VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OBJECT DETECTION ON GPU

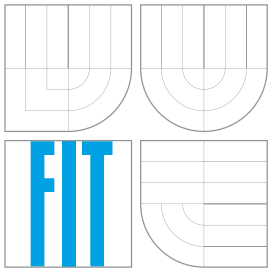
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL MACENAUER

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBJEKTŮ NA GPU

OBJECT DETECTION ON GPU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL MACENAUER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN JURÁNEK, Ph.D.

BRNO 2015

Abstrakt

Výtah (abstrakt) práce v českém jazyce.

Abstract

Výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Klíčová slova v českém jazyce.

Keywords

Klíčová slova v anglickém jazyce.

Citace

Pavel Macenauer: Object Detection on GPU, diplomová práce, Brno, FIT VUT v Brně, 2015

Object Detection on GPU

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana ...

.....
Pavel Macenauer
January 12, 2015

Poděkování

Zde je možné uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc.

Contents

1 GPGPU **2**
- 1.1 Parallel computing platforms . . . 3
- 1.2 NVIDIA CUDA . . . 4
  - 1.2.1 Programming model . . . 4
  - 1.2.2 Memory model . . . 5

2 Object detection **7**
- 2.1 AdaBoost . . . 7
- 2.2 Waldboost . . . 7
- 2.3 Features . . . 7
  - 2.3.1 LBP . . . 7

3 Implementation **8**
- 3.1 Memory organization . . . 8
- 3.2 Program structure . . . 8
- 3.3 Acceleration . . . 8

4 Results **9**
- 4.1 Summary . . . 9
- 4.2 Future work . . . 9
- 4.3 Musíme mít co říci . . . 9
- 4.4 Musíme vědět, komu to chceme říci . . . 9
- 4.5 Musíme si dokonale promyslet obsah . . . 10
- 4.6 Musíme psát strukturovaně . . . 10

5 Několik formálních pravidel **11**

6 Nikdy to nebude naprosto dokonalé **12**

7 Typografické a jazykové zásady **13**
- 7.1 Co to je normovaná stránka? . . . 14

8 Závěr **16**

Chapter 1

GPGPU

With high demand for real-time image processing, computer vision applications and a need for fast calculations in the scientific world, general-purpose computing on graphics processor units, also known as the GPGPU, has become a popular programming model to accelerate programs traditonally coded on the CPU (Central Processing Unit) using the data-parallel processing powers of the GPU.

Until the last decade or so, when technologies for GPGPU became available, the GPU was used mostly to render data given to it by the CPU. This has changed in a way, that the GPU, with its massive parallel capabilities, isn't used only for displaying, but also for computation. The traditional approach is to transfer data bidirectionally between the CPU and the GPU, which on one hand brings the overhead of copying the data, but on the other enables to do the calculations many times faster due to the architecture of the GPU. As shown on 1.1 many more transistors are dedicated to data processing instead of cache or control, which leads to a higher memory bandwidth.

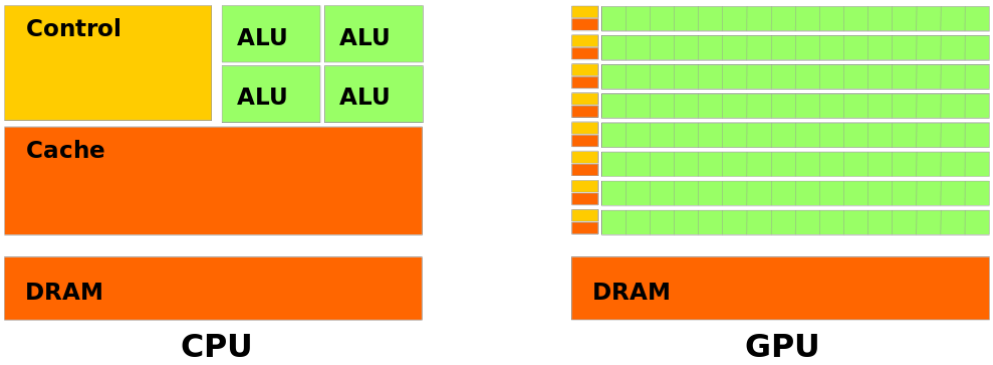


Figure 1.1: CPU and GPU architecture comparison ([**?**])

GPUs are also designed with demand for floating-point capabilities in mind, which can be taken advantage of in applications such as object detection, where most of the math is done in single-point arithmetic.

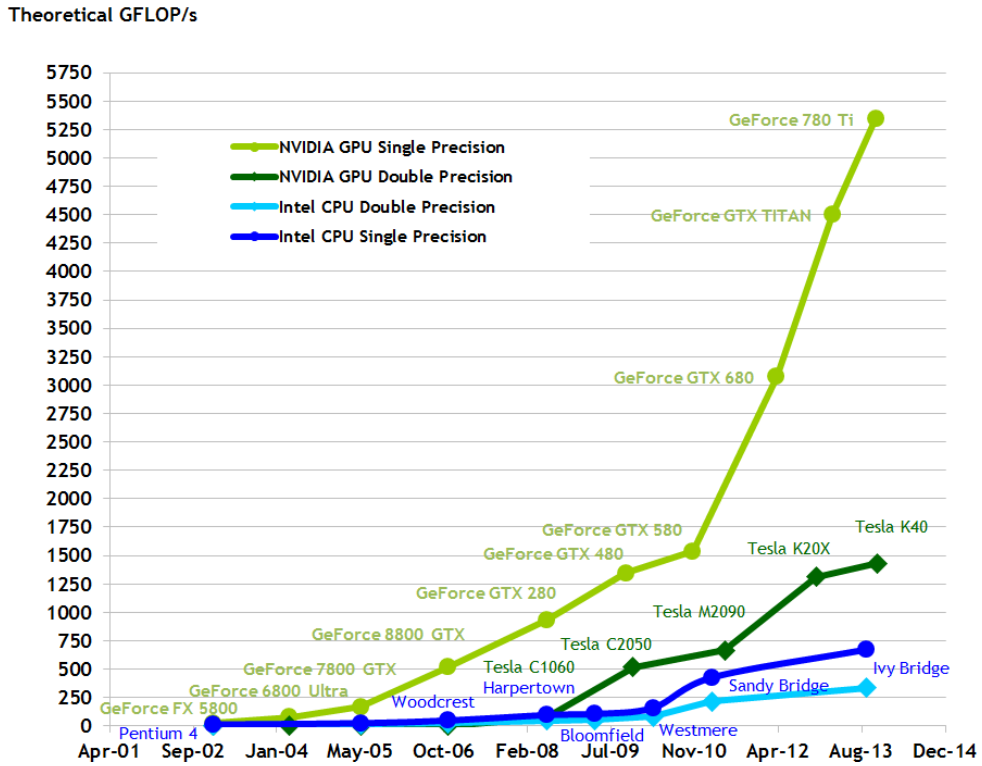


Figure 1.2: Floating-Point operations per second for the CPU and GPU ([?])

1.1 Parallel computing platforms

In November 2006 the first parallel computing platform - CUDA (Compute Unified Device Architecture) was introduced by NVIDIA. Since then several others were created by other vendors:

- CUDA - NVIDIA
- OpenCL - Khronos Group
- C++ AMP - Microsoft
- Compute shaders - OpenGL
- DirectCompute - Microsoft

All of the technologies above allow access to the GPU computing capabilities. The first two - CUDA and OpenCL work on a kernel basis. As a programmer you have access to low-level GPU capabilities and have to manage all the resources yourself. The standard approach is the following:

1. Allocate memory on the GPU
2. Copy data from the CPU to the allocated memory on the GPU
3. Run a GPU based kernel (written in CUDA or OpenCL)

4. Copy processed data back from the GPU to the CPU

C++ AMP is a more higher-level oriented library. Introduced by Microsoft as a new C++ feature for Visual Studio 2012 with STL-like syntax, it is designed to accelerate code using massive parallelism. Currently it is supported by most GPUs, which have a DirectX 11 driver.

The last two - Compute shaders and DirectCompute also work in a more high-level fashion, but also quite differently from C++ AMP. They are not a part of the rendering pipeline, but can be set to be executed among other OpenGL or DirectX shaders. The difference between compute shaders and other shaders is, that they don't have specified input or output. These must be specified by the programmer. Theoretically it is then possible to write the whole rendering pipeline using compute shaders only.

1.2 NVIDIA CUDA

NVIDIA CUDA is a programming model enabling direct access to the instruction set and memory of NVIDIA GPUs.

1.2.1 Programming model

CUDA C extends C and uses NVCC compiler to generate code for the GPU. It also allows to write C-like functions called kernels. A kernel is defined by the `__global__` declaration specifier and executed using a given configuration wrapped in `<<< ... >>>`. The configuration is called a grid and takes as parameters the number of blocks and the number of threads. The same kernel code is run by the whole grid. Also code run by the kernel is called to device code, where as the code run outside of the kernel is called the host code.

Threads are a basic computational unit identified by a 3-dimensional id `threadIdx`, which is typically used to index arrays.

Blocks are groups of threads, where each block resides on a single processor core, therefore a kernel can be run with the maximum of 1024 threads.

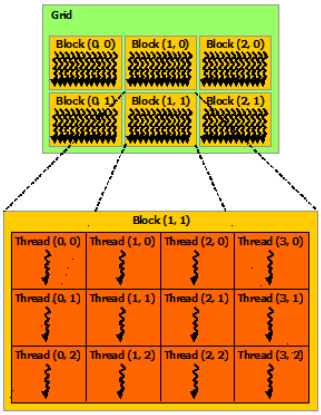


Figure 1.3: A grid of blocks and threads run by a kernel ([**?**])

Kernel configuration parameters can be passed as integers or `dim3` structures. `dim3` specifies the number of threads or blocks in every dimension, therefore a `dim3 threadsPerBlock(4,4,1)` would run a kernel with 16 threads per block, where `threadIdx.x` would range between 0 and 3 and the same for `threadIdx.y`.

Example 1.4 shows how to add 2 arrays in parallel using N threads and 1 block.

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

Figure 1.4: Example of vector addition in CUDA ([**?**])

1.2.2 Memory model

CUDA threads may access the following types of memories:

Global memory is generally the slowest memory type. Its acccess is the bottleneck for most applications with access latency ranging from 400 to 800 cycles.

Constant memory is read-only type memory. Access can be even slower than global memory, on the other hand broadcast enables very short latency and high bandwidth when all the threads access the same location.

Unified memory is a memory type introduced in CUDA 6.0. It enables to use the same memory addresses both in host and device code, which simplifies writing code. On the other as of spring 2014, there doesn't seem to be any hardware support [**?**] and the unified memory performs very similar to global memory.

Memory	Keyword	Scope	Lifetime
Local memory	-	Thread	Kernel
Shared memory	`__shared__`	Block	Kernel
Global memory	-	Grid	Application
Constant memory	`__constant__`	Grid	Application

Table 1.1: Memory types

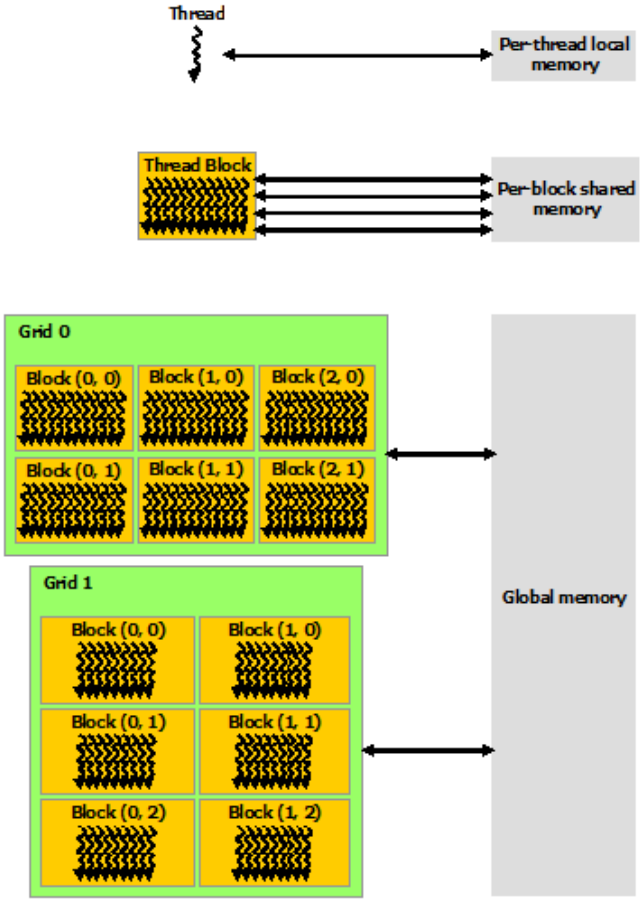


Figure 1.5: Memory hierarchy ([**?**])

Chapter 2

Object detection

2.1 AdaBoost

2.2 Waldboost

2.3 Features

2.3.1 LBP

Chapter 3

Implementation

3.1 Memory organization

3.2 Program structure

3.3 Acceleration

Chapter 4

Results

4.1 Summary

4.2 Future work

Abychom mohli napsat odborný text jasně a srozumitelně, musíme splnit několik základních předpokladů:

- Musíme mít co říci,
- musíme vědět, komu to chceme říci,
- musíme si dokonale promyslet obsah,
- musíme psát strukturovaně.

Tyto a další pokyny jsou dostupné též na školních internetových stránkách [**?**].

Přehled základů typografie a tvorby dokumentů s využitím systému LaTeXje uveden v [2].

4.3 Musíme mít co říci

Dalším důležitým předpokladem dobrého psaní je *psát pro někoho*. Píšeme-li si poznámky sami pro sebe, píšeme je jinak než výzkumnou zprávu, článek, diplomovou práci, knihu nebo dopis. Podle předpokládaného čtenáře se rozhodneme pro způsob psaní, rozsah informace a míru detailů.

4.4 Musíme vědět, komu to chceme říci

Dalším důležitým předpokladem dobrého psaní je psát pro někoho. Píšeme-li si poznámky sami pro sebe, píšeme je jinak než výzkumnou zprávu, článek, diplomovou práci, knihu nebo dopis. Podle předpokládaného čtenáře se rozhodneme pro způsob psaní, rozsah informace a míru detailů.

4.5 Musíme si dokonale promyslet obsah

Musíme si dokonale promyslet a sestavit obsah sdělení a vytvořit pořadí, v jakém chceme čtenáři své myšlenky prezentovat. Jakmile víme, co chceme říci a komu, musíme si rozvrhnout látku. Ideální je takové rozvržení, které tvoří logicky přesný a psychologicky stravitelný celek, ve kterém je pro všechno místo a jehož jednotlivé části do sebe přesně zapadají. Jsou jasné všechny souvislosti a je zřejmé, co kam patří.

Abychom tohoto cíle dosáhli, musíme pečlivě organizovat látku. Rozhodneme, co budou hlavní kapitoly, co podkapitoly a jaké jsou mezi nimi vztahy. Diagramem takové organizace je graf, který je velmi podobný stromu, ale ne řetězci. Při organizaci látky je stejně důležitá otázka, co do osnovy zahrnout, jako otázka, co z ní vypustit. Příliš mnoho podrobností může čtenáře právě tak odradit jako žádné detaily.

Výsledkem této etapy je osnova textu, kterou tvoří sled hlavních myšlenek a mezi ně zařazené detaily.

4.6 Musíme psát strukturovaně

Musíme začít psát strukturovaně a současně pracujeme na co nejsrozumitelnější formě, včetně dobrého slohu a dokonalého značení. Máme-li tedy myšlenku, představu o budoucím čtenáři, cíl a osnovu textu, můžeme začít psát. Při psaní prvního konceptu se snažíme zaznamenat všechny své myšlenky a názory vztahující se k jednotlivým kapitolám a podkapitolám. Každou myšlenku musíme vysvětlit, popsat a prokázat. Hlavní myšlenku má vždy vyjadřovat hlavní věta a nikoliv věta vedlejší.

I k procesu psaní textu přistupujeme strukturovaně. Současně s tím, jak si ujasňujeme strukturu písemné práce, vytváříme kostru textu, kterou postupně doplňujeme. Využíváme ty prostředky DTP programu, které podporují strukturovanou stavbu textu (předdefinované typy pro nadpisy a bloky textu).

Chapter 5

Několik formálních pravidel

Naším cílem je vytvořit jasný a srozumitelný text. Vyjadřujeme se proto přesně, píšeme dobrou češtinou (nebo zpravidla angličtinou) a dobrým slohem podle obecně přijatých zvyklostí. Text má upravit čtenáři cestu k rychlému pochopení problému, předvídat jeho obtíže a předcházet jim. Dobrý sloh předpokládá bezvadnou gramatiku, správnou interpunkci a vhodnou volbu slov. Snažíme se, aby náš text nepůsobil příliš jednotvárně používáním malého výběru slov a tím, že některá zvlášť oblíbená slova používáme příliš často. Pokud používáme cizích slov, je samozřejmým předpokladem, že známe jejich přesný význam. Ale i českých slov musíme používat ve správném smyslu. Např. platí jistá pravidla při používání slova *zřejmě*. Je *zřejmé* opravdu zřejmé? A přesvědčili jsme se, zda to, co je *zřejmé* opravdu platí? Pozor bychom si měli dát i na příliš časté používání zvratného se. Například obratu *dokázalo se*, že… zásadně nepoužíváme. Není špatné používat autorského *my*, tím předpokládáme, že něco řešíme, nebo například zobecňujeme spolu se čtenářem. V kvalifikačních pracích použijeme autorského *já* (například když vymezujeme podíl vlastní práce vůči převzatému textu), ale v běžném textu se nadměrné používání první osoby jednotného čísla nedoporučuje.

Za pečlivý výběr stojí i symbolika, kterou používáme ke *značení*. Máme tím na mysli volbu zkratek a symbolů používaných například pro vyjádření typů součástek, pro označení hlavních činností programu, pro pojmenování ovládacích kláves na klávesnici, pro pojmenování proměnných v matematických formulích a podobně. Výstižné a důsledné značení může čtenáři při četbě textu velmi pomoci. Je vhodné uvést seznam značení na začátku textu. Nejen ve značení, ale i v odkazech a v celkové tiskové úpravě je důležitá důslednost.

S tím souvisí i pojem z typografie nazývaný *vyznačování*. Zde máme na mysli způsob sazby textu pro jeho zvýraznění. Pro zvolené značení by měl být zvolen i způsob vyznačování v textu. Tak například klávesy mohou být umístěny do obdélníčku, identifikátory ze zdrojového textu mohou být vypisovány `písmem typu psací stroj` a podobně.

Uvádíme-li některá fakta, neskrýváme jejich původ a náš vztah k nim. Když něco tvrdíme, vždycky výslovně uvedeme, co z toho bylo dokázáno, co teprve bude dokázáno v našem textu a co přebíráme z literatury s uvedením odkazu na příslušný zdroj. V tomto směru nenecháváme čtenáře nikdy na pochybách, zda jde o myšlenku naši nebo převzatou z literatury.

Nikdy neplýtváme čtenářovým časem výkladem triviálních a nepodstatných informací. Neuvádíme rovněž několikrát totéž jen jinými slovy. Při pozdějších úpravách textu se nám může některá dříve napsaná pasáž jevit jako zbytečně podrobná nebo dokonce zcela zbytečná. Vypuštění takové pasáže nebo alespoň její zestručnění přispěje k lepší čitelnosti práce! Tento krok ale vyžaduje odvahu zahodit čas, který jsme jejímu vytvoření věnovali.

Chapter 6

Nikdy to nebude naprosto dokonalé

Když jsme už napsali vše, o čem jsme přemýšleli, uděláme si den nebo dva dny volna a pak si přečteme sami rukopis znovu. Uděláme ještě poslední úpravy a skončíme. Jsme si vědomi toho, že vždy zůstane něco nedokončeno, vždy existuje lepší způsob, jak něco vysvětlit, ale každá etapa úprav musí být konečná.

Chapter 7

Typografické a jazykové zásady

Při tisku odborného textu typu *technická zpráva* (anglicky *technical report*), ke kterému patří například i text kvalifikačních prací, se často volí formát A4 a často se tiskne pouze po jedné straně papíru. V takovém případě volte levý okraj všech stránek o něco větší než pravý – v tomto místě budou papíry svázány a technologie vazby si tento požadavek vynucuje. Při vazbě s pevným hřbetem by se levý okraj měl dělat o něco širší pro tlusté svazky, protože se stránky budou hůře rozevírat a levý okraj se tak bude oku méně odhalovat.

Horní a spodní okraj volte stejně veliký, případně potištěnou část posuňte mírně nahoru (horní okraj menší než dolní). Počítejte s tím, že při vazbě budou okraje mírně oříznuty.

Pro sazbu na stránku formátu A4 je vhodné používat pro základní text písmo stupně (velikosti) 11 bodů. Volte šířku sazby 15 až 16 centimetrů a výšku 22 až 23 centimetrů (včetně případných hlaviček a patiček). Proklad mezi řádky se volí 120 procent stupně použitého základního písma, což je optimální hodnota pro rychlost čtení souvislého textu. V případě použití systému LaTeX ponecháme implicitní nastavení. Při psaní kvalifikační práce se řiďte příslušnými závaznými požadavky.

Stupeň písma u nadpisů různé úrovně volíme podle standardních typografických pravidel. Pro všechny uvedené druhy nadpisů se obvykle používá polotučné nebo tučné písmo (jednotně buď všude polotučné nebo všude tučné). Proklad se volí tak, aby se následující text běžných odstavců sázel pokud možno na *pevný rejstřík*, to znamená jakoby na linky s předem definovanou a pevnou roztečí.

Uspořádání jednotlivých částí textu musí být přehledné a logické. Je třeba odlišit názvy kapitol a podkapitol – píšeme je malými písmeny kromě velkých začátečních písmen. U jednotlivých odstavců textu odsazujeme první řádek odstavce asi o jeden až dva čtverčíky (vždy o stejnou, předem zvolenou hodnotu), tedy přibližně o dvě šířky velkého písmene M základního textu. Poslední řádek předchozího odstavce a první řádek následujícího odstavce se v takovém případě neoddělují svislou mezerou. Proklad mezi těmito řádky je stejný jako proklad mezi řádky uvnitř odstavce.

Při vkládání obrázků volte jejich rozměry tak, aby nepřesáhly oblast, do které se tiskne text (tj. okraje textu ze všech stran). Pro velké obrázky vyčleňte samostatnou stránku. Obrázky nebo tabulky o rozměrech větších než A4 umístěte do písemné zprávy formou skládanky všité do přílohy nebo vložené do záložek na zadní desce.

Obrázky i tabulky musí být pořadově očíslovány. Číslování se volí buď průběžné v rámci celého textu, nebo – což bývá praktičtější – průběžné v rámci kapitoly. V druhém případě se číslo tabulky nebo obrázku skládá z čísla kapitoly a čísla obrázku/tabulky v rámci kapitoly – čísla jsou oddělena tečkou. Čísla podkapitol nemají na číslování obrázků a tabulek žádný vliv.

Tabulky a obrázky používají své vlastní, nezávislé číselné řady. Z toho vyplývá, že v odkazech uvnitř textu musíme kromě čísla udat i informaci o tom, zda se jedná o obrázek či tabulku (například "... *viz tabulka 2.7* ..."). Dodržování této zásady je ostatně velmi přirozené.

Pro odkazy na stránky, na čísla kapitol a podkapitol, na čísla obrázků a tabulek a v dalších podobných příkladech využíváme speciálních prostředků DTP programu, které zajistí vygenerování správného čísla i v případě, že se text posune díky změnám samotného textu nebo díky úpravě parametrů sazby. Příkladem takového prostředku v systému LaTeX je odkaz na číslo odpovídající umístění značky v textu, například návěští (`\ref{navesti}` – podle umístění návěští se bude jednat o číslo kapitoly, podkapitoly, obrázku, tabulky nebo podobného číslovaného prvku), na stránku, která obsahuje danou značku (`\pageref{navesti}`), nebo na literární odkaz (`\cite{identifikator}`).

Rovnice, na které se budeme v textu odvolávat, opatříme pořadovými čísly při pravém okraji příslušného řádku. Tato pořadová čísla se píší v kulatých závorkách. Číslování rovnic může být průběžné v textu nebo v jednotlivých kapitolách.

Jste-li na pochybách při sazbě matematického textu, snažte se dodržet způsob sazby definovaný systémem LaTeX. Obsahuje-li vaše práce velké množství matematických formulí, doporučujeme dát přednost použití systému LaTeX.

Mezeru neděláme tam, kde se spojují číslice s písmeny v jedno slovo nebo v jeden znak – například *25krát*.

Členicí (interpunkční) znaménka tečka, čárka, středník, dvojtečka, otazník a vykřičník, jakož i uzavírací závorky a uvozovky se přimykají k předcházejícímu slovu bez mezery. Mezera se dělá až za nimi. To se ovšem netýká desetinné čárky (nebo desetinné tečky). Otevírací závorka a přední uvozovky se přimykají k následujícímu slovu a mezera se vynechává před nimi – (takto) a "takto".

Pro spojovací a rozdělovací čárku a pomlčku nepoužíváme stejný znak. Pro pomlčku je vyhrazen jiný znak (delší). V systému TeX (LaTeX) se spojovací čárka zapisuje jako jeden znak "pomlčka" (například "Brno-město"), pro sázení textu ve smyslu intervalu nebo dvojic, soupeřů a podobně se ve zdrojovém textu používá dvojice znaků "pomlčka" (například "zápas Sparta – Slavie"; "cena 23–25 korun"), pro výrazné oddělení části věty, pro výrazné oddělení vložené věty, pro vyjádření nevyslovené myšlenky a v dalších situacích (viz Pravidla českého pravopisu) se používá nejdelší typ pomlčky, která se ve zdrojovém textu zapisuje jako trojice znaků "pomlčka" (například "Další pojem — jakkoliv se může zdát nevýznamný — bude neformálně definován v následujícím odstavci."). Při sazbě matematického mínus se při sazbě používá rovněž odlišný znak. V systému TeX je ve zdrojovém textu zapsán jako normální mínus (tj. znak "pomlčka"). Sazba v matematickém prostředí, kdy se vzoreček uzavírá mezi dolary, zajistí vygenerování správného výstupu.

Lomítko se píše bez mezer. Například školní rok 2008/2009.

Pravidla pro psaní zkratek jsou uvedena v Pravidlech českého pravopisu [1]. I z jiných důvodů je vhodné, abyste tuto knihu měli po ruce.

7.1 Co to je normovaná stránka?

Pojem *normovaná stránka* se vztahuje k posuzování objemu práce, nikoliv k počtu vytištěných listů. Z historického hlediska jde o počet stránek rukopisu, který se psal psacím strojem na speciální předtištěné formuláře při dodržení průměrné délky řádku 60 znaků a při 30 řádcích na stránku rukopisu. Vzhledem k zápisu korekturních značek se používalo řádkování 2 (ob jeden řádek). Tyto údaje (počet znaků na řádek, počet řádků a proklad mezi nimi) se nijak

nevztahují ke konečnému vytištěnému výsledku. Používají se pouze pro posouzení rozsahu. Jednou normovanou stránkou se tedy rozumí 60*30 = 1800 znaků. Obrázky zařazené do textu se započítávají do rozsahu písemné práce odhadem jako množství textu, které by ve výsledném dokumentu potisklo stejně velkou plochu.

Orientační rozsah práce v normostranách lze v programu Microsoft Word zjistit pomocí funkce *Počet slov* v menu *Nástroje*, když hodnotu *Znaky (včetně mezer)* vydělíte konstantou 1800. Do rozsahu práce se započítává pouze text uvedený v jádru práce. Části jako abstrakt, klíčová slova, prohlášení, obsah, literatura nebo přílohy se do rozsahu práce nepočítají. Je proto nutné nejdříve označit jádro práce a teprve pak si nechat spočítat počet znaků. Přibližný rozsah obrázků odhadnete ručně. Podobně lze postupovat i při použití OpenOffice. Při použití systému LaTeX pro sazbu je situace trochu složitější. Pro hrubý odhad počtu normostran lze využít součet velikostí zdrojových souborů práce podělený konstantou cca 2000 (normálně bychom dělili konstantou 1800, jenže ve zdrojových souborech jsou i vyznačovací příkazy, které se do rozsahu nepočítají). Pro přesnější odhad lze pak vyextrahovat holý text z PDF (např. metodou cut-and-paste nebo *Save as Text...*) a jeho velikost podělit konstantou 1800.

Chapter 8

Závěr

Závěrečná kapitola obsahuje zhodnocení dosažených výsledků se zvlášť vyznačeným vlastním přínosem studenta. Povinně se zde objeví i zhodnocení z pohledu dalšího vývoje projektu, student uvede náměty vycházející ze zkušeností s řešeným projektem a uvede rovněž návaznosti na právě dokončené projekty.

Bibliography

[1] Kolektiv autorů. *Pravidla českého pravopisu.* Academia, 2005. ISBN 80-200-1327-X.

[2] Jiří Rybička. *LaTeX pro začátečníky.* Konvoj, 1999. ISBN 80-85615-77-0.