **.svg** – značkovací jazyk a formát popisující 2D vektorovou grafiku pomocí jazyka XML.
- **.zmf** – formát pro grafický program Zoner Calisto.
- **.pdf** – založen na PostScriptu. Přeložíme-li dokument pdfLaTeXem, můžeme použít přímého vkládání .pdf obrázků do dokumentu.

PostScript

Obrázek uložený či vytvořený v PostScriptu je nejpoužitelnější pro vkládání do LaTeXu. Vznikl v roce 1985 firmou Adobe jako jazyk pro popis stránek.

Tento programovací jazyk se stal univerzálním jazykem pro komunikaci grafických aplikací. Používají ho tiskárny a tiskové procesory, je základním (a dlouhou dobu byl jediným) výměnným formátem vektorové grafiky, v UNIXu je téměř výhradním jazykem pro tisk z aplikací.

Použito z www stránek [6]

Jedná se o souřadnicový systém, kde pomocí souřadnicových os x a y, určíme text a grafické prvky, které se zadávají v prostoru pomocí bodů.

Obrázek vytvořený v Postscriptu vložíme do LaTeXu pomocí balíčku graphicx, který je v základní instalaci LaTeXu. Tento balík musíme do našeho dokumentu naimportovat, uděláme to příkazem:

```
\usepackage[driver]{graphicx}
```

Poté můžeme vložit obrázek ve formátu *.eps* do LaTeXu pomocí příkazu:

```
\includegraphic[parametry]{název_souboru.eps}
```

Pokud *parametry* nevyplníme, bude vložený obrázek bez úprav. Za *parametry* můžeme dosadit příkazy:

`angle` - natočení obrázku o libovolný počet stupňů
(v kladném i záporném směru)
`height` - výška obrázku
`width` - šířka obrázku

Například obrázek pojmenovaný `obrazek.eps`, který chceme otočit o *90°* a jeho šířku požadujeme o velikosti *10cm*, zapíšeme následovně:

```
\includegraphic[angle=90, width=10cm]{obrazek.eps}
```

Použito z www stránek [7]

Též je možné obrázek vkládat pomocí balíku pgf, který nám slouží při opakovaném vkládání obrázků, kde se nám uloží jen 1x tentýž obrázek. Má také možnost pracovat s poloprůhlednými obrázky.

5.1.2 Rastrové/Bitmapové obrázky

Jedná se o obrázky, které jsou tvořeny pomocí jednotlivých pixelů, například fotografie. Jejich výhodou je vkládání těchto obrázků do dokumentu. Nevýhodou je však velikost obrázku na paměťovém médiu. Další nevýhodou je při případných změnách velikosti obrázku či jiné transformaci obrázku zhoršení kvality celého obrázku. Obrázek ztrácí původně zbarvené pixely a obrázek je deformovaný. Například k této ztrátě barevných pixelů při zvětšení či zmenšení obrázku, dochází k nechtěnému efektu zubatění nebo ztráty či prolínání některých barev. Pokud tedy potřebujeme vytvořit obrázek, který má několik odstínů barev, použijeme tuto formu. Pokud však potřebujeme vytvořit obrázek pomocí matematických výpočtů, tedy pomocí geometrických útvarů, zvolíme vektorovou formu.

Rastrové/Bitmapové formáty

Formáty, které se nejčastěji používají při tisku v rastrové grafice:
Tyto formáty jsou nejvhodnější pro vkládání rastrových obrázků do LaTeXu:

- **.bmp** – BitMaP, základní formát pro tzv. Bitmapy. Kapacitně velký výstupní soubor v řádech MB. Používá se převážně kvůli své bezstrátové komprese.
- **.pcx** - Paintbrush, původně byl tento formát k ukládání obrázků v programu PC Paintbrush. Tento formát má jako předchozí formát bezstrátovou kompresi.
- **.gif** – Graphics Interchange Format, podporuje nejvíce 256 barev s bezstrátovou kompresí. Formát *.gif* rozdělujeme dále na dva typy:

  `GIF87a`: Podporuje prokládání a ukládání multiple souborů, je standardem.
  `GIF89a`: Rozšiřuje předchozí o průhlednost, textové komentáře a animaci objektů. [8]
- **.tiff** - Tag Image File Format, neoficiální standard pro ukládání rastrové grafiky. Původně pro černobílý tisk, dnes i pro barevný.

- **.tga** - Targa, formát vhodný pro nekomprimovaný soubor v 32bit barevné paletě. Dnes se již nevyužívá. Tento formát se využívá v oblasti počítačových her a pro tvorbu realistické grafiky. Například ve hře Warcraft III je zde použit tento formát při zachycení snímku obrazovky, tzv. screenshot. V prosředí 3D grafiky je například využit ve hře Half life.
- **.xpm** - X PixMap, tento formát je využit v grafickém prostředí X Window System. Tento soubor je editován či upravován v kterémkoliv textovém editoru.

Pro černobílou grafiku jsou vhodné formáty: .bmp, .gif a .pcx. Pro barevnou grafiku formáty .tiff a .tga. Pro systém X Window je vhodný poslední formát .xpm.

5.2 Nástroje pro úpravu obrázků pro vložení do LaTeXu

Pokud bychom potřebovali vložit a následně upravit rastrový obrázek pro náš dokument vytvořený v LaTeXu, tak můžeme v některých grafických programech převést daný obrázek například na znaky. K tomu slouží program bm2font, který je zdarma a vytvořil ho Friedhelm Sowa.

Potřebujeme-li změnit bitmapový obrázek na vektorový, můžeme použít například program Autotrace, který nám z formátu Bitmap vytvoří formát a změní jej na PostScript. Tento program je též zdarma a jeho tvůrce je Martin Weber.

Když bychom chtěli změnit rastrový obrázek .bmp na .eps či .ps, se kterými LaTeX pracuje, tak je možno upravit v grafickém programu Gimp. Převod je v tomto případě pro zobrazení obrázků v LaTeXu zcela nezbytný. LaTeX s formáty .pdf, .bmp, .jpg nepracuje, museli bychom místo LaTeXu použít pdfLaTeX. pdfLaTeX nepracuje s formáty .eps, .ps. Tedy vše záleží na uživateli.

5.2.1 bm2font

bm2font pracuje s grafickými formáty typu .gif, .pcx, .bmp, .iff/.lbm, .tiff, .img, .cut a dalšími čistými bitmapovými soubory, kde je nutno zadat počet bytů na jeden řádek obrazu.

Princip práce programu spočívá v rozdělení obrazu na pravoúhlé části, z kterých jsou vygenerována písmena jednoho nebo několika fontů. Zde dochází k vytvoření .pk souboru a příslušného souboru .tfm. [9]

Soubor převedeme následným příkazem:

```
bm2font -h640 -v480 názevsouboru.tif,
```

kde:

h - nám udává horizontální pozici (číslo 640 vyjadřuje rozlišení obrázku v x-ové souřadnici)

v - nám udává vertikální pozici (číslo 480 je v x-ové souřadnici
a výsledný obrázek má rozlišení 640x480 dpi)

Poté se nám vygeneroval soubor *názevsouboru.tfm* společně se soubory *názevsouboru.pk* a *názevsouboru.tex*

Pro další práci a pro zobrazení obrázku v LaTeXovém dokumentu použijeme příkazy:

`\input{`*`názevsouboru.tex`*`}` – pro připojení definičního souboru
`\fbox{\set` *`názevsouboru`*`}` – pro vlastní zobrazení [9]

5.2.2 Autotrace

Autotrace je program, který převádí nebo-li konvertuje bitmapové soubory na vektorové. Převádí vstupní bitmapové soubory .bmp, .tga, .pnm, .ppm, .pgm, .pbm a jiné podporované programem ImageMagick na výstupní vektorové soubory PostScript, .svg, .xfig, .swf, .pstoedit, .emf, .dxf, .cgm, .mif, .p2e a .sk. [10]

Tento program se zdá býti dobrým konvertorem bitmapových souborů na vektorové. Instalace je možná jak v operačním systému Linux, tak i v MS Windows. Je zcela zdarma a to byl též důvod pro uvedení tohoto programu v této práci.

Dalšími možnými programy jsou například AdobeStreamline nebo CorelTrace, které ale ani nekonvertují obrázky v takové kvalitě jako Autotrace. Obdobný program je program Potrace, ale ten bohužel umí pracovat jen s černobílými obrázky. Autotrace umí pracovat i s barevnými.

5.2.3 Gimp

Tento program je výborný pomocník při tvorbě bitmapové grafiky. V našem případě je též vhodný pro převedení bitmapového obrázku například ze souboru .bmp, .jpg, .png a jiného do našeho dokumentu do formátu .eps nebo .ps, avšak též do bitmapového obrázku. Tento program sice nezvládá převedení bitmapového obrázku do vektorového, ale určitě nám bude velkým pomocníkem, který je též zcela zdarma, při tvorbě LaTeXovského dokumentu. Tento program si uživatel může nainstalovat jak do operačního systému Linux, tak i do MS Windows.

Kapitola 6

METAFONT

METAFONT jak už z názvu vyplývá je program na tvorbu fontů, tedy písma nebo obrázků, které se v dokumentu tváří jako písmo. Vytvořený obrázek můžeme použít jako písmo v LaTeXu. Jeho vývoj byl, dalo by se říci, ukončen. Jen jednou za rok se opraví chyby, které byly uživateli nalezeny.

Jedná se o velice stabilní program pro tvorbu a návrh písma, ale v dnešní době bych si dovolil i říci, že spíše než na tvorbu písma je vhodnější na tvorbu vektorových obrázků, tzv. pérovék.

Jeho využitelnost je převážně pro tvorbu různých diagramů, schémat, geometrických útvarů a jiných. Všude tam, kde je potřeba použít nějakého obrázku v dokumentu, kde si můžeme představit ruční zdlouhavé kreslení tuší.

V současnosti je nespočet kvalitních písem, které jsou zdarma, takže je celkem zbytečné se v této práci zabývat tvorbou nějakého písma, navíc se nepovažuji v žádném případě za odborníka, který vytváří písma. Proto jsem se vydal touto jednodušší cestou a to tvorbou jednoduchých obrázků pro úplné začátečníky. Věřím, že tak bude tato cesta i zábavnější.

METAFONT je velice podobný programu LaTeX. Vstupem METAFONTu je též textový soubor, v němž jsou nadefinovány, pomocí geometrických pravidel, jednotlivé tahy souřadnicových bodů, ze kterých se poté stává nějaký námi nadefinovaný grafický výstup, tedy v našem případě obrázek. Hlavním souborem pro čtení dokumentů je soubor .mf. Do tohoto souboru zapisujeme příkazy pro náš požadovaný výstup. Je to hlavní soubor pro generování fontu nebo písma.

6.1 Nastavení METAFONTu v operačním systému Linux

Zřejmě nejstabilnějším programem pro MS Windows je program MikTEX od Christiana Schenka. Uživatel si může stáhnout tento program zdarma. Jeho výhodou je snadná instalace a nastavení. Vše potřebné se dá jednoduše zjistit ve vyhledávačích webových stránek. Proto se zde nebudu zabývat nastavením a instalaci tohoto programu.

Osobně pracuji, jak v operačním systému Linux, tak v MS Windows. Pro mě osobně bylo velice složité se rozhodnout pod jakým operačním systémem budu tuto práci vypracovávat. Nakonec jsem zvolil Linux, ve kterém bylo vše napsáno.

Tímto jsem se tedy rozhodl pracovat v Linuxu, konkrétně na Ubuntu 8.04 LTS, kde jsem vše bez jakéhokoliv problému stáhl a nastavil. Ze správce balíků Synaptic jsem vyhledal program TEX Live 2007-13, který podporuje, jak TEX, LATEX, tak i práce s METAFONTem. Dále jsem nainstaloval přes příkazový řádek (Terminál) program *Midnight commander*, pomocí příkazu `sudo apt-get install mc`. Tímto programem jsem konvertoval soubory, jak *.tex*, tak i *.mf* do dalších nezbytných souborů pro správnou funkcionalitu vytvářeného dokumentu.

Vytvořil jsem si textový soubor, který jsem nazval obrázek a příponu tohoto nově vytvořeného textového souboru jsem zadal *.mf*. Tedy výsledný textový soubor byl nazván `obrazek.mf`. Takto se značí hlavní soubor, se kterým METAFONT pracuje. Pro představu v LATEXu je vytvořen hlavní soubor s příponou `.tex`. Do tohoto souboru se pak vypisuje a tím i definuje pomocí zadávání příkazů navrhnutá písma nebo v našem případě obrázky. Jelikož jde o celkem rozsáhlou možnost tvorby v METAFONTu, budu se konkrétními příkazy zabývat v dalších podkapitolách. Budu tedy seznamovat začínajícího uživatele postupně a nikoliv tzv. vychrlením veškerých informací ihned bez jakýchkoliv předchozích znalostí.

Poté jsem spustil Terminál (příkazový řadek v Linuxu) a spustil jsem příkazem `mc` Midnight commander, podobný Total Commanderu ve Windows. Vyhledal jsem místo, kam jsem soubor uložil a následně jsem spustil z příkazového řádku Midnight commanderu program METAFONT příkazem `mf`. Zobrazilo se okno, ve kterém byla vypsána verze programu METAFONT, konkrétně se jednalo v mém případě o verzi 2.71828 (Web2C 7.5.6), kde se číslo verze přibližuje k Eulerovu číslu `e`. Pod číslem verze se též zobrazilo **. Za těmito dvěma hvězdičkami METAFONT očekává odpověď uživatele. Zde jsem vypsal podle knižní publikace *LATEX kompletní průvodce* příkaz `mode=localfont; mag=1; input obrazek` [11], kde

po tomto vypsání příkazu a následném odkliknutí klávesy enter METAFONT vytvořil pořebné soubory.

Jedná se o soubor s příponou .xxxgf, kde za xxx se skrývá číslo, respektive rozlišení tiskárny v dpi a gf vyjadřuje generický font. Pokud bychom ve vypsaném příkazu udali hodnotu mag=2, tak toto znamená, že výsledná hodnota na výstupu bude 2x větší. Například se nám vygeneruje soubor o velikosti 600gf a při zapsání předchozího příkazu znovu, ale s hodnotou mag=2, vygeneruje se nám generický font v rozlišení 1200gf, tedy pro tiskárnu 1200 dpi. Formát `.xxxgf` není komprimovaným formátem, proto ho zkomprimujeme do formátu `.pk` pomocí programu *gftopk*, který je v každé distribuci METAFONTu či TEXu příkazem `gftopk obrazek.xxxgf obrazek.xxxpk`.

Se souborem *.gf* se nám též vytvoří soubor, který má příponu *.tfm*. Tento soubor nám vytváří metrické údaje.

Dalším vytvořeným souborem je soubor textit.log. Do tohoto souboru se zapíší po překladu předchozím příkazem údaje o fontu, respektive obrázku.

Dále můžeme převést soubor `obrazek.xxxgf` do DVI souboru, který nám umožní výsledný zpracovaný obrázek zobrazit na monitoru, pomocí příkazu `gftodvi obrazek.xxxgf`. Teď už jsem jen v prohlížeči souborů DVI výsledný obrázek prohlédl na výstupním zařízení, zda-li jsem nakreslil přesně to, co jsem definoval různými příkazy. V případě opravení výstupního obrázku musí uživatel METAFONTu opravit konkrétní chybu a poté znovu postupovat podle výše vypsaných instrukcí. Vrátil jsem se zpět do souboru `obrazek.mf` a upravil zdrojový kód a celý překlad jsem absolvoval znovu a znovu. Dokud jsem nezískal přesně ten obrazec, který jsem požadoval. V tomto případě jednoznačně platí pravidlo, čím více člověk dělá s tímto programem, tím více mu porozumí a také obrázky jdou poté rychleji a pohodlněji nakreslit a urychlí se tím i práce při různých složitějších vyobrazených schématech. Samozřejmě toto není tak jednoduché, jak se zdá, proto jsem si připravil více menších ukázek, u kterých bych práci s METAFONTem upřesnil a pokusil se na nich vysvětlit problematiku. Nebudu se v této práci zabývat kreslením různých náčrtků či schémat, které jsou například z oblasti elektroniky, ale zkusím toto vzít trošku z jiného pohledu a snad i zábavnějšího.

6.2 Popis příkazů a následné vytvoření obrázku

Abychom mohli začít pracovat s METAFONTem musíme znát zápis, který budeme zapisovat do souboru .mf, který ukáži na vhodných příkladech. Tak jako v LATEXu se tvoří příkazy, tak je tomu stejně v METAFONTu, ale s takovým rozdílem, že pochopení podstaty tvorby v LATEXu je méně

časově náročné, než v samém METAFONTu. Zde je to více náročné na čas a ze začátku má většina začínajících uživatelů spíše pocit, že by udělali stejný obrázek, například v Adobe Photoshop pro MS Windows či v Gimpu pro Linux daleko rychleji než v METAFONTu. Osobně jsem měl též stejný pocit, protože už pár let program Adobe Photoshop používám a znám ho v patřičné míře, abych udělal i v tomto programu časově rychleji nakreslený obrázek, než v METAFONTu. Ze začátku mě toto velice odrazovalo a časově velice zpomalovalo, ale jak čas ubíhal, tak i některé zkušenosti přibývaly a v závěru se vůbec nemohl obrázek vytvořený v kterémkoliv grafickém programu srovnávat s vytvořeným obrázkem v METAFONTu. Tedy pokud máme na mysli tvorbu černobílých obrázků pro následný tisk, například pro matematické či elektrotechnické a i jiné odborné dokumenty atd.

Máme vytvořený soubor .mf, ale co dál, co zde budu muset napsat, abych podle výše popsané minimální zkušenosti mohl začít pracovat s programem? To se pokusím popsat v následujících řádcích.

Abychom mohli s tímto programem začít pracovat, je mou povinností upřesnit a popsat pár základních věcí, bez kterých se s prací v METAFONTu neobejdeme. Jedná se například o délkové jednotky, se kterými METAFONT pracuje, dále například druhy pera, která můžeme použít a další základní vědomosti pro tvorbu obrázku.

6.2.1 Délkové jednotky

METAFONT je program, kde se obrázky kreslí pomocí zadávání pozic souřadnicových bodů pomocí kartézského souřadnicového systému. Neměl bych v této práci zapomenout na celkové seznámení s jednotkami, které nám určí velikost čar atd. Body jsou určeny pomocí x a y souřadnicové osy od souřadnice (0,0). Základní jednotkou, se kterou METAFONT pracuje je 1px (pixel). Při tvorbě písma či obrázku METAFONT pracuje také s milimetrem, centimetrem, palcem a také s délkovými typografickými jednotkami (viz tabulka 1), které z hlediska praxe jsou vhodné pro zmenšování či zvětšování obrázku bez jeho ztráty. Tedy obrázek zůstane při jakémkoliv zásahu při změně velikosti pořad stejný.

Ve veškerých příkladech zde budu používat délkovou jednotku pica. Tato jednotka se mi ukázala při tvorbě obrázků v METAFONTu nejideálnější. Zejména pak při tvorbě jednotlivých obrázků, kde jsem původně měl nastavenou délkovou jednotku milimetr. Tato jednotka se mi zdála z úplného počátku mých výtvorů zcela idální, avšak v budoucí práci s touto jednotkou mi přišly obrázky až přehnaně velké, až zcela při dalších pokusech na METAFONTu nepoužitelné. Navíc, když jsem použil jednotku pica,

tak výsledný obrázek jsem nemusel převádět do dalších a dalších souborů, abych mohl ihned vidět můj vytvořený obrázek na výstupním zařízení, resp. v tomto případě na monitoru. Použil jsem k této možnosti příkaz `screenstrokes;` o kterém se čtenář dočte v podkapitole této kapitoly *Záhlaví dokumentu.*

Jednotka	**Význam**
`mm`	milimetr
`cm`	centimetr (= 10 mm)
`in`	Palec (inch) = 25,4 mm
`pt`	Anglosaský typografický bod, tiskařský bod (0,351mm, 72,27 pt = 1 in)
`pc`	pica = 12 pt (= 4,212 mm)
`bp`	big point (72 bp = 1 in) (= 0,3527 mm)
`dd`	Didôtův bod (0,3759 mm, 1157 dd = 1238 pt)
`cc`	cicero = 12 dd (= 4,5108 mm)

Tabulka 6.1: Délkové jednotky

6.2.2 Záhlaví dokumentu

Záhlaví, které slouží pro práci s METAFONTem, kde si můžeme nadefinovat, jakou vstupní délkovou jednotku budeme brát v úvahu – viz výše, vytvoříme následovně:

```
mode_setup;                  % nastavíme velikost rozlišení, zvětšení
                             % vstup. souboru při spuštění METAFONTu
u#:=1pt#;                    % pro kreslení použij jednotku pt.
define_pixels(u);            % jednotku pt převede do výstupu na pixely
                             % (výstup pracuje s obdélníkovými
                             % nebo čtvercovými pixely).
screenstrokes;               % náhled obrázku po prvním přeložení.
                             % (nepovinné, avšak velmi užitečné)
beginchar(1,20u#,30u#,0);    % tento příkaz nám nastaví velikost plátna,
                             % ve kterém kreslíme obrazce.
```

Příkazy použity z elektronické knihy KRESLIME.dvi [3]

Číslo *1* vyjadřuje jaké jsme vybrali číslo znaku v ASCII tabulce. Udává číslování obrázků. Jeden obrázek jedno číslo, jako je tomu u fontů.
20u je k určení *x-ové* velikosti obrázku (tedy šířky obrázku) v pt.
30u znázorňuje *y-ovou* velikost obrázku (výška obrázku) v pt.
Číslo *0* je pro trojrozměrné obrázky (v našem případě nastaveno na 0).

6.2.3 Kreslící nástroje

Bod

Bod je základním pojmem pro kreslení v METAFONTu, který pomocí nadefinovaných souřadnic vygeneruje místo kam se má vykreslit daný bod za pomocí nástroje pera. Má dva parametry x a y, které zapisujeme do závorky.

Bod se zapisuje takto:

```
z1=(0,0);          % bod z1 na x-ové a y-ové souřadnici je 0,0
                   % – tj. počáteční stav.
z2=(20u,0);        % bod z2 je na x-ové souřadnici vzdálen 20 pt
                   % a na y-ové souřadnici je 0.
z3=(20,10);        % bod z3 je na x=20 pixelů a y=10 pixelů.
```

Tedy pokud si v záhlaví popíšeme, že požadujeme jednotky v pica, to znázorňuje `u#:=1pt#;` a například zapomeneme uvést písmenko *u* za námi definované číslo, způsobí nám to, že se nebude měřit daná hodnota v pt, ale v pixelech.

Můžeme použít tento zápis nebo můžeme použít zápis, kde nemusíme body *z1* – *zn* definovat.

Tento zápis bude vypadat například takto:

```
draw (0,0)--(20u,0)--(20u,10u)--(0,0);
```

Tímto zápisem jsme ušetřili 3 řádky kódu, ale tento zápis byl jen další možnost, jak se dá v METAFONTu zapsat body do souboru mf. Tento zápis bych nedoporučoval jako předchozí zápis. Předchozí je více přehlednější, než-li tento zápis a když se budete v budoucnosti k obrázku vracet, určitě oceníte přehlednost a rychlejší orientaci, jak je tomu v předchozím případě.

Další možnost, jak zapsat body. V tomto našem dalším případě nám automaticky sám METAFONT vyhledá požadované body a spojí je dle našeho požadavku.

Zjištění bodů tzv. xpart a ypart:

Pokud potřebujeme spojit ostatní body, které jsme vypsali nebo si chceme ulehčit zápis, můžeme vynechat bod *z2* a použít zápisy *z1* a *z3*. Následně přepíšeme do tvaru *z1* a *z2* a za bod *z3* napíšeme:

```
z1=(0,0);
z2=(20u,10u);
z3=(xpart z2,ypart z1);
```

Toto nám způsobí to, že v *x-ové* souřadnici se nám v bodě *z2* na délce *20u* vytvoří další bod, totéž u bodu *z1*, kde máme nulovou *x-ovou* a v tomto případě *y-onovou* souřadnici. Poté příkazem `draw` spojíme tento obrazec a pomocí příkazu `scaled` nadefinujeme jakým perem budeme konkrétní obrázek kreslit (více v další podkapitole **Pero**). Jedná se podle zápisu souřadnic bodů o pravoúhlý trojúhelník, kde jeho bod *z2*, resp. bod *C* je v pravém úhlu (více v doplňujících informací).

Pero

Bez pera by nám byly naše nadefinované body úplně k ničemu. Proto METAFONT obsahuje základní kreslící nástroj a to kreslící pero.

Při kreslení a tvorbě obrázků nebo písma v METAFONTu máme hned několik nástrojů. Základní druhy pera jsou 3 typy:

- **pencircle** – pero, které je v kruhovém tvaru. Toto pero se používá nejčastěji.
- **pensquare** - pero, které je ve čtvercovém tvaru, doporučuje se pro vykreslování ostrých pravoúhlých rohů.
- **penrazor** – pero ve tvaru úsečky, vhodné pro deformaci a rotaci pera pro vznik tzv. kaligrafického efektu.

Pro vybrání konkrétního pera musíme použít v METAFONTu příkaz `pickup` a za tento příkaz uvést jeden ze tří typů pera a poté příkaz `scaled`, který nám vyjadřuje tloušťku-průměr kreslících čar, kde za příkaz tento parametr napíšeme. Pokud však hodnotu `scaled` vynecháme METAFONT si s tímto poradí a nastaví tloušťku čar na *0,4 pt*.

Například pokud chceme pero v kruhovém tvaru o průměru 2 mm musíme jej vypsat takto:

```
pickup pencircle scaled 2mm;
```

(Pokud ovšem nemáme tuto hodnotu v záhlaví dokumentu, ale máme zde též nadefinované milimetry, stačí nám místo 2mm vypsat 2u, tímto si zkrátíme zápis o jedno písmenko :))

V METAFONTu je možné ukládat pera do proměnných, která následně můžeme jednoduše volat, a pracovat hned s několika pery najednou. Jedná se o proměnnou typu `pen`. Tato proměnná nám určitě v některých obrázcích ulehčí práci. Zde si na začátku obrázku navolíme pera, která budeme muset v budoucnu používat a poté je jednoduše zavoláme. Ukažme si to na příkladě:

Nadefinujeme si celkem 3 pera, která budeme v průběhu našeho kresleného obrázku volat. Toto uděláme následovně.

Vytvoříme si proměnnou typu `pen`, kde za tuto proměnnou napíšeme názvy nových per, v tomto příkladě jsme si je pojmenovali *pena, penb* a *penc*. Poté si jednotlivá pera nadefinujeme. Potom podle známého postupu zvolíme typ pera a jeho tloušťku.

```
pen pena,penb,penc;
pena:=pencircle scaled 5u;
penb:=pensquare scaled .5u;
penc:=pencircle scaled 10u;
pickup pena;
draw z1--z2;
pickup penb;
draw z2--z3;
pickup penc;
draw z3--z4;
pickup penc;
draw z1--z4;
```

Další příkazy pro keslení v METAFONTu, bez kterých se při práci neobejdeme:

draw

Vykreslí nám danou křivku, kterou si nadefinujeme.

drawdot

Tento příkaz nám slouží k zvýraznění bodu. Vybereme typ pera a jeho tloušťku a poté za pomocí příkazu `drawdot` aplikujeme na libovolný bod, který jsme si nadefinovali. Bod se zapíše za tento příkaz. Můžeme také bod zvýraznit libovolným perem, které jsme si vytvořili. Pokud chceme zvýraznit více bodů stejným perem o stejné tloušťce, oddělíme jednotlivé příkazy středníkem.

Například:

Vybrané pero a tloušťka hrotu:

```
pickup pencircle scaled 5u;
drawdot z1; drawdot z2; ... atd.
```

filldraw společně s cycle

Pokud bychom potřebovali daný obrázek vyplnit, použijeme příkaz `filldraw` společně s příkazem `cycle`. Příkaz cycle nám danou křivku uzavře a příkaz `filldraw` jí vyplní. Příkaz `filldraw` nahrazuje příkaz `draw`. Můžeme si to představit jako ve většině grafických programů jako je GIMP či Adobe Photoshop, kde nakreslíme například kružnici o daném průměru a tloušťky pera a do prázdného pole kružnice použijeme plechovku s barvou.

```
filldraw z1..z2..cycle;
```

fill společně s cycle

Další možností jak vyplníme uzavřenou křivku, ale tak, že tloušťka pera, kterou jí kreslíme zde nebude započtena, respektive s nulovou tlouštkou pera, je pomocí příkazu `fill`.

```
fill z1..z2..cycle;
```

erase draw

V nemálo případech bychom potřebovali nějakou část smazat (vygumovat). Příkazem `erase draw` je toto možné. Nejdříve si nadefinujeme pomocí příkazu `pickup` pencircle scaled; tloušťku gumy a poté pomocí příkazu `erase` a následného příkazu `draw` nadefinujeme, co a kde chceme smazat. Například jsme chtěli smazat přímku, která je vykreslená mezi bodem *z1* a *z2*. Vymažeme jí nasledovně:

```
pickup pencircle scaled 10u;
erase draw z1--z2;
```

Křivky

Bez křivek bychom se v METAFONTu rozhodně též neobešli. Pokud máme nadefinované body a chceme je jakkoliv spojit, tak k tomu slouží jakákoliv křivka. Definuje se v příkazu `draw` a má nespočet možností, jak si výsledný obrázek pomocí křivky vytvořit. Vyjmenuji proto jen pár základních příkazů.

Spojení pomocí „--“

Pro spojení, které je pomocí přímek, nám slouží příkaz:

```
draw z1--z2;
```

Spojení pomocí „..“

Tímto zápisem získáme křivku, která se propojí volně v konkrétních bodech, bude nám kopírovat kružnici.

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

Spojení pomocí spojení „--“ a „..“

Pokud chceme spojit obě varianty, ve spojení použijeme například tento tvar:

```
draw z1..z2--z3..cycle;
```

6.2.4 Kreslíme s METAFONTem

V této kapitole se budu zabývat tvorbou jednoduchých matematických útvarů, bez kterých se v mnoha případech neobejdeme. Jedná se v první řadě o přímku, která nám bude spojovat konkrétní dva body. Trojúhelník, kde si ukážeme propojení tří bodů. Čtverec a obdélník se čtyřmi body.

V neposlední řadě si vyzkoušíme nakreslení kružnice, půlkružnice a čtvrtkružnice, kde budeme vycházet ze dvou a více bodů.

V poslední řadě se naučíme a ukážeme spojování dvou a více bodů pomocí Beziérové křivky. Více řešených příkladů, které si můžete sami vyzkoušet, naleznete v příloze.

Přímka

Potřebujeme-li nakreslit přímku, tak jí nakreslíme například pomocí příkazu:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(1,30u#,10u#,0);
z1=(0,0);
z2=(30u,10u);
pickup pencircle scaled u;
draw z1--z2;
endchar;
end;
```

Výsledný obrázek:

Trojúhelník

Máme pravoúhlý trojúhelník, kde v bodě B je pravý úhel. Tento trojúhelník sestrojíme následovně:

Začneme tím, že vytvoříme soubor *trojuhelnik.mf*, kde napíšeme výše vypsané záhlaví a následně vypíšeme tyto příkazy k určení souřadnic bodů a vykreslení trojúhelníku na obrazovku:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(2,20u#,10u#,0);
z1=(0,0);
z2=(20u,0);
z3=(20u,10u);
pickup pencircle scaled u;
draw z1--z2--z3--z1;
endchar;
end;
```

U *z1*, který jsme nově napsali vyjadřuje počáteční stav bod *z1* o souřadnicích *(0,0)*, vyjadřuje bod *A*, kde x=0 a y=0. Když si představíme náčrt pomocí x a y souřadnic, tak je zřejmé, že *z2*, resp. bod *B* bude mít hodnotu *x-ovou 20u* a jelikož v tomto našem případě leží s rovinou s bodem *z2* resp. bodem *B*, tak musí mít hodnotu na *y-ové* souřadnici *0*, tedy *(20u,0)*. Totéž je i u bodu *z3*, resp. bodu *C*, kde jeho *x-ová* souřadnice nabývá hodnoty *20u* a *y-ová 10u*.

Pokud si takto nadefinujeme souřadnice, tak poté příkazem draw spojíme perem, které jsme nadefinovali výše pomoci příkazu *pickup pencircle scaled u;* (tedy jedná se o pero s kruhovým hrotem), body *z1, z2, z3* a zpět do *z1* pro uzavření trojúhelníku.

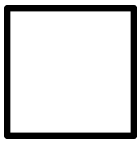
Výsledný obrázek:

Čtverec

Čtverec je další nepostradatelnou možností, jak nakreslit obrázek. Nakreslíme ho následovně:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(3,20u#,20u#,0);
z1=(0,0);
z2=(20u,0);
z3=(20u,20u);
z4=(0,20u);
pickup pensquare scaled u;
draw z1--z2--z3--z4--z1;
endchar;
end;
```

Výsledný obrázek:



Obdélník

Obdélník se zapisuje obdobně jako čtverec, díky jeho čtyřem stranám. Rozdíl je jen v jeho roměrech, kde musí být jedna strana obdélníku delší a jedna kratší. Můžeme obdélník zapsat do METAFONTu takto:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(4,20u#,20u#,0);
z1=(0,0);
z2=(20u,0);
z3=(20u,10u);
z4=(0,10u);
pickup pensquare scaled u;
draw z1--z2--z3--z4--z1;
endchar;
end;
```

Výsledný obrázek:

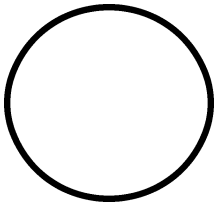
Co se stane když zaměníme příkaz *draw z1--z2--z3--z4--z1;*, který nám znázorňuje ostré propojení všech bodů za příkaz *draw z1..z2..z3..z4..z1;*. Z dřívější kapitoly **Kreslící nástroje** podkapitoly **Křivky** známe, že by se mělo jednat o propojení bodů kopírující kružnici. Tedy výsledným obrázkem je kružnice, kde její průměr je dán úhlopříčkou obdélníku.

```
⋮

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

⋮
```

Výsledný obrázek:



Kružnice

Nadefinujeme si 2 body *z1* a *z2*, kde *z1* je na počátečních bodech *(0,0)* a *z2* je na libovolných souřadnicích. Jde zde o vytvoření kružnice pomocí přímky, kde velikost přímky udává velikost průměru kružnice. Nakonec kružnici vyplníme příkazem `fill`.

`draw z1--z2;` na tento tvar: `draw z1..z2..z3;`

Například:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(5,5u#,5u#,0);
z1=(0,0);
z2=(10u,0);
pickup pencircle scaled u;
draw z1..z2..z1;
pickup pencircle scaled u;
fill z1..z2..cycle;
endchar;
end;
```

Výsledný obrázek:

Půlkružnice

Půlkružnici sestrojíme, v tomto případě jako v předchozím příkladě, pomocí dvou bodů o nějaké vzdálenosti, které nám udávají průměr kružnice, resp. půlkružnice. Musíme však ještě dopočítat poloměr kružnice, což víme, že je **0,5 * d** a vytvořit bod *z3*, který bude na *x-ové* souřadnici v polovině bodů *z1* a *z2* o jeho *y-onové* souřadnici též v poloměru, tedy polovině bodů *z1* a *z2*. Nesmíme zapomenout na fakt, že METAFONT si umí sám hodnoty dopočítat a v některých případech bychom obrázek velice těžko sestrojili. Pokud máme například hodnoty, které se dají dělit dvěma, tak si hodnoty můžeme sami vypočítat a doplnit za bod *z3*. Jde zde spíše o elegantnost zápisu. Proto jsem si připravil následující příklad.

Půlkružnice se tedy může sestrojit takto:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(6,5u#,5u#,0);
z1=(0,0);
z2=(9.31u,0);
z3=(0.5[x1,x2],0.5[x1,x2]);
% dopočítané ručně: z3=(4.655u, 4.655u);
pickup pencircle scaled u;
draw z1..z3..z2;
```

Výsledný obrázek:

Čtvrtkružnice

U čtvrtkružnice jsem si dovolil zcela nový zápis, pomocí něhož si ulehčíme práci. Tyto příkazy bych chtěl v závěru shrnout pro většinu vypsaných geometrických útvarů.

Prvním způsobem bylo nakreslení půlkružnice a pomocí příkazu `erase draw` vygumovat námi chtěnou část, v tomto případě jednu ze dvou částí půlkružnice.

Druhý způsob byl obdobný. Nakreslení plné kružnice, kde bych jednotlivé časti vymazal, tedy opět pomocí příkazu `erase draw`.

Oba dva způsoby se mi zdály naprosto nevhodnými, proto bych chtěl čtenáře seznámit i s několika příkazy, které v mnoha případech ulehčí uživateli METAFONTu práci.

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(7,10u#,10u#,0);
pickup pensquare scaled u;
draw quartercircle scaled 10u;
% velikost od středu je r=10u
endchar;
end;
```

Výsledný obrázek:

Křivka

Jako křivku jsem si připravil tvorbu sinusoidy. Možnost, kterou jsem si zvolil pro vytvoření sinusoidy bylo pomocí spojovací křívky, která neprotíná body rovnou čárou, ale tzv. obloukem. Tedy jedná se o spojení dvou a více bodů pomocí .. (dvou teček).

Po stránce obtížnosti není tento způsob obzvláště složitým, ale v jeho jednoduchosti se též skrývá jedna velmi negativní vlastnost. Tou vlastností je chápáno to, že pokud budeme požadovat 2 krát větší tuhost křivky, než je vzdálenost mezi jednotlivými body na ose x, způsobí nám tato možnost vytvoření sinusoidy v mírně deformativním vzhledu.

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(8,35u#,5u#,0);
z1=(-15u,5u);
z2=(-10u,0);
z3=(-5u,-5u);
z4=(0,0);
z5=(5u,5u);
z6=(10u,0);
```

```
z7=(15u,-5u);
z8=(20u,0);
z9=(25u,5u);
z10=(30u,0);
z11=(35u,-5u);
pickup pencircle scaled u;
draw z1..z2..z3..z4..z5..z6..z7..z8..z9..z10..z11;
endchar;
end;
```

Výsledný obrázek:

Další možnosti při kreslení v METAFONTu

V této kapitole jsme se naučili pomocí geometrických pravidel pochopit zápis v METAFONTu a po vyzkoušení a pochopení případně sami vytvářet námi požadované obrázky. Je zde však pár pravidel, které jsem nechal na úplný závěr této práce a to je ulehčení zápisu pomocí pevně definovaných pravidel v METAFONTu. Můžeme si toho všimnout na posledním zápisu čtvtkružnice, kde se objevuje nový námi neznámý příkaz `quartercircle`.

Mezi ulehčený zápis zahrnuji nakreslení čtverce, kružnice, půlkružnice a čtvrtkružnice.

Mezi tyto příkazy patří:

`unitsquare` - Čtverec, obdélník
`fullcircle` - Kružnice, elipsa
`halfcircle` - Půlkružnice
`quartercircle` - Čtvrtkružnice

Tyto příkazy nám urychlí a ulehčí práci při tvorbě obrázku. Tento způsob byl zmíněn v této kapitole u čtvrtkružnice.

Nejprve si nadefinujeme, jako v předchozích příkladech, typ pera a jeho tloušťku příkazem `pickup`. Poté napíšeme požadovaný průměr či velikost strany, podle toho, zda se jedná o kružnici nebo čtverec. Tuto hodnotu společně s příkazem požadovaného obrazce zapíšeme do příkazu `draw`.

Kapitola 7

Závěr

V této práci jsem sepsal a seznámil čtenáře s jednotlivými možnostmi práce s grafikou v LaTeXu. Zaměřil jsem se převážně na METAFONT, který je pro běžného uživatele počítače neznámým pojmem. U tvorby a úpravy grafiky, která se týká samotného LaTeXu, je nespočet kvalitních knižních i webových publikací, ohledně METAFONTu nikoliv. To byl další důvod, který mě směřoval touto cestou.

U LaTeXu bylo hlavním cílem čtenáři vypsat a poskytnout možnosti práce s grafikou. Práce s vektorovými a bitmapovými (rastrovými) obrázky, kde je aplikovat a kde nikoliv. Jaké jsou jejich formáty a jakými programy je případně upravit pro vložení do LaTeXu.

V další kapitole jsem čtenáře seznámil s METAFONTem. Seznamoval jsem pomocí příkladů, které jsem se snažil čtenáři postupně, podle složitosti, popsat a vysvětlit. Jako shrnutí těchto příkazů jsem na závěr této práce vytvořil výsledný obrázek.

Tato práce byla pro mě zajímavá, ale zároveň z úplného začátku při seznamování s METAFONTem složitá. Asi největším problémem, se kterým jsem se při psaní této práce potýkal, bylo nastavení METAFONTu. Publikace, které jsem přečetl jsou velice kvalitní, ale pro mě, který předtím ani nevěděl, že nějaký program METAFONT existuje, byly nepřehledné. Informace napsané v různých publikacích se navíc lišily a vzájemně si odporovaly. Nemálo mě toto od této práce odrazovalo. Nakonec jsem vše nastavil i spustil.

Tuto práci jsem popsal dle mých zkušeností s tímto programem. Předpokládám, že tato práce bude pro čtenáře a začínající METAFONTisty zábavná a hlavně užitečná.

Tuto práci naleznete též na www stránkách [12].

Literatura

[1] Rybička, Jiří. *LaTeX pro začátečníky*, 2. vydání, Brno: Konvoj, 2003, 238 stran, ISBN 80-7302-049-1

[2] Olšák, Petr. *Typografický systém TeX*. 2. vydání, Brno: Konvoj, 2000, ISBN 80-85615-91-6

[3] Šedivý, Přemysl. Brož, Miroslav. Gřondilová, Jana. Píše, Michal. Houfek, Karel. *Kreslíme METAFONTem*. [online] 1997 [citováno 2008-12-19]. Dostupné na web. str.: <http://www.cstug.cz/kreslime/kreslime.zip>, ISSN 1213-8185

[4] Horák, Karel. *Můj zápas s METAFONTem aneb pérovky a jiná zvěrstva*. [online] 1991 [citováno 2008-12-19]. Dostupné na web. str.: <http://bulletin.cstug.cz/pdf/bul913.pdf>, ISSN 1211-6661

[5] Lukeš, Vladimír. *METAFONT - Jak kreslit obrázky*. [online] 1999 [citováno 2008-12-19]. Dostupné na web. str.: <http://www.kiv.zcu.cz/~herout/html_sbo/metafont/toc.htm>.

[6] Brabec, Stanislav. *Grafika v UNIXu - PostScript*. [online] 2001 [citováno 2009-03-26]. Dostupné na web. str.: <http://www.root.cz/clanky/grafika-v-unixu-ix-postscript/>, ISSN 1212-8309, citace.

[7] Švamberg, Michal. *Jak na LaTeX*. [online] 2003 [citováno 2009-01-06]. Dostupné na web. str.: <http://www.root.cz/clanky/jak-na-latex-graphicx-comment>, ISSN 1212-8309, citace.

[8] DUBEC, Jakub. *Seriál webdesign - 3. část.* [online] 2006 [citováno 2009-03-26]. Dostupné na web. str.: <http://programujte.com/index.php?akce=clanek&cl=2006120403-serial-webdesign-3-cast>, ISSN 1801-1586, citace.

[9] RYBIČKA, Jiří. *LaTeX pro začátečníky*, 2. vydání, Brno: Konvoj, 2003, 238 stran, ISBN 80-7302-049-1, citace ze str. 99

[10] WEBER, Martin. *Program pro konvertování bitmapových souborů na vektorové.* [online] 2002 [citováno 2008-12-29]. Dostupné na web. str.: <http://autotrace.sourceforge.net>.

[11] KOPKA, Helmut. DALY, Patrick W. *LaTeX kompletní průvodce.* 1. vydání, Brno: Computer press, 2004, ISBN 80-722-6973-9, citace ze str. 435

[12] BOUČEK, Pavel. *Práce s grafikou v LaTeXu.* [online] 2009 [citováno 2009-04-11]. Dostupné na web. str.: <http://metafont.ic.cz>.

[13] OTAKAR *Program pro konvertování bitmapových souborů na vektorové.* [online] 2002 [citováno 2008-12-29]. Dostupné na web. str.: <http://www.abclinuxu.cz/software/grafika/vektory/autotrace>, ISSN 1214-1267.

[14] RŮŽIČKA. *Vložení METAFONTu do LaTeXu, balíček mflogo.* [online] 2002 [citováno 2009-01-04]. Dostupné na web. str.: <http://www.fi.muni.cz/∼xruzick7/cviceni-pb029>.

[15] GRANDSIRE, Christophe. *The METAFONTtutorial.* [online] 2004 [citováno 2009-01-06]. Dostupné na web. str.: <http://metafont.tutorial.free.fr/downloads/mftut.pdf>.

Přílohy

Příloha A

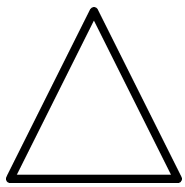
Doplňující obrázky v METAFONTu

A.1 Ukázky běžného zápisu

A.1.1 Rovnoramenný trojúhelník

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(9,20u#,20u#,0);
z1=(0,0);
z2=(20u,0);
z3=(0.5[x1,x2],20u);
pickup pencircle scaled u;
draw z1--z2--z3--z1;
endchar;
end;
```

Výsledný obrázek:



A.1.2 Rovnostranný trojúhelník

Zde je potřeba spočítat výšku pokud ji neznáme, tu dosadíme do *z3=(x3,y3)* jako parametr za *y3*.

Výšku spočítáme podle vzorce:

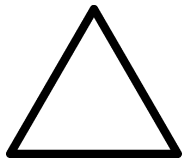
$$v = 0.5 * \sqrt{3} * a$$

Odmocnina se značí v METAFONTu `sqrt`.

Rovnostranný trojúhelník bude vypadat následovně:

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(10,20u#,20u#,0);
z1=(0,0);
z2=(20u,0);
z3=(0.5[x1,x2],0.5*sqrt3*20u);
pickup pencircle scaled u;
draw z1--z2--z3--z1;
endchar;
end;
```

Výsledný obrázek:



A.1.3 Trojúhelník pomocí xpart a ypart

```
mode_setup;
u#:=1mm#;
define_pixels(u);
screenstrokes;
beginchar(11,30u#,30u#,0);
z1=(0,0);
z2=(30u,10u);
z3=(xpart z2,ypart z1);
pickup pencircle scaled u;
draw z1--z2--z3--z1;
endchar;
end;
```

Výsledný obrázek:

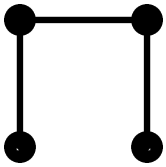
A.1.4 Čtverec

V tomto příkladě jsem připravil čtverec, kde jeho všechny 4 strany jsou vyznačeny body a mezi body *z1* a *z2* nejsou propojeny. Je to díky příkazu `undraw z1--z2`, čímž se přímka mezi těmito body smazala.

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(12,25u#,25u#,0);
z1=(0,0);
z2=(20u,0);
z3=(20u,20u);
z4=(0,20u);
pickup pencircle scaled u;
draw z1--z2--z3--z4--z1;
pickup pencircle scaled 5u;
drawdot z1;
drawdot z2;
```

```
drawdot z3;
drawdot z4;
pickup pensquare scaled u;
undraw z1--z2;
endchar;
end;
```

Výsledný obrázek:

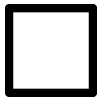


A.2 Ukázky zjednodušeného zápisu

A.2.1 Příkaz `unitsquare` - Čtverec

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(13,10u#,10u#,0);
pickup pencircle scaled u;
draw unitsquare scaled 10u;
endchar;
end;
```

Výsledný obrázek:



A.2.2 Příkaz `unitsquare` - Obdélník

Tento příkaz můžeme použít jak u čtverce, tak i obdélníku. Jediné, co zde musíme pozměnit je zápis příkazu pro vykreslení `draw unitsquare scaled 10u;`. Příkaz bychom měli obohatit o jednu stranu velikostně rozdílnou. To uděláme pomocí příkazů `xscaled` a `yscaled`, kde první příkaz určuje velikost vodorovné a druhý příkaz velikost svislé strany obdélníku.

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(14,10u#,5u#,0);
pickup pencircle scaled u;
draw unitsquare xscaled 10u yscaled 5u;
endchar;
end;
```

Výsledný obrázek:

A.2.3 Příkaz `fullcircle` - Kružnice

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(15,10u#,10u#,0);
pickup pencircle scaled u;
draw fullcircle scaled 10u;
endchar;
end;
```

Výsledný obrázek:

A.2.4 Příkaz `fullcircle` - Elipsa

Pokud bychom potřebovali nakreslit elipsu, můžeme vycházet z příkazu kružnice doplněné o 2 příkazy. Jedná se o stejný způsob sestavení jako u sestrojení kružnice z příkazu `fullcircle`. Zde budeme muset též nastavit hodnoty, které nám určí výšku a šířku elipsy. Jsou to příkazy `xscaled` a `yscaled`.

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(16,10u#,5u#,0);
pickup pencircle scaled u;
draw fullcircle xscaled 10u yscaled 5u;
endchar;
end;
```

Výsledný obrázek:

A.2.5 Příkaz `halfcircle` - Půlkružnice

```
mode_setup;
u#:=0.2pt#;
define_pixels(u);
screenstrokes;
beginchar(17,10u#,10u#,0);
pickup pencircle scaled u;
draw halfcircle scaled 10u;
endchar;
end;
```

Výsledný obrázek:

Příloha B

Zdrojový kód výsledného obrázku

```
mode_setup;
u#:=0.2mm#;
define_pixels(u);
beginchar(18,42u#,34u#,0);
%---------------------------
%Obdélník, tvar baráku a jeho velikost
z1=(0,0);
z2=(30u,0);
z3=(30u,15u);
z4=(0,15u);
%---------------------------
%Římsa - obdélník pod okapem
z5=(-3u,15.5u);
z6=(33u,15.5u);
z7=(33u,15.8u);
z8=(-3u,15.8u);
%---------------------------
%Okap - zkosen z obou stran
z9=(-4u,15.8u);
z10=(34u,15.8u);
z11=(35u,16.2u);
z12=(-4.4u,16.2u);
%---------------------------
%Římsa - obdélník nad okapem
z13=(-3u,16.2u);
z14=(33u,16.2u);
z15=(33u,16.5u);
```

```
z16=(-3u,16.5u);
%--------------------------
%Střecha
z17=(15u,25u);
%--------------------------
%Okno levý roh - malé
z18=(5u,8u);
z19=(10u,8u);
z20=(10u,12u);
z21=(5u,12u);
%--------
%Výběžek1
z22=(4u,7u);
z23=(11u,7u);
z24=(11u,13u);
z25=(4u,13u);
%--------
%Parapet1
z26=(3u,6.5u);
z27=(12u,6.5u);
z28=(12u,7u);
z29=(3u,7u);
%--------------------------
%Okno pravý roh - velké
z30=(15u,8u);
z31=(25u,8u);
z32=(25u,12u);
z33=(15u,12u);
%--------
%Výběžek2
z34=(14u,7u);
z35=(26u,7u);
z36=(26u,13u);
z37=(14u,13u);
%--------
%Parapet2
z38=(13u,6.5u);
z39=(27u,6.5u);
z40=(27u,7u);
z41=(13u,7u);
%--------------------------
```

```
%Rozdělení velkého okna na 2 části
z42=(20u,8u);
z43=(20u,12u);
%---------------------------
%Okap 1. část levá strana čtyřúhelníku
z44=(-3.9u,14.5u);
z45=(-3.1u,14.9u);
z46=(-3.1u,15.8u);
z47=(-3.9u,15.8u);
%--------
%Okap 2. část levá strana trojúhelníku
z48=(-3.1u,14u);
%--------
%Okap 3. část levá strana čtyřúhelníku
z49=(-1.4u,13u);
z50=(-0.6u,13.4u);
%--------
%Okap 4. část levá strana trojúhelníku
z51=(-1.4u,13.8u);
%--------
%Okap 5. poslední část čtyřúhelníku
z52=(-1.4u,-0.4u);
z53=(-0.6u,-0.4u);
%---------------------------
%Komín
z54=(17u,20.9u);
z55=(18.4u,20.1u);
z56=(18.4u,27u);
z57=(17u,27u);
%--------
%Komín - horní část
z58=(16.7u,27u);
z59=(18.7u,27u);
z60=(18.7u,27.5u);
z61=(16.7u,27.5u);
%---------------------------
%Vyčnívající kmen stromu tvar obdélníku
z62=(-13u,-0.4u);
z63=(-11u,-0.4u);
z64=(-11u,3u);
z65=(-13u,3u);
```

```
%--------
%1. část - trojúhelník stromu od spodní části
z66=(-20u,3u);
z67=(-4u,3u);
z68=(-12u,7u);
%--------
%2. část - trojúhelník stromu
z69=(-18u,7u);
z70=(-6u,7u);
z71=(-12u,10u);
%--------
%3. část - trojúhelník stromu
z72=(-16u,10u);
z73=(-8u,10u);
z74=(-12u,12u);
%--------
%4. část - trojúhelník stromu
z75=(-14u,12u);
z76=(-10u,12u);
z77=(-12u,13u);
%---------------------------
%Slunce
z78=(-5u,30u);
z79=(0,30u);
%---------------------------
%Okap 1. část pravá strana čtyřúhelníku
z80=(33.1u,14.9u);
z81=(33.9u,14.5u);
z82=(33.9u,15.8u);
z83=(33.1u,15.8u);
%--------
%Okap 2. část pravá strana trojúhelníku
z84=(33.1u,14u);
%--------
%Okap 3. část pravá strana čtyřúhelníku
z85=(31.4u,13u);
z86=(30.6u,13.4u);
%--------
%Okap 4. část pravá strana trojúhelníku
z87=(31.4u,13.8u);
%--------
```

```
%Okap 5. poslední část čtyřúhelníku
z88=(31.4u,-0.4u);
z89=(30.6u,-0.4u);
%---------------------------
%1. schod dolní
z90=(31.4u,-0.4u);
z91=(35.4u,-0.4u);
z92=(35.4u,0.9u);
z93=(31.4u,0.9u);
%--------
%výběžek 1. schodu
z94=(31.4u,0.9u);
z95=(35.5u,0.9u);
z96=(35.6u,1u);
z97=(31.4u,1u);
%--------
%2. schod
z98=(31.4u,1u);
z99=(33.4u,1u);
z100=(33.4u,2.3u);
z101=(31.4u,2.3u);
%--------
%výběžek 2. schodu
z102=(31.4u,2.3u);
z103=(33.5u,2.3u);
z104=(33.6u,2.4u);
z105=(31.4u,2.4u);
%---------------------------
%Slunce - levé oko
z106=(-3.5u,30.5u);
z107=(-3u,30.5u);
%--------
%Slunce - pravé oko
z108=(-2u,30.5u);
z109=(-1.5u,30.5u);
%--------
%Slunce - levé oko, obočí
z110=(-3.6u,30.8u);
z111=(-2.9u,30.8u);
z112=(-3.25u,31.05u);
%--------
```

```
%Slunce - pravé oko, obočí
z113=(-2.1u,30.8u);
z114=(-1.4u,30.8u);
z115=(-1.75u,31.05u);
%--------
%Slunce - usměv
z116=(-3.6u,28.5u);
z117=(-1.4u,28.5u);
z118=(-2.5u,28u);
%--------------------------
%Mrak
z119=(24u,30u);
z120=(32u,28u);
z121=(34u,29u);
z122=(37u,26u);
z123=(40u,28u);
z124=(42u,30u);
z125=(38u,32u);
z126=(36u,34u);
z127=(34u,33u);
z128=(28u,34u);
%--------------------------
%Čára znázorňující zem
z129=(-20u,-0.5u);
z130=(42u,-0.5u);
%--------------------------
%čára o tloušce pera 1 pt
%--------
pickup pensquare scaled u;
%Obdélník
draw z1---z2---z3---z4---z1;
%Okno levý roh - malé
draw z18--z19--z20--z21--z18;
%Okno pravý roh - velké
draw z30--z31--z32--z33--z30;
%--------------------------
%čára o tloušce pera 0,2 pt
%--------
pickup pensquare scaled .2u;
%Římsa pod
draw z5---z6---z7---z8---z5;
```

```
%Okap
draw z9--z10--z11--z12--z9;
%Římsa nad
draw z13---z14---z15---z16---z13;
%Střecha vyplněná
fill z16--z15--z17--cycle;
%Okno levý roh
draw z22--z23--z24--z25--z22;
%Výběžek1
draw z18--z22;
draw z19--z23;
draw z20--z24;
draw z21--z25;
%Okno pravý roh
draw z34--z35--z36--z37--z34;
%Výběžek2
draw z30--z34;
draw z31--z35;
draw z32--z36;
draw z33--z37;
%Parapet1
draw z26--z27--z28--z29--z26;
%Parapet2
draw z38--z39--z40--z41--z38;
%Rozdělení velkého okna
draw z42--z43;
%Okap 1. část levá strana
draw z44--z45--z46--z47--z44;
%Okap 2. část levá strana
draw z44--z48--z45;
%Okap 3. část levá strana
draw z49-z50-z45-z48-z49;
%Okap 4. část levá strana
draw z49--z51;
%Okap 5. část levá strana
draw z49--z52--z53--z50;
%Komín
draw z55--z56--z57--z54;
%Komín - vršek
draw z58--z59--z60--z61--z58;
%Vyčnívající kmen stromu
```

```
fill z62--z63--z64--z65--z62--cycle;
%1. část - trojúhelník stromu
draw z66--z67--z68--z66;
%2. část - trojúhelník stromu
draw z69--z70--z71--z69;
%3. část - trojúhelník stromu
draw z72--z73--z74--z72;
%4. část - trojúhelník stromu
draw z75--z76--z77--z75;
%Okap 1. část pravá strana
draw z80--z81--z82--z83--z80;
%Okap 2. část pravá strana
draw z80--z84--z81;
%Okap 3. část pravá strana
draw z85--z86--z80--z84--z85;
%Okap 4. část pravá strana
draw z85--z87;
%Okap 5. část pravá strana
draw z85--z88--z89--z86;
%1. schod - dolní
draw z90--z91--z92--z93;
%Výběžek - 1. schod
draw z94--z95--z96--z97;
%2. schod - horní
draw z98--z99--z100--z101;
%Výběžek - 2. schod
draw z102--z103--z104--z105;
%Čára znázorňující zem
draw z129--z130;
%---------------------------
%Pero v kruhovém tvaru o velikosti 0,2 pt
%--------
pickup pencircle scaled .2u;
%Slunce
draw z78..z79..z78;
%1. mrak
draw z119..z120..z121..z122..z123..z124..z125
   ..z126..z127..z128..z119;
%---------------------------
%Pero v kruhovém tvaru o velikosti 0,1 pt
%--------
```

```
pickup pencircle scaled .1u;
%Slunce - levé oko
draw z106..z107..z106;
%Slunce - pravé oko
draw z108..z109..z108;
%Slunce - levé oko, obočí
draw z110..z112..z111;
%Slunce - pravé oko, obočí
draw z113..z115..z114;
%Slunce - úsměv
draw z116..z118..z117;
endchar; end;
```

B.1 Výsledný obrázek v METAFONTu