Žilinská univerzita v Žiline
Spoločnosť pre otvorené informačné technológie

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie
OSSConf 2016

29. júna–1. júla 2016
Žilina, Slovensko

Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
29. júna–1. júla 2016, Žilina, Slovensko

Vedeckí garanti konferencie:

prof. Ing. Karol Matiaško, PhD., Žilinská univerzita, Žilina

Vedecký a programový výbor:

prof. Ing. Miloš Šrámek, PhD., Austrian Academy of Sciences, Wien (AUT) – **predseda**
prof. Oleg Černojarov, DrSc., Moscow Power Engineering Institute (RU)
doc. Ing. Karol Grondžák, PhD., Žilinská univerzita, SOIT, Žilina
doc. RNDr. Štefan Peško, CSc., Žilinská univerzita, SOIT, Žilina
doc. Ing. Jiří Rybička, Dr., MZLU Brno (CZ)
RNDr. Rudolf Blaško, PhD., Žilinská univerzita, SOIT, Žilina
Mgr. Peter Czimmermann, PhD., Žilinská univerzita, SOIT, Žilina
Mgr. Michal Kaukič, CSc., Žilinská univerzita, SOIT, Žilina
RNDr. Aleš Kozubík, PhD., Žilinská univerzita, SOIT, Žilina
Dr inż. Remigiusz Olejnik, Zachodniopomorski Uniwersytet Technologiczny w Szczecinie (PL)
Ing. Pavel Stříž, Ph.D., Nakladatelství Martin Stříž, Bučovice (CZ)
RNDr. Ladislav Ševčovič, PhD., FEI, Technická Univerzita Košice, SOIT, Košice
Ing. Jiří Eischman, Red Hat CZ, SOIT, Brno (CZ)

Organizačný výbor:

Aleš Kozubík, Žilinská univerzita, SOIT, Žilina – **predseda**
Rudolf Blaško, Žilinská univerzita, SOIT, Žilina
Michal Kaukič, Žilinská univerzita, SOIT, Žilina
Michal Chovanec, Žilinská univerzita, SOIT, Žilina
Tomáš Majer, Žilinská univerzita, SOIT, Žilina
Miloslav Ofúkaný, GeoCommunity, SOIT, Bratislava
Pavel Stříž, Nakladatelství Martin Stříž, Bučovice
Peter Štrba, Gymnázium Mikuláša Galandu, SOIT, Turčianske Teplice

Vydavateľ: Žilinská univerzita v Žiline

ISBN 978-80-554-1292-4

Žilinská univerzita v Žiline

Spoločnosť pre otvorené informačné technológie

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH

Zborník príspevkov medzinárodnej konferencie
OSSConf 2016

29. júna–1. júla 2016
Žilina, Slovensko

Recenzenti:

Blaško, Rudolf, RNDr., PhD., Grondžák, Karol, doc., Ing., PhD., Kaukič, Michal, Mgr., CSc., Kovalík Štefan, RNDr., PhD., Kozubík, Aleš, RNDr., PhD., Kvaššay Miroslav,Ing., PhD., Peško, Štefan, doc., RNDr., PhD., Rybička Jiří, doc., Ing., Dr., PhDr. Miroslav Sedlák, Stříž, Pavel, Ing., Ph.D., Zaitseva Elena, doc., Ing., PhD.

Editori:

Rudolf Blaško, Aleš Kozubík

Všetky práce, uverejnené v zborníku, boli posúdené dvomi nezávislými recenzentmi.

Za jazykovú úroveň zodpovedajú autori príspevkov. Jazykovou korektúrou pod vedením PhDr. Miroslava Sedláka prešli iba príspevky zo sekcie Open GIS.

Obsah

Úvod 5

Luboš Balážovič, Lukáš Mužla
Fyzickogeografická charakteristika územia s využitím nástroja QGIS 7

Miroslav Biňas, Dominik Juhás, Dominik Matta
Aplikácia metodík Objects first a Design patterns first so softvérovým rámcom GameLib 13

Rudolf Blaško
LaTeX a zrkadlo sadzby 21

Zuzana Borčinová
Rozdeľuj a počítaj s MPI 29

Zdena Dobešová
Vizuální programování v QGIS 35

Tomáš Hála
Tabulky v ConTeXTu: přístupy, možnosti, algoritmy 43

Michal Chovanec, Peter Šarafín
Adaptive sparse distributed memory as function approximator 61

Aleš Kozubík
Ako som objavil knižnicu tikzDevice 69

Miroslav Kvaššay
Investigation of Errors in Virtual Model of Human Body using Blender 77

Dominika Maršáleková, Miroslav Biňas, Martin Sarnovský
Tvorba vlastných aplikácií pre interaktívne tabule 85

Remigiusz Olejnik
GNU Radio toolkit and its application in the area of RF signal processing 93

Jan Přichystal
Usnadnění tvorby dokumentů v aplikaci TeXonWeb 99

Jan Růžička, Vojtěch Lhotský
ElasticSearch pro prostorová data 105

Jiří Rybička
Jednoduché není jednoduché – řádkový proklad pod lupou 117

Pavel Stříž
Užití TEXu a Lua při sazbě knihy – průvodce světem Arduina 125

Pavel Stříž
Arduino mi pomáhá na divadle: vstupujeme do světa světelného designu .. 133

Peter Šarafín, Róbert Žalman, Michal Chovanec, Veronika Olešnaníková
Interactive outdoor game based on NFC "Find The Tree" 143

Abstrakty nerecenzovaných ukážok a prezentácií 149

Vážení čitatelia,

dostáva sa vám do rúk zborník, ktorý je výstupom v poradí už ôsmej samostatnej konferencie, venovanej slobodnému a otvorenému softvéru a jeho využitiu vo vzdelávaní, vede a ostatných oblastiach, ktoré si vyžadujú IT riešenia. Hoc v poradí ôsma, ostáva naša konferencia stále prvou a jedinou konferenciou na Slovensku, venovanou otvoreným technológiám.

Tento ročník konferencie prináša tradičné sekcie „LaTeX a jeho priatelia“, „Open GIS“, „OSS vo vede a vzdelávaní“ a taktiež „Open Hardware“, ktorá sa sľubne rozvíja. Sekcia „LaTeXa jeho priatelia“ má svoje pevné miesto na všetkých ročníkoch a dlhodobo patrí medzi príspevkovo najbohatšie sekcie. Taktiež sekcia „Open GIS“ nechýbala ani na jednom ročníku OSSConf. Treťou z tradičných sekcií je „OSS vo vede a vzdelávaní“, ktorá je asi obsahovo najrôznorodejšia. Relatívne nová, ale svoje pevné miesto si budujúca, je sekcia „Open Hardware“, ktorá na rozdiel od predchádzajúcich je venovaná otvorenému hardvéru.

Touto cestou by som sa chcel osobitne poďakovať vedeniu Fakulty radenia a informatiky ŽU zastúpenému osobou jej dekana pána doc. Ing. Emila Kršáka, PhD., za dlhoročnú podporu našej konferencie a bezodplatné poskytnutie konferenčných miestností a laboratórií v priestoroch fakulty.

Na záver chcem popriať všetkým účastníkom veľa poučenia a príjemných zážitkov v komunite priaznivcov otvoreného softvéru a otvorených technológií.

Za organizačný výbor OSSConf2016

Aleš Kozubík
predseda

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 7–12

FYZICKOGEOGRAFICKÁ CHARAKTERISTIKA ÚZEMIA S VYUŽITÍM NÁSTROJA QGIS

ĽUBOŠ BALÁŽOVIČ (SK) A LUKÁŠ MUŽLA (SK)

Abstrakt. Výskum v každej oblasti vyžaduje svoje nástroje. V geografii sú predmetom výskumu priestorové vzťahy, ich identifikácia a interpretácia. V prírodných vedách ako je geografia, geológia, enviromentalistika a biológia patrí medzi časté úlohy charakteristika študovaného územia a nadväzujúce priestorové analýzy. Cieľom tohto článku je na príklade analýzy územia povodia Tajovského potoka ukázať, že tieto analytické úlohy voľne šíriteľný a otvorený geografický informačný systém QGIS plne pokrýva a môže tak slúžiť nielen geoinformatikom, ale aj odborníkov z rôznych iných disciplín, ktorí potrebujú jednoducho a rýchlo spracovať charakteristiku vybraného územia.

Kľúčové slová. QGIS, geografická charakteristika územia, GIS, tvorba máp.

PHYSICAL GEOGRAPHIC CHARACTERISTICS USING QGIS

Abstract. Research of each science requires its tools. In geography the object of research identification, research and interpretation of spatial relationships. In natural sciences such as geography, geology, environmental science and biology the most common tasks are geographical characteristics and spatial analyses of the studied area. The paper is an example how the area of Tajovský stream basin analyses could be easly done using QGIS.

Keywords. QGIS, geography characteristics, GIS, map creation.

Úvod

QGIS[1] je v súčasnosti najsofistikovanejší otvorený geografický informačný systém (GIS). Hoci stále existujú oblasti, kde proprietárny softvér ponúka riešenie, ktoré prostredníctvom otvorených GISov nie je plnohodnotne nahraditeľné (viď. napr. [5]), možno jednoznačne konštatovať, že bežný prírodovedne zameraný používateľ je schopný svoje potreby pokryť funkcionalitou, ktorú ponúka QGIS.

V nasledujúcom texte sme vymedzili niekoľko základných máp, ktoré sú potrebné pri fyzickogeografickej charakteristike územia, a s ktorými sa bežne stretneme v prácach geografov, krajinných ekológov a biológov. Uvádzame aj stručný popis použitých údajov a nástrojov v rámci QGISu, pomocou ktorých je možné sa k nim dopracovať.

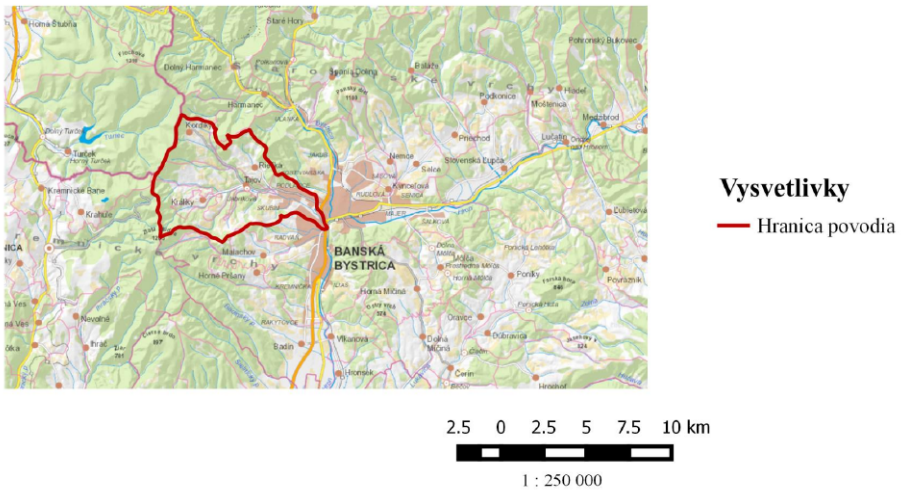
[1] `http://qgis.org`.

1. Vymedzenie územia

Mapu vymedzenia územia je možné realizovať pomocou digitalizácie nad existujúcim podkladom alebo pridaním vektorovej vrstvy so záujmovým územím (podrobnejšie pozri [1,3]). Ako podkladovú mapu možno použiť vrstvu zo služby WMS (vrstva zo ZB GIS[2]), alebo s využitím modulu zásuvného modulu OpenLayers[3] je k dispozícii výber medzi podkladom Google Maps/Satellite/Hybrid, Bing Map a OpenStreetMaps. Pri vymedzení územia treba dbať na vhodný súradnicový systém vektorovej vrstvy. Po pridaní vrstvy z externých služieb, poskytovaných cez OpenLayers sa automaticky zmení súradnicový systém projektu.

Výslednú mapu je možné finalizovať v nástroji *Tvorba mapových výstupov* doplnením legendy, číselnej a grafickej mapovej mierky a ďalšími mapovými prvkami. Mapu je následne možné vyexportovať ako obrázok vo vysokom rozlíšení alebo PDF (v tomto prípade sú niektoré elementy uložené ako vektory). Príklad výslednej mapy je zobrazený na obr. 1.

Nevýhodou takto zostavenej mapy je absencia legendy prvkov z podkladovej PDF. V súčasnosti síce vo WMS špecifikácii existuje nepovinný dopyt na WMS server s požiadavkou `GetLegendGraphic` avšak reálne ho nebolo možné použiť, kvôli chýbajúcej podpore našich WMS serverov a tiež chybe v QGIS-e[4].



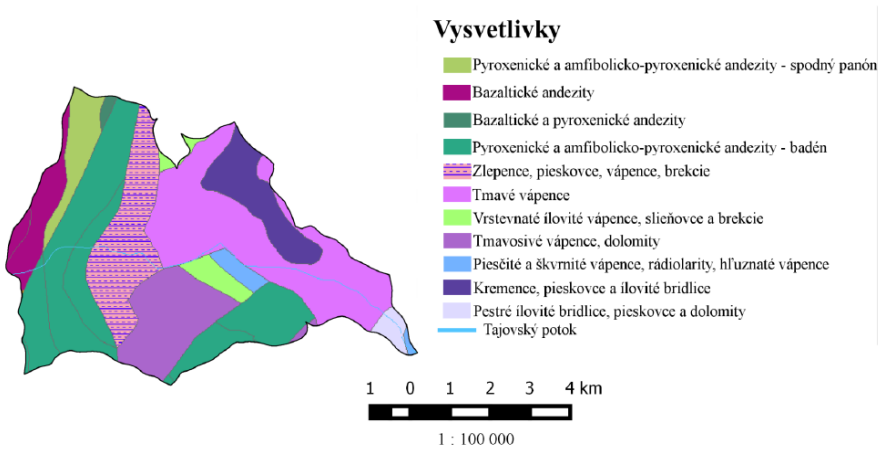
Obr. 1. Mapa vymedzenia územia Tajovského potoka

[2] `https://www.geoportal.sk/sk/sluzby/mapove-sluzby/wms/wms-zbgis.html`.
[3] `https://plugins.qgis.org/plugins/openlayers_plugin/`.
[4] `https://hub.qgis.org/issues/12610`.

2. Čiastkové fyzickogeografické mapy

Podobným spôsobom ako sme popísali v predchádzajúcej sekcii je možné vytvoriť aj rôzne ďalšie mapy charakterizujúce fyzickogeografickú stránku vybraného územia. Zobraziť len záujmové územie v hraniciach daných vektorovou polygónovou vrstvou možno jednoduchou zmenou štýlu tejto vrstvy na štýl *Inverted polygons*, ktorý spôsobí, že bude danou farbou vykresľované všetko územie, okrem samotného polygónu, ktorý ostane prázdny. Tak je možné maskovať všetko nežiadané územie a ponechať len záujmové územie (ukážka výslednej mapy na obr. 2). Týmto spôsobom sme zostavili nasledujúce mapy:

- Mapa geologického podkladu – podklad k nej je voľne k dispozícii ako WMS služba Geologického ústavu Dionýza Štúra (GÚDŠ)[5].



Obr. 2. Mapa – geologický podklad

- Mapa geomorfologického členenia – voľne k dispozícii ako WMS služba GÚDŠ[6].
- Pôdna mapa – z Výskumného ústavu pôdoznalectva a ochrany pôdy dostupná ako WMS služba[7].
- Mapa klimatických oblastí, mapa potenciálnej vegetácie a ďalšie sa nachádzajú v ArcGIS Services Directory Slovenskej agentúry životného prostredia, avšak nie sú dostupné cez službu WMS, preto sme sa k týmto údajom dostali len prostredníctvom exportu z pôvodnej DVD GIS aplikácie vydanej spolu s Atlasom krajiny SR.

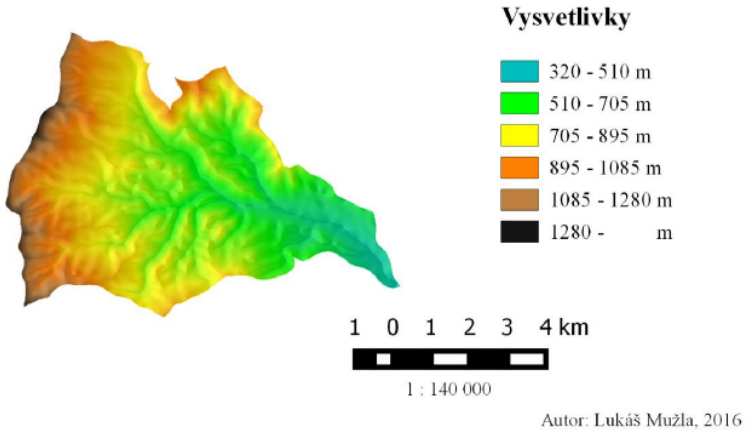
[5] `http://mserver.geology.sk:8399/arcgis/services/WMS/GMSR1000/MapServer/WMSServer`.
[6] `http://mserver.geology.sk:8399/arcgis/services/WMS/GMC/MapServer/WMSServer`.
[7] `http://sscri.vupop.sk/arcgis/services/vupop_wms/MapServer/WMSServer?`.

- Mapa krajinnej pokrývky Corine Landcover (SAŽP) dostupná ako WMS službu[8].

3. Interpolácia rastra výšok z nepravidelného bodového poľa

Vzhľadom na to, že digitálny model reliéfu DMR3 nie je voľne prístupný vo formáte, ktorý by umožňoval ďalšie analýzy, väčšina užívateľov volí cestu jednoduchej interpolácie terénu z vrstevníc z mapového diela ZM 50 alebo ZM 10. Na vytvorenie digitálneho modelu reliéfu (DMR) sme využili interpolačnú metódu RST – regularizovaný splajn s tenziou. Tá je dostupná v QGISe len ako modul GRASSu `v.surf.rst` dostupný cez zásuvný modul GRASSu, alebo cez nástroj *Processing Toolbox* (menu Processing). Vstupné dáta tvorila digitalizovaná vektorová vrstva vrstevníc zo ZM50. Rozlíšenie výsledného rastra bolo nastavené na 8x8 metrov. Pôvodné nastavenia tenzie a zhladenia bolo potrebné upraviť tak, aby sa na interpolovanom povrchu a na z neho odvodených rastroch sklonov a orientácií nenachádzali žiadne anomálie (viac v [2]). Nakoľko niektoré chyby bolo možné odstrániť len doplnením ďalších vstupných bodov, použili sme aj bodové pole namerané cez geodetický GNSS[9] rover priamo v teréne.

DMR sme ďalej pomocou mapovej algebry a modulu `r.mapcalc` prenásobili rastrovou vrstvou územia povodia a získali tak DMR vymedzený len územím Tajovského potoka (obr. 3).



Obr. 3. Mapa digitálneho modelu reliéfu (metóda RST)

[8] `http://nipi.sazp.sk/arcgis/services/atlassr/atlassr_05_rest/MapServer/WMSServer`.
[9] GNSS – globálny satelitný navigačný systém.

4. Vybrané priestorové analýzy

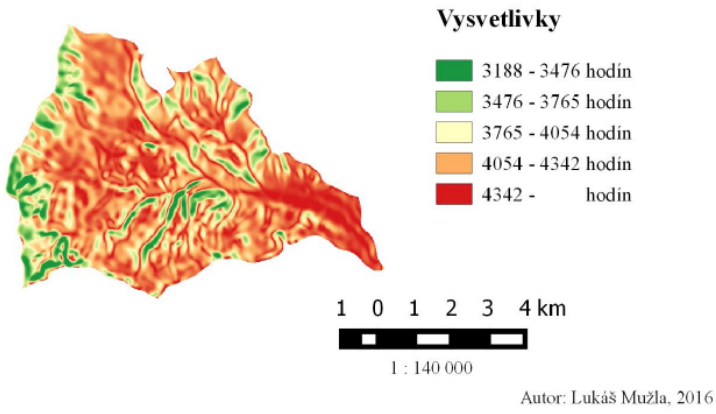
4.1. Výpočet morfometrických parametrov

Priamo z DMR je možné odvodiť raster orientácie terénu voči svetovým stranám a raster sklonov v smere spádnice cez modul GRASSu `r.slope.aspect`. Práve tvorba týchto rastrov nám umožnila objaviť nedostatky v DMR získaným metódou RST a jeho spätné pregenerovanie s inými nastaveniami.

4.2. Analýza oslnenia

Slnečné žiarenie ovplyvňuje množstvo prírodných procesov, ale vplýva aj na ľudskú aktivitu (poľnohospodárstvo, lesníctvo, energetika). Pri priestorovej analýze územia je preto potrebné poznať priestorovú a časovú distribúciu slnečnej radiácie. Modul GRASSu `r.sun` je len teoretickým modelom, ktorý berie do úvahy reliéf a sklon dopadu slnečných lúčov, zanedbáva však oblačnosť, vlastnosť povrchu absorbovať, či odraziť slnečné žiarenie a mnohé iné. Napriek tomu výsledná mapa môže byť výraznou pomocou pri plánovaní poľnohospodárstva, lokalizácii ekologických solárnych panelov, či určovaní oblastí, kde sa na jar bude najrýchlejšie topiť snehová pokrývka a môže tak vzrásť povodňové riziko.

Vstupom pre modelovanie sú tri rastrové vrstvy: digitálny model reliéfu, raster sklonu reliéfu v smere spádnice a raster orientácie reliéfu voči svetovým stranám. Kombináciou týchto vstupných dát sa pri modelovaní zohľadní zatienenie reliéfu. Výstupom je raster celkového slnečného žiarenia, ktoré dopadne na záujmové územie za jeden rok a raster dĺžky času slnečnej radiácie na povodie za jeden rok (obr. 4).



Obr. 4. Mapa dĺžky oslnenia povodia Tajovského potoka

5. Záver

Podarilo sa nám ukázať, že otvorený softvér QGIS a voľný prístup ku GIS údajom vytvárajú efektívny nástroj na geografickú charakteristiku územia využiteľnú v prírodných vedách, technických vedách ale aj vo verejnej správe. Vzhľadom na obmedzený priestor sme prezentovali len niekoľko vybraných ukážkových analýz, avšak analogicky by bolo možné preskúmať aj ďalšie vlastnosti územia (viditeľnosť z vybraných bodov, hydrologickú charakteristiku územia, vymedzenie subpovodí, štatistické zhodnotenie krajinnej pokrývky atď.). Alternatívne by bolo možné použiť aj iné otvorené GIS napr. uDig, OpenJump alebo GRASS GIS. Oproti softvéru QGIS však poskytujú menší užívateľský komfort (zložitejšie používateľské rozhranie, obmedzené nástroje pri finalizácii mapového tlačového výstupu alebo v nich absentuje podpora slovenského jazyka). Poslednou možnosťou je využitie niektorého mapového portálu (napr. mapový klient ZB GIS) umožňujúceho aj generovanie máp z externých WMS služieb, v súčasnosti však neposkytuje výstupy v kvalite porovnateľnej s QGIS riešením.

Prekážky použitia tvorí ešte stále nie celkom dobudovaná otvorená infraštruktúra geopriestorových informácií, zvlášť implementácia kontextuálnej legendy dostupnej cez WMS.

Poďakovanie. Tento príspevok vznikol s prispením grantu APVV-15-0050.

Literatúra

[1] BALÁŽOVIČ, Ľ.: *Spracovanie a analýza geopriestorových údajov*, Banská Bystrica: Univerzita Mateja Bela v Banskej Bystrici, 2015, ISBN 978-80-557-0953-6, 150 s.

[2] GALLAY, M.: , Košice: Univerzita Pavla Jozefa Šafárika v Košiciach, 2015, ISBN 978-80-8152-289-5, 118 s.

[3] KAŇUK, J.: *Priestorové analýzy a modelovanie*, Košice: Univerzita Pavla Jozefa Šafárika v Košiciach, 2015, ISBN 978-80-8152-290-1, 114 s.

[4] MUŽLA, L.: *Priestorová analýza povodia Tajovského potoka v GIS*, bakalárska práca, UMB, 2016, 54 s.

[5] GISGeography: *27 Differences Between ArcGIS and QGIS – The Most Epic GIS Software Battle in GIS History*, GIS Geography august 2015 [cit 13 jún 2016], `http://gisgeography.com/qgis-arcgis-differences/`.

Kontaktné adresy

Mgr. Ľuboš Balážovič, PhD., Katedra geografie a geológie, Fakulta prírodných vied, Univerzita Mateja Bela, Tajovského 40, 974 01 Banská Bystrica, Slovenská Republika,
E-mailová adresa: `lubos.balazovic@umb.sk`, `http://www.fpv.umb.sk/lbalazovic/`

Lukáš Mužla, Katedra geografie a geológie, Fakulta prírodných vied, Univerzita Mateja Bela, Tajovského 40, 974 01 Banská Bystrica, Slovenská republika,
E-mailová adresa: `lukas.muzla@studenti.umb.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf 2016: 13–20

APLIKÁCIA METODÍK OBJECTS FIRST A DESIGN PATTERNS FIRST SO SOFTVÉROVÝM RÁMCOM GAMELIB

MIROSLAV BIŇAS (SK), DOMINIK JUHÁS (SK) A DOMINIK MATTA (SK)

Abstrakt. Tento príspevok sa venuje metodikám výučby objektového programovania *objects-first* a *design patterns first* pomocou softvérového rámca *GameLib*. Opisuje jeho nové vlastnosti, experimentálne overenie, ako aj výhody, ktoré rámec priniesol v podobe zvýšenej motivácie u študentov.

Kľúčové slová. Objektové programovanie, výučba, object-first.

APPLICATION OF METHODS OBJECTS FIRST AND DESIGN PATTERNS FIRST WITH FRAMEWORK GAMELIB

Abstract. This article is dedicated to the methods of teaching object programming *objects-first* and *design patterns first* with framework *GameLib*. It describes its new features, experimental verification, as well as benefits that the framework increased motivation in students.

Keywords. Object-oriented programming, teaching, object-first.

Úvod

Objektové programovanie je v súčasnom modernom softvérovom priemysle jedným zo základných stavebných pilierov moderných aplikácií. Predstavuje *paradigmu programovania*, resp. *metodiku*, ktorá si vyžaduje odlišnú filozofiu v porovnaní napr. so *štruktúrovaným programovaním*. Je súčasťou mnohých technológií a zároveň spĺňa všetky požiadavky na znovupoužitie kódu a jeho jednoduchšie udržiavanie.

Spôsob výučby objektového programovania taktiež podlieha vývoju a neustále sa mení a zlepšuje. Kým kedysi bolo objektové programovanie považované za vyšší stupeň programovania a jeho výučba bola dostupná len pre študentov vyšších ročníkov, dnes sa dostáva aj do nižších stupňov štúdia. Je dokonca bežné, že sa študenti s touto paradigmou stretávajú v prvých ročníkoch na univerzitách a nezriedka si základy osvoja už na stredných školách.

Prechod zo štruktúrovaného programovania na objektové však nebýva najjednoduchší. Aj keď sa stále jedná o programovanie, objektový prístup vo veľkej miere súvisí s návrhom a architektúrou problému, čo so sebou prináša mnoho abstrakcie, ktorá bez dostatočných a zrozumiteľných praktických ukážok dokáže spoľahlivo demotivovať každého študenta. Niektoré prístupy a niektorí autori sa

napr. snažia obe paradigmy kombinovať, aby si ich študenti osvojili naraz a vnímali problém ako kompozíciu objektov so svojimi vlastnosťami a správaním.

Jedným z najväčších problémov, s ktorým ako učitelia pri výučbe programovania zápasíme, je udržanie pozornosti študentov. Okrem voľby konkrétnej metodiky výučby, vhodného jazyka a technológií, je nesmierne dôležité dbať aj na správnu motiváciu študentov, ktorá má vo výučbe nesmierne veľkú silu. K dosiahnutiu tohto cieľa môže pomôcť vyučovanie programovania pomocou tvorby hier, ktoré ťaží z jednoduchého cieľa, ktorým je vytvorenie vlastnej hry iteratívnym prístupom. Jasná vízia a viditeľné výsledky práce po každej iterácii zabezpečia, že jeho účastníci zostávajú motivovaní počas celého kurzu, pričom svoj aktuálny stav si vedia kedykoľvek overiť [5].

Tento spôsob výučby programovania môže pomôcť zefektívniť a modernizovať kurzy programovania nielen na vysokých školách, ale o to viac na stredných a základných školách. Túto metodiku sme úspešne aplikovali vo výučbe programovania aj na *Katedre počítačov a informatiky* na *Technickej univerzite v Košiciach* už pred niekoľkými rokmi a v tomto článku opíšeme aktualizáciu softvérového rámca, ktorý používame vo výučbe predmetu *Objektové programovanie*.

1. Skúsenosti s aplikáciou metodiky Objects-first

Metodika *objects-first* sa stala populárnou vďaka nástroju *BlueJ*[1], ktorý vytvoril *Michael Kölling* a prakticky prezentoval v knihe *Objects First with Java* [1]. U nás sa za najväčšieho propagátora tohto prístupu považuje *Rudolf Pecinovský*, ktorý tento prístup prezentoval v niekoľkých článkoch ako aj knihách [6].

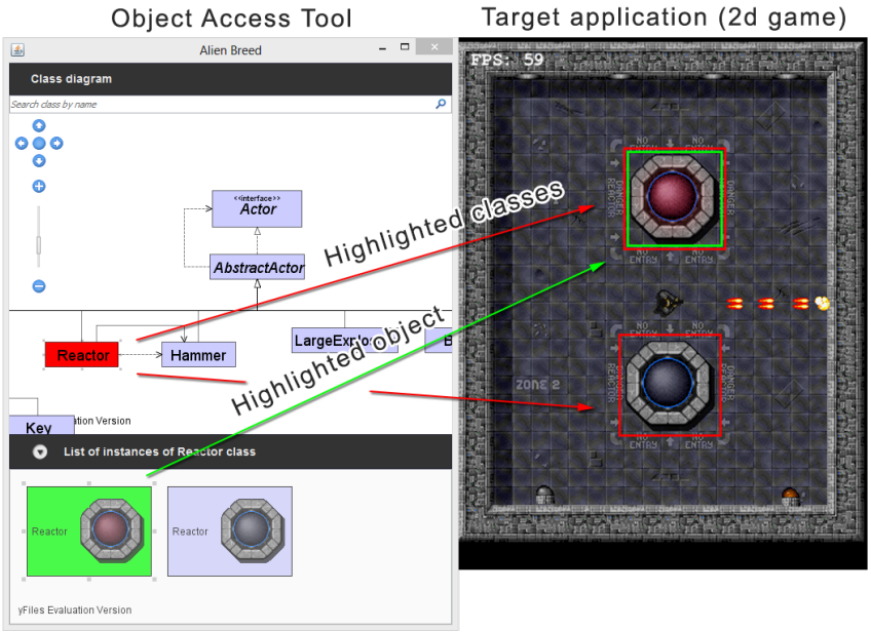
Na predmetoch ponúkaných *Katedrou počítačov a informatiky* pri *Technickej univerzite v Košiciach* sme s aplikáciou prístupu *objects-first* v predmete *Objektové programovanie* začali okolo roku 2008. Inšpiráciou sa stal príklad z knihy [1] s názvom *The World of Zuul*, ktorý predstavuje jednoduchú textovú hru. Na základe tohto príkladu sme vytvorili jednoduchý rámec s názvom *Indiana Jones Project* a sadu iteratívnych cvičení [2], na konci ktorých boli študenti schopní vytvoriť vlastné dobrodružstvo s využitím princípov objektového programovania.

Keďže sa nám tento prístup značne osvedčil, inovovali sme ho a na základoch vytvoreného rámca pre textovku sme vytvorili rámec s názvom *GameLib*, ktorý bol tentokrát postavený na hernej knižnici *Slick2D*[2]. Okrem samotného rámca sme však vytvorili nástroj *Inšpektor*, ktorý sa podobá na nástroje obsiahnuté v prostredí *BlueJ*. Tento nástroj kombinuje diagram tried a tzv. *Object bench*, v ktorom sa nachádza zoznam vytvorených inštancií (ukážka tohto nástroja sa nachádza na obrázku 1). Vďaka tomuto nástroju sme boli schopní doslova začať s objektami ešte skôr, ako študenti začali programovať [4].

[1] `http://bluej.org/`

[2] `http://slick.ninjacave.com/`

Študenti v tomto prípade iteratívnym spôsobom vytvárali remake hry *Alien Breed*[3] známu z počítača *Amiga*.



Obr. 1. Nástroj Inšpektor spolu s 2D hrou ako cieľovou aplikáciou [4]

S odstupom času je možné konštatovať, že motivácia študentov vďaka práci na grafickej hre, stúpla. V procese prípravy materiálov sme však museli bojovať s tým, že sa jednalo o akčný typ hry. To nám neumožnilo rozprávať komplexnejšie príbehy, ako tomu bolo v prípade adventúry. Taktiež sme boli značne obmedzení sprajtami, z ktorých bola táto hra postavená. Práve tieto dva faktory a aj fakt, že knižnicu *Slick2D* je možné v súčasnosti považovať za mŕtvy projekt, nás viedlo k inovácii scenárov, ako aj samotného vytvoreného vzdelávacieho rámca.

2. Softvérový rámec GameLib v2.0

Ako bolo uvedené vyššie, rámec *GameLib*, ktorý vznikol na základe rámca pre textovú adventúru *Indiana Jones Project*, už používame vo výučbe niekoľko rokov. Pre jeho plánovanú aktualizáciu sme teda vychádzali zo skúseností získaných počas jeho používania.

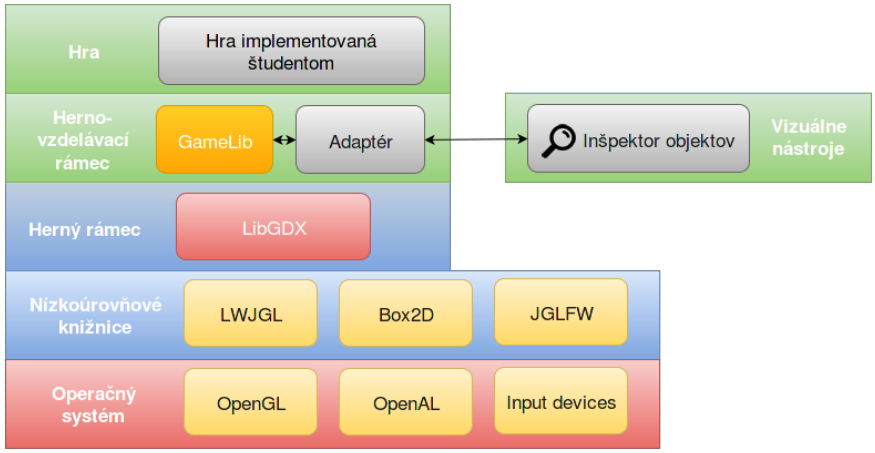
Vo všeobecnosti je možné konštatovať, že rámec sa nám osvedčil a jeho API sme chceli meniť len minimálne. Vytvorené API je dostatočne čisté a jednoduché

[3] `https://en.wikipedia.org/wiki/Alien_Breed`

a čo je najpodstatnejšie - neodpútava zbytočne študentov od objektových princípov technologickými detailmi. Zmenu si však vyžadoval herný rámec *Slick2D*, na ktorom je náš rámec založený, pretože v priebehu 3 rokov vyšli len 2 aktualizácie (posledná pred vyše rokom). Ako vhodná alternatíva sa nám ponúkol herný rámec *LibGDX*[4], v ktorom je v súčasnosti vytvorených mnoho herných titulov a teší sa obľube najmä medzi vývojármi hier pre *OS Android*.

Aktualizácie sa dočkal aj *Inšpektor*, ktorého hlavný komponent založený na proprietárnej knižnici *yFiles*, sa podarilo nahradiť za plne otvorené riešenie *jGraph*.

Výsledná architektúra aktualizovaného rámca sa nachádza na obrázku 2.



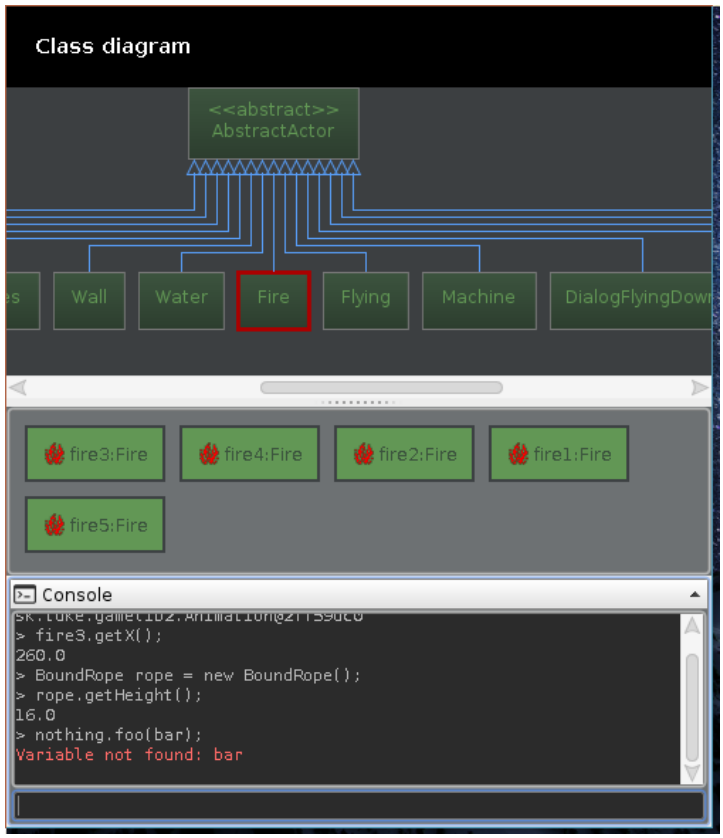
Obr. 2. Vrstvy vytvoreného herno-vzdelávacieho rámca [5]

Najdôležitejšie vlastnosti, o ktoré bol nový rámec rozšírený, je *podpora dialógov*, vďaka ktorým je možné v hre budovať príbeh, *časovač*, ktorý ilustruje využitie návrhového vzoru *observer*, a pomocou ktorého je možné vytvárať časované úlohy, a *príkazový riadok* v inovovanom inšpektorovi, vďaka ktorému je možné písať priamo konštrukcie jazyka *Java* bez nutnosti prekladu (viď obrázok 3).

3. Nová prípadová štúdia

Výraznými zmenami prešla aj samotná prípadová štúdia. Tentokrát sme sa snažili skĺbiť práve výhody oboch predchádzajúcich typov prípadových štúdií, ktoré sme v predchádzajúcich rokoch v rámci predmetu používali. Rozprávanie príbehov v textovej adventúre poskytovalo študentom dostatočný priestor pre rozvinutie ich vlastnej kreativity, ale textový výstup hry nebol veľmi atraktívny. Na druhej strane grafická strieľačka bola na pohľad modernejšia a zaujímavejšia,

[4] `https://libgdx.badlogicgames.com/`

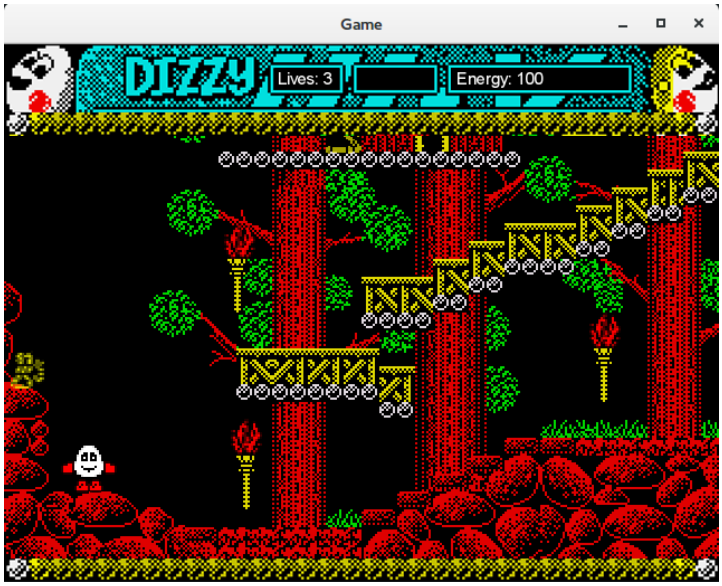


Obr. 3. Základné zobrazenie nového inšpektora objektov [5]

ale vďaka formátu hry a dostupnej grafike, neposkytovala príliš veľa možností pre vlastnú kreativitu, ako tomu bolo v prípade textovej adventúry.

Cieľom teda bolo nájsť hru, ktorá by bola kombináciu oboch týchto prístupov, kedy by študenti mohli vytvárať grafickú adventúru s jednoduchými akčnými prvkami. Vzhľadom na túto kombináciu sa nám ako vhodná alternatíva ponúkla populárna herná séria *Dizzy*[5] z 90-rokov minulého storočia. Vďaka niekoľkým oficiálne vydaným dielom a vďaka fanúšikom, ktorí aj dnes neustále vytvárajú nové príbehy, nie je núdza o grafické podklady. Študenti sa teda môžu na začiatku sústrediť na osvojenie si objektových princípov a v neskorších týždňoch semestra môžu naplno rozvinúť svoju kreativitu a vytvoriť svoje vlastné dobrodružstvo.

[5] *Dizzy*, *The Yolkfolk* and all related characters and titles are trademarks of *Blitz Games Limited* (*Blitz*) and *The Codemasters Software Company Limited* (*Codemasters*). All rights reserved. *Dizzy* and *The Yolkfolk* created by *The Oliver Twins*. `https://en.wikipedia.org/wiki/Dizzy_(series)`.



Obr. 4. Záber z hry vytvorenej pomocou nového softvérového rámca [3]

4. Overenie riešenia a jeho použiteľnosti

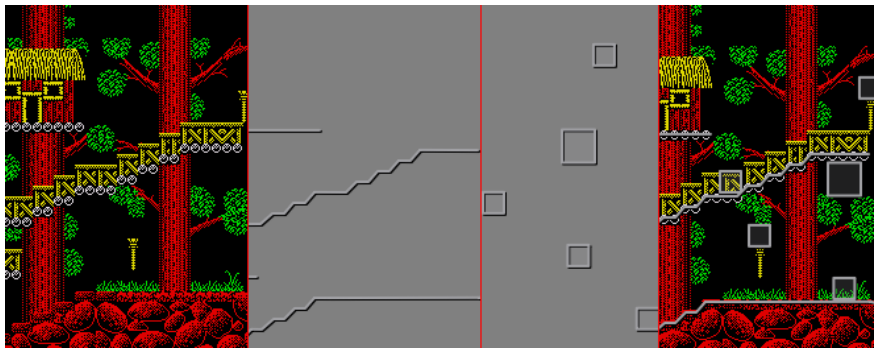
Prvým krokom pri testovaní možností nového softvérového rámca bolo vytvorenie jednoduchej hry. Ako predloha poslúžila hra *Dizzy 3 and a half* z roku 1991. Cieľom implementácie bolo overiť aktualizované API, že je naozaj funkčné a použiteľné. Výslednú implementáciu je možné vidieť na obrázku 4.

Herný svet je možné vytvoriť pomocou editora *Tiled*[6]. Skladá sa z niekoľkých vrstiev, v ktorých je vyobrazené pozadie herného sveta, prekážky, cez ktoré nie je možné prejsť a rozmiestnenie samotných herných objektov. Vyobrazenie jednotlivých vrstiev ako aj ich celkové spojenie sa nachádza na obrázku 5.

Po overení novej verzie rámca sme vytvorili niekoľko scenárov cvičení pre študentov. Tieto sme vyskúšali na dvoch skupinách študentov predmetu *Objektové programovanie* v školskom roku 2015/16 v priebehu šiestich týždňov. Študenti mali možnosť vyjadrovať sa k nedostatkom scenárov ako aj k nedostatkom softvérového rámca počas celého testovania. Aj vďaka ich pripomienkam a postrehom bol samotný rámec ešte viac vyladený.

Cieľom každého scenára bolo predstaviť niektorú z vlastností objektového programovania alebo niektorý z *návrhových vzorov*. Napr. v rámci prvých cvičení, počas ktorých študenti vytvárali svoje prvé herné objekty, sme ich v rámci scenárov vedome viedli k porušeniu tzv. *DRY* princípu. Upozornili sme ich na to hneď

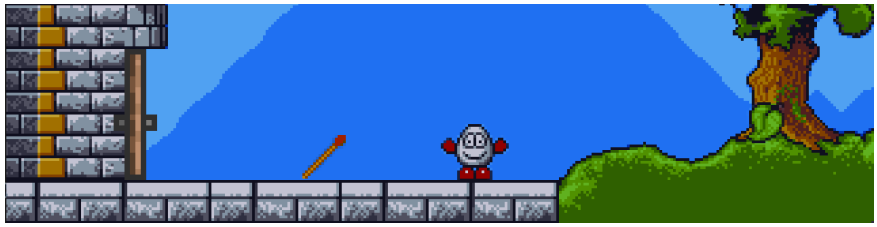
[6] `http://www.mapeditor.org/`



Obr. 5. Vrstvy herného sveta zľava doprava - pozadie, hranice, objekty a všetky spolu [3]

v nasledujúcom scenári a naučili sme ich, ako dokážu písať efektívnejší kód, ak začnú využívať *abstraktné triedy*.

Kvôli zvýšeniu motivácie sme sa snažili čo najviac scenárov realizovať vo forme jednoduchých herných epizód, kedy bolo potrebné niečo vyriešiť nielen programátorsky, ale aj herne. Napr. na otvorenie dverí vedúcich do hradu je potrebné k nemu položiť suché lístie, zapáliť ho a po zhorení dverí ich stačí len uhasiť vedrom vody (viď obrázok 6).



Obr. 6. Implementácia jedného zo scenárov.

Neoddeliteľnou súčasťou scenárov sú aj *diagramy tried*, ktoré pomáhajú získať celkový pohľad na problém a vzťahy jednotlivých tried.

Samotným scenárom ako aj experimentu sa bližšie venuje práca [3].

5. Záver

V rámci článku sme predstavili aktuálny stav výučby predmetu *Objektové programovanie* ponúkaného *Katedrou počítačov a informatiky* na *Technickej univerzite v Košiciach*. V rámci predmetu sme úspešne nasadili metodiky *objects first* a *design patterns first*. To sa nám podarilo najmä vytvorením herných prípadových štúdií založených na herno-vzdelávacom rámci *GameLib*.

Na aktualizácii tohto rámca ako aj na príprave experimentálnych cvičení na jeho overenie sa aktívne podieľali dvaja študenti tretieho ročnika oboru *Informatika Dominik Matta* a *Dominik Juhás*. Obaja boli čerstvými absolventmi uvedeného predmetu, takže v riešení sa odrazili aj ich vlastné skúsenosti s absolvovaním predmetu a nimi vytvoreného zadania.

Aj napriek tomu, že dnes už existuje množstvo nástrojov, metodík a materiálov, ktoré sa zaoberajú tým, ako si jednoduchšie osvojiť princípy objektového programovania, tá najťažšia úloha stále zostáva na učiteľovi, ktorý je akýmsi živým mostom medzi študentmi a samotným objektovým programovaním. A hernovzdelávací rámec *GameLib* mu v tejto neľahkej úlohe môže len a len pomôcť.

Literatúra

[1] BARNES, D. – KÖLLING, M.: *Objects first with Java : a practical introduction using BlueJ*, Pearson, Boston, 2012.

[2] BIŇAS, M. – NOVÁK, M. – MICHALKO, M.: *Learning of object oriented programmingas big adventure*, 2012.

[3] JUHÁS, D.: *Aktualizácia rámca gamelib pre výucbu metódou objects first i*, Master's thesis, Technická univerzita v Košiciach, Letná 9, Košice, Slovensko, 2016, bakalárska práca.

[4] LIVOVSKÝ, J. – BIŇAS, M. – PORUBÄN, J.: *Teaching object-oriented programming using object benches: Practical experience*, 2013.

[5] MATTA, D.: *Aktualizácia rámca gamelib pre výucbu metódou objects first ii*, Master's thesis, Technická univerzita v Košiciach, Letná 9, Košice, Slovensko, 2016, bakalárska práca.

[6] PECINOVSKÝ, R.: *Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky*, Grada, Praha, 2009.

Kontaktné adresy

Ing. Miroslav Biňas, PhD., Katedra počítačov a informatiky, Fakulta elektrotechniky a informatiky, Technická univerzita v Košiciach, Slovenská republika,
E-mailová adresa: `miroslav.binas@tuke.sk`

Bc. Dominik Juhás, Katedra počítačov a informatiky, Fakulta elektrotechniky a informatiky, Technická univerzita v Košiciach, Slovenská republika,
E-mailová adresa: `dominik.juhas@student.tuke.sk`

Bc. Dominik Matta, Katedra počítačov a informatiky, Fakulta elektrotechniky a informatiky, Technická univerzita v Košiciach, Slovenská republika,
E-mailová adresa: `dominik.matta@student.tuke.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 21–28

LaTeX a zrkadlo sadzby

RUDOLF BLAŠKO (SK)

Abstrakt. V dnešnej dobe značne pokročili nielen schopnosti počítačov, ale aj zručnosti ich používateľov. S touto pokročilou dobou súvisí aj úmyselné či neúmyselné znižovanie významu typografických pravidiel. Dnešné kvalitné DTP aplikácie majú tieto pravidlá v sebe zakomponované, ale taktiež poskytujú možnosti ako ich meniť. Môžeme meniť sadzobný obrazec, zrkadlo sadzby, kompozíciu stránky. Otázkami ostáva, či je to vždy vhodné, či dokáže bežný užívateľ bez typografických znalostí vytvoriť kvalitnú tlač. I keď človeku s nadpriemerným estetickým cítením sa to môže podariť. Na druhej strane, vo väčšine situácií sú základné typografické znalosti nevyhnutné.

Kľúčové slová. Sadzobný obrazec, zrkadlo sadzby, optický stred, pravidlo tretín, zlatý rez.

Abstract. At present, not only the computer skills but also the skills of their users have advanced significantly. This advanced stage is also characterized by an intentional or unintentional reduction of the importance of typography rules. Although high-quality DTP applications have these rules incorporated in them, they also provide a way to change them. They may vary the Canons of page construction, type area, composition of the page. In this regard, the question arises whether this is always appropriate and whether a typical user can create a quality print without typographic knowledge. Based on experience, it can be stated that the creation of high-quality print can only be mastered successfully by the person with the above-average aesthetic sense. However, in most situations, the typographic knowledge are necessary.

Keywords. Canons of page construction, type area, optical center, Rule of thirds, golden ratio.

Úvod

V dnešnej dobe značne pokročili nielen schopnosti počítačov, ale aj zručnosti ich používateľov. S touto pokročilou dobou súvisí aj úmyselné či neúmyselné znižovanie významu typografických pravidiel. Dnešné kvalitné DTP aplikácie majú tieto pravidlá v sebe zakomponované, ale taktiež poskytujú možnosti ako ich meniť. Môžeme meniť rôzne parametre sadzby a upravovať ju na svoj obraz. Otázkami ostáva, či je to vždy vhodné, či dokáže bežný užívateľ bez typografických znalostí vytvoriť kvalitnú tlač. Pri bežnej hladkej sadzbe je najjednoduchšie a asi aj najvhodnejšie nechať úpravu sadzby na počítač, t. j. kvalitný program. Ale je mnoho rôznych publikáčných výstupov (vizitky, plagáty, paspartovanie

grafiky ap.), ktoré by sme chceli vytlačiť kvalitne, ale hlavne podľa vlastných požiadaviek. Preto je dôležité poznať aspoň základné pravidlá, ktorými by sme sa mali riadiť, aby náš výstup bol kvalitný. Ak sa chceme kochať týmto produktom iba sami, všetko záleží od nášho vkusu. Ale v prípade, že ho chceme zverejniť, mal by sa páčiť aj ostatným ľuďom.

Základným pojmom je sadzba. Sadzbou rozumieme výsledok práce sadzačov vo forme zalomených stránok so všetkými typografickými a grafickými náležitosťami. Podobne ako ručne písaný text môže byť pekný a čitateľný, pekný a nečitateľný alebo nepekný a nečitateľný, aj sadzba môže byť pekná čitateľná alebo aj nepekná a tým pádom aj ťažko čitateľná. Tlač sa musí človeku páčiť a potom sa aj príjemne číta. Pravidlá sadzby nie sú samoúčelné, sú empiricky potvrdené, vytvárali sa tisícročia a ich nerešpektovanie alebo degenerovanie vedie k znetvoreniu sadzby a následnému negatívnemu zážitku z nej. V súčasnosti si väčšina autorov myslí, že uvedené pravidlá nepotrebuje a že sami iba pomocou počítača vytvoria nejaký článok, knihu alebo iba blog. Dôsledok toho je internet zaplavený rôznymi nevkusnými až hnusnými autorskými počinmi. Problém je, že písať vie každý a skoro každý si taktiež o sebe myslí, že má estetické cítenie. Ono je pravda, že každý človek má estetické cítenie, ale ono nemusí korešpondovať s tým, čo považujú za pekné iní ľudia.

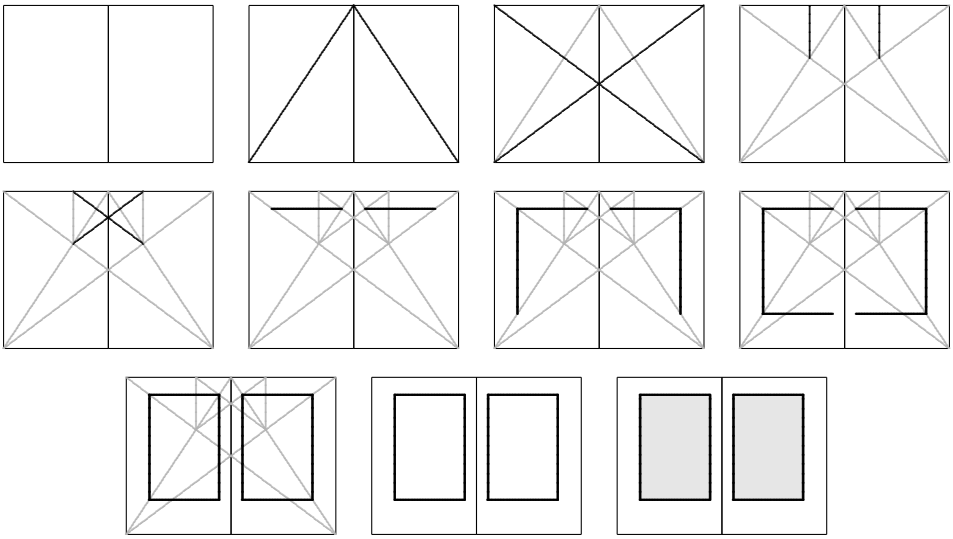
1. Sadzobný obrazec

Sadzobný obrazec je nákres umiestnenia sadzby na strane resp. dvojstrane a má veľký význam pre tvorbu kníh. Plocha sadzby umiestnená na strane slúži na výpočet rozsahu sadzby, určenie veľkosti a umiestnenia obrázkov a tabuliek. Udáva sa šírkou a výškou podľa druhu tlačoviny, pričom výška je udaná počtom riadkov a medziriadkovým prekladom. Okolo tlačenej plochy je biely okraj, ktorý je pre rôzne druhy publikácií rôzny. Napríklad ak sádzame básne bude väčší, ako pri sadzbe slovníkov alebo odborných kníh. Rozmer sadzobného obrazca ovplyvňuje zvolený formát papiera, stupeň písma, rozsah a charakter publikácie.

Zo skúsoností overených časom vyplýva základné pravidlo sadzby: **„Stred plochy sadzby je umiestnený vždy nad geometrickým stredom stránky.“** Z tohto pravidla je odvodená konštrukcia optického stredu stránky, ktorej sa venujeme neskôr.

Existuje mnoho konštrukcií sadzobného obrazca, v tomto príspevku sa venujeme iba dvom u nás najčastejšie používaným. Prvá je **klasická konštrukcia podľa uhlopriečok**, ktorá vychádza z uhlopriečneho delenia stránok. Je to univerzálne riešenie, ktoré postačí pre väčšinu prípadov. Na obr. 1 je ilustrovaný postup klasickej konštrukcie sadzobného obrazca na dvojstránke. Najprv vedieme uhlopriečky na jednotlivých stránkach z vnútorných horných rohov smerom k dolným vonkajším rohom stránok. Potom vedieme uhlopriečky na celej dvojstránke a z priesečníkov týchto uhlopriečok vedieme vertikálne úsečky smerom k hornej

hrane dvojstránky. Takto vznikne menší obdĺžnik a prieniky jeho uhlopriečok s uhlopriečkami jednotlivých stránok určia základné rohy sadzobného obrazca. Sadzobný obrazec dokreslíme rovnobežkami s hranami stránok tak, aby jeho rohy ležali na uvedených uhlopriečkach stránok a dvojstránky.

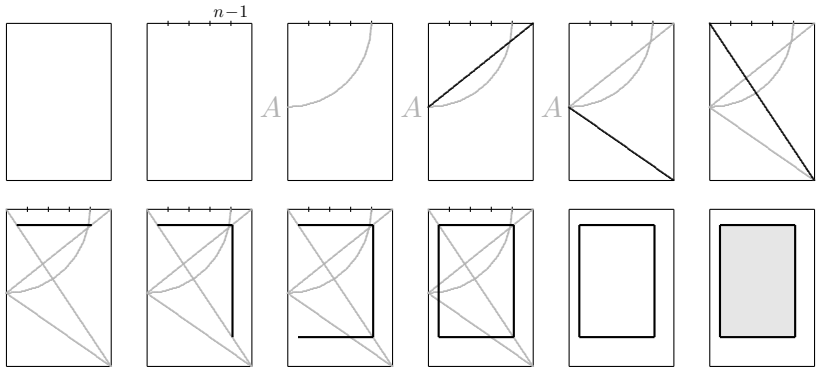


Obr. 1. Klasická konštrukcia sadzobného obrazca podľa uhlopriečok

Najčastejšie používanou konštrukciou sadzobného obrazca je **rozšírená konštrukcia podľa uhlopriečok**. Túto konštrukciu môžeme meniť v rôznych pomeroch v závislosti od potlačenej plochy, ktorú potrebujeme. Pre výtvarné publikácie konštruujeme menší sadzobný obrazec a teda väčšie okraje. Naopak pre beletriu alebo odborné a vedecké publikácie používame čo najväčšiu plochu papiera. Do úvahy musíme zobrať aj počet strán a s tým súvisiace otváranie knihy.

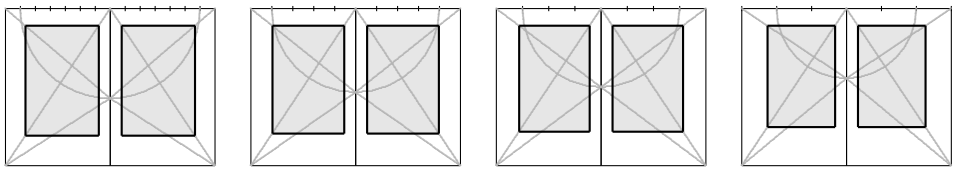
Pri rozšírená konštrukcii podľa uhlopriečok si musíme najprv zvoliť prirodzené číslo n, ktoré nám určí pomer konštrukcie. Na obr. 2 je znázornený postup konštrukcie pre samostatnú stránku. Pre dvojstránku je postup analogický (obr. 3). Najprv rozdelíme hornú hranu čistého formátu na n častí a označíme $n-1$ častí. Následne týchto $n-1$ častí nanesieme zhora na chrbtovú stranu, pričom dostaneme bod, ktorý pre jednoduchosť označme A. Z bodu A vedieme uhlopriečky do protiľahlých rohov stránky, potom vedieme z ľavého horného rohu stránky do jej pravého dolného rohu uhlopriečku. Nad priesečníkom týchto uhlopriečok nanesieme horizontálne šírku sadzby a vertikálnymi úsečkami a dolnou vodorovnou čiarou doplníme sadzobný obrazec tak, aby jeho pravý dolný roh ležal na uhlopriečke stránky.

Rozšírená konštrukcia podľa uhlopriečok sa používa v rôznych pomeroch pre rôzne druhy publikácie. V pomere $3:2$ najmenej zaplňuje plochu papiera a používa



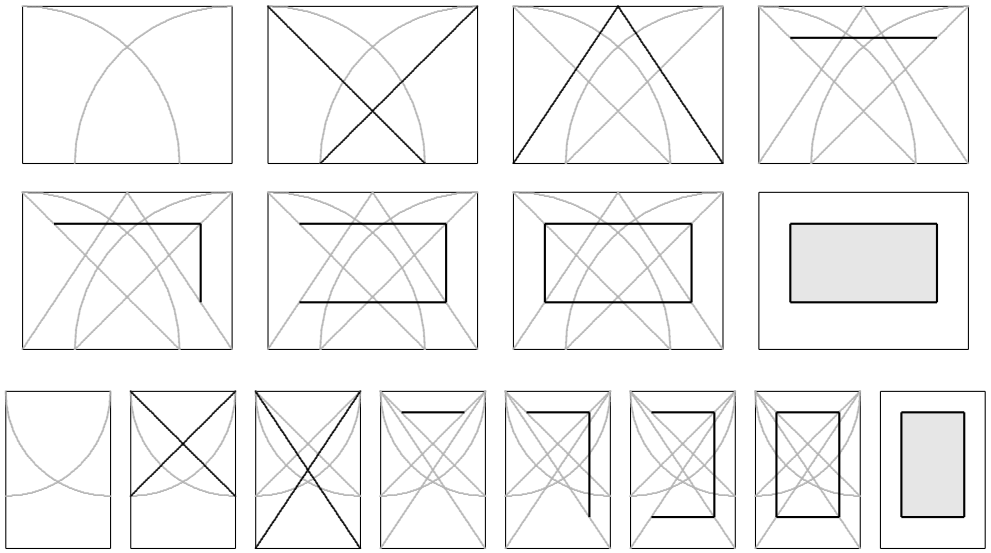
Obr. 2. Rozšírená konštrukcia sadzobného obrazca podľa uhlopriečok

sa napr. na sadzbu básni, v pomere 4 : 3 sa používa pre náročnejšie textové a obrazové publikácie a v pomere 5 : 4 sa najčastejší pre beletriu a taktiež pre vedeckú literatúru. Pri úprave vedeckých a odborných kníh sa neodporúča experimentovať so sadzobným obrazcom. Kniha je určená k štúdiu a nepodlieha módnym trendom.



Obr. 3. Dvojstránka pri rovnakom odsadení od vnútornej a od hornej strany pri pomeroch 7 : 6, 5 : 4, 4 : 3 a 3 : 2

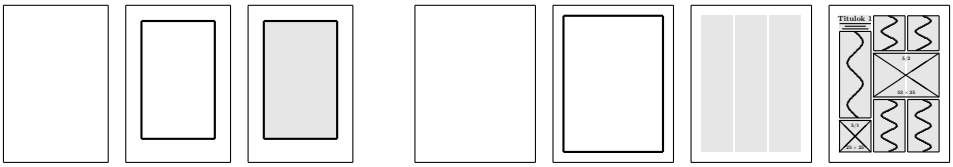
Ak chceme umiestniť text (obrázok, grafiku ap.) **na samostatnú stránku**, platia analogické pravidlá ako pre sadzobný obrazec knihy (najväčší okraj v spodnej časti). Na obr. 4 je znázornená konštrukcia sadzobného obrazca do optického stredu, ktorá je vhodná napríklad pre paspartovanie grafiky. Pri orientácii na výšku začíname v horných rohoch stránky a pri orientácii na šírku začíname v dolných rohoch stránky. Z týchto rohov vedieme kružnice s polomerom rovným šírke (pri orientácii na výšku), resp. výške stránky (pri orientácii na šírku). Z priesečníkov týchto kružníc s bočnými hranami, resp. dolnou hranou stránky vedieme uhlopriečky do protiľahlých horných rohov stránky. Zostrojíme uhlopriečky stránky, resp. zo stredu hornej hrany vedieme uhlopriečky do dolných rohov stránky. Nad priesečníkom týchto uhlopriečok nanesieme horizontálne šírku obrazca a vertikálnymi úsečkami a dolnou vodorovnou čiarou doplníme sadzobný obrazec tak, aby jeho dolné rohy ležali na naposledy zostrojených uhlopriečkach.



Obr. 4. Paspartovanie grafiky

2. Zrkadlo sadzby

Zrkadlo sadzby je zákres, z ktorého je možné zistiť polohu sadzby na strane, veľkosť okrajov, počet stĺpcov, ich šírku a medzistĺpcové medzery, veľkosť a polohu titulkov, obrázkov, tabuliek, liniek alebo iných prvkov na stránke. Jeho súčasťou býva aj predpis druhu a veľkosti písma a medziriadkového prekladu. Zhotovuje ho výtvarník alebo technický redaktor (obr. 5).

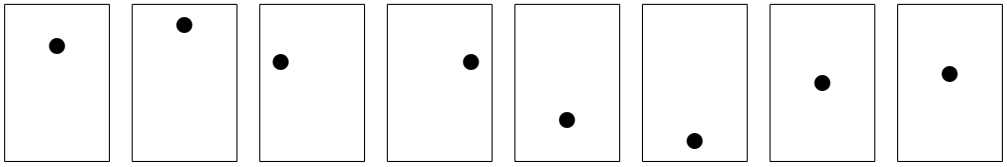


Obr. 5. Zrkadlo sadzby

3. Umiestňovanie bodu na stránke

Zákony matematické a geometrické nie vždy korešpondujú s estetickým (optickým) vnímaním človeka. Táto skutočnosť sa stala natoľko súčasťou nášho podvedomia, že sa výrazne prejavuje aj pri posudzovaní kompozície. Človek do svojho vnímania nevedomky premieta svoje skúsenosti a ľudské oko dokáže neskutočne klamať. Dokazuje to zákon optického stredu ako analógia gravitačného zákona.

Keďže je príťažlivosť k spodnej hrane väčšia ako hornej, opticky sa nám javí bod umiestnený do **geometrického stredu strany** (priesečníka uhlopriečok) **nie uprostred obrazca, ale až pod ním** (stredom). S uvedenou skutočnosťou súvisí už spomínané základné pravidlo získané na základe historických skúseností: **„Stred plochy sadzby musí byť umiestnený vždy nad geometrickým stredom strany. “**



Obr. 6. Umiestňovanie bodu na stránke

Na obr. 6 je znázornený bod v rôznych pozíciach na stránke. Bod v hornej polovici plochy pôsobí dojmom vnášania. Je ľahký, celá kompozícia je vzdušná a pokojná. Tento dojem potrvá až do takej výšky, kedy budeme mať pocit, že je na ňom plocha zavesená. Bod bude priťahovaný hornou hranou plochy. Napätie sa uvoľní, až keď bod splynie s hranou.

Každá strana stránky pôsobí na bod inou silou. Ľavá strana priťahuje bod menej ako pravá. Spôsobuje to spôsob nášho čítania zľava doprava, ktorý považujeme za prirodzený a plynulý.

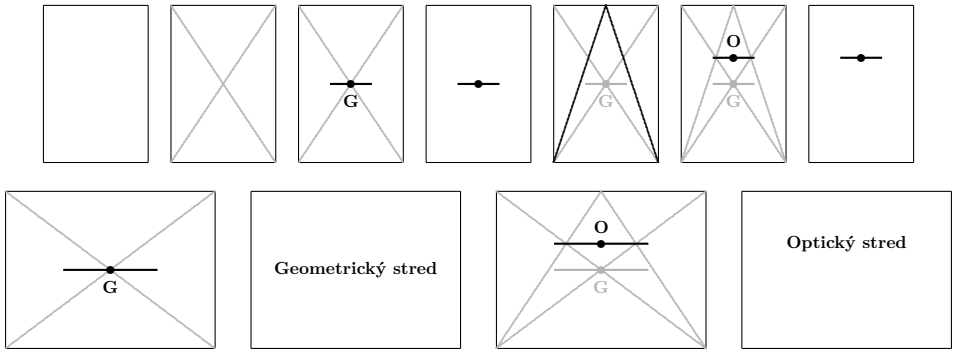
Spodná časť stránky reprezentuje zem a jej gravitačnú silu. Čím je bod nižšie k spodnej hrane, tým je silnejšie priťahovaný ku spodnej hrane. Bod umiestnený na spodnej hrane je úplne pokojný, leží na nej.

Kvôli väčšej príťažlivosti k spodnej hrane sa nám bod umiestnený do geometrického stredu javí opticky nižšie. Aby sa nám bod javil v strede, musíme ho posunúť trochu vyššie, najlepšie do optického stredu stránky. Na **optický stred**, t. j. miesto, v ktorom si naše oči predstavujú stred stránky (základný a rokmi overený kompozičný poznatok), sa síce často zabúda, ale mnoho ľudí s estetickým cítením ho používa bez toho, aby si to uvedomovalo. Optický stred je umiestnený približne v hornej tretine strany. Rôzne knihy vyžadujú špecifické riešenia sadzobného obrazca a teda aj optického stredu. V praxi sa použije viacero návrhov, tie sa vytlačia a vyberie sa najlepší (najkrajší) z nich.

Jeden z osvedčených (ale nie nevyhnutných) spôsobov konštrukcie optického stredu O je znázornený na obr. 7 (geometrický stred G).

4. Zlatý rez a zlaté body

Je dokázané, že určité proporčné vzťahy pôsobia na človeka prirodzenejšie ako iné. Jeden z takýchto pomerov sa nazýva zlatý rez. V prírode sa opakuje nespočítateľne veľa krát a naše oko je naň podvedome zvyknuté. Pri svojej tvorbe

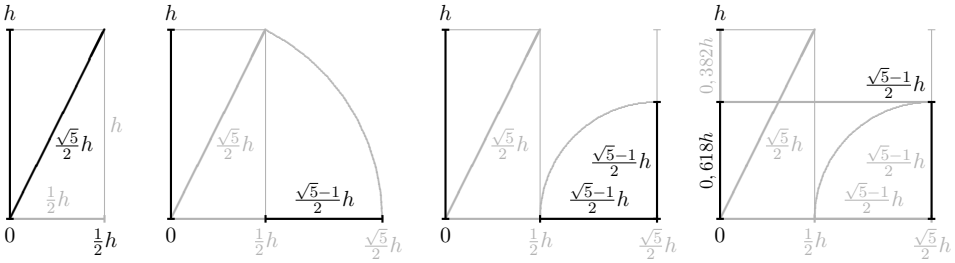


Obr. 7. Konštrukcia optického stredu stránky

ho využívalo a aj využíva mnoho umelcov. **Zlatý rez** na úsečke dĺžky h vyjadruje taký bod x, pre ktorý platí pomer $\frac{x}{h} = \frac{h-x}{x}$, resp. $x^2 + hx - h^2 = 0$,

$$\text{t. j. } x = \frac{-h+\sqrt{h^2+4h^2}}{2} = \frac{\sqrt{5}-1}{2}h \approx 0,618h.$$

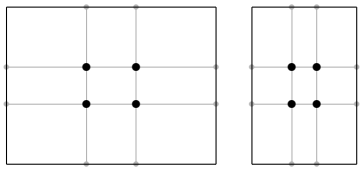
V praxi teda delíme úsečku v pomere $0,618 : 0,382$. Konštrukcia zlatého rezu je znázornená na obr. 8.



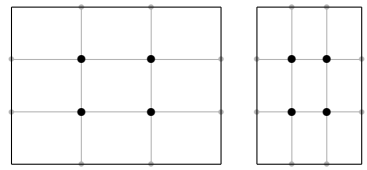
Obr. 8. Geometrická konštrukcia zlatého rezu

Zlatý rez sa často používa nielen v typografii, ale aj pri kompozícii kresby alebo fotografie. Nebolo by správne tvrdenie, že bez zlatého rezu sa dobrá kompozícia nezaobíde. Niekedy je vhodnejšie použiť stredovú kompozíciu alebo pravidlo tretín. Zlatý rez môžeme urobiť pre každú stranu obdĺžnika a v priesečníkoch rovnobežiek so stranami získame **zlaté body** (obr. 9).

Často, najmä v kompozícii výtvarných a fotografických diel, sa používa **pravidlo tretín** (angl. Rule of Thirds). V tomto prípade rozdelíme kompozičnú stránku na 9 rovnakých častí (podľa tretín jednotlivých strán). Cieľom je rozmiestniť objekty, resp. subjekty záujmu do blízkosti jednej z línií tak, aby bol obraz rozdelený na tri rovnaké časti. Druhotným cieľom je umiestnenie týchto objektov do priesečníkov tretinových línií.



Obr. 9. zlaté body



Obr. 10. Pravidlo tretín

Poďakovanie. Príspevok vznikol s prispením grantu KEGA 011–4/2014ŽU „Experimentálna matematika – zviditeľnenie neviditeľného“ podporeného Slovenskou kultúrno-edukačnou grantovou agentúrou.

Literatúra

[1] BRINGHURST, R.: The Elements of Typographic Style, Hartley Marks Publishers 2004, ISBN 0-88179-205-5.
[2] KNUTH, D. E.: *The TEXbook*, Volume A of *Computers and Typesetting*, Addison-Wesley Publishing Company (1984), ISBN 0-201-13448-9.
[3] KOPKA, H. – DALY, P. W.: *LATEX – Podrobný průvodce*, Brno, Computer Press, 2004, ISBN 80-722-6973-9.
[4] PANÁK, J. – ČEPPAN, M. – DVONKA, V. – KARPINSKÝ, Ľ. – KORDOŠ, P. – MIKULA, M. – JAKUCEWICZ, S.: *Polygrafické minimum*, Bratislava, Typoset, 2008, ISBN 978-80-970069-0-7.
[5] RYBIČKA, J.: *LATEX pro začátečníky*, Brno, KONVOJ 2003, ISBN 80-7302-049-1.
[6] ŠÍP, R.: *Typografické minimum*, Bratislava, SOU Polygrafické, Zväz polygrafie na Slovensku, 2000, ISBN 80-967598-5-X.

Kontaktná adresa

RNDr. Rudolf Blaško, PhD., Katedra matematických metód a operačnej analýzy, Fakulta riadenia a informatiky, Žilinská univerzita, Univerzitná 8215/1, 010 26 Žilina, Slovensko,
E-mailová adresa: `beerb@frcatel.fri.uniza.sk`

8. ročník Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf
OSSConf 2016: 29–34

ROZDEĽUJ A POČÍTAJ S MPI!

ZUZANA BORČINOVÁ (SK)

Abstrakt. Existuje veľa úloh, ktorých riešenie je časovo náročné napríklad preto, že obsahuje množstvo sérií cyklov a hoci jeden cyklus nemusí trvať dlho, vykonanie všetkých zaberie veľa času. Keby sme však úlohu dali riešiť paralelne viacerým procesorom, mohli by sme výpočet urýchliť. Základom úspechu je, aby program dokázal rozdeliť úlohu na menšie podúlohy a zabezpečiť komunikáciu medzi kooperujúcimi procesormi. V tomto článku názorne ukážeme paralelizáciu algoritmu od identifikácie paralelizmu, rozdelenia úloh viacerým procesorom až po implementáciu v programovacom jazyku Python s použitím balíčka mpi4py.

Kľúčové slová. Paralelné výpočty, Eratostenovo sito, MPI, mpi4py.

DIVIDE AND CALCULATE WITH MPI!

Abstract. There is a great number of tasks, which are very time-consuming to solve, partly because they consist of many series of loops and while one loop is not very time-expensive, evaluating all of them is. However, if the task was being solved simultaneously by multiple processors, we could lower the time needed to get the result. Main basis to success is in effective division of task to smaller sub-tasks and communication and synchronization between cooperating processors. In this paper, we'll show parallelizing an algorithm, from identification of parallelism, task division and passing the tasks to multiple processors all the way to implementation in Python language software with mpi4py package.

Keywords. Parallel computations, Sieve of Eratosthenes, MPI, mpi4py.

Úvod

Najdôležitejším krokom pri paralelizácii algoritmu je nájdenie nezávislých výpočtov, ktoré sa dajú vykonať paralelne. V tomto článku názorne ukážeme paralelizáciu Eratostenovho sita od identifikácie paralelizmu, rozdelenia úloh viacerým jadrám až po implementáciu v programovacom jazyku Python s použitím balíčka mpi4py.

1. Eratostenovo sito

Eratostenovo sito je jednoduchý algoritmus na nájdenie všetkých prvočísel menších ako zadané prirodzené číslo n [5]. Najprv na príklade objasníme jeho klasickú (sekvenčnú) verziu a identifikujeme, ktoré kroky algoritmu sa dajú vykonať paralelne.

1.1. Algoritmus

Máme vyhľadať všetky prvočísla menšie ako prirodzené číslo $n = 26$.

1. Napíšeme si čísla od 2 do 25 (číslo 1 nie je prvočíslo).
   2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
2. Označíme číslo 2 a vyškrtneme všetky jeho násobky.
   2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
3. Označíme ďalšie najmenšie nevyškrtnuté číslo, čiže 3 a vyškrtneme všetky jeho násobky.
   2 **3** × 5 × 7 × 9 × 11 × 13 × 15 × 17 × 19 × 21 × 23 × 25
4. Tak pokračujeme až dovtedy, kým v zozname neoznačíme posledné nevyškrtnuté číslo. Označené čísla sú prvočísla.
   2 3 × **5** × 7 × × × 11 × 13 × × × 17 × 19 × × × 23 × 25
   2 3 × 5 × **7** × × × 11 × 13 × × × 17 × 19 × × × 23 × ×
   2 3 × 5 × 7 × × × **11** × 13 × × × 17 × 19 × × × 23 × ×
   2 3 × 5 × 7 × × × 11 × **13** × × × 17 × 19 × × × 23 × ×
   2 3 × 5 × 7 × × × 11 × 13 × × × **17** × 19 × × × 23 × ×
   2 3 × 5 × 7 × × × 11 × 13 × × × 17 × **19** × × × 23 × ×
   2 3 × 5 × 7 × × × 11 × 13 × × × 17 × 19 × × × **23** × ×

Vidíme, že keď označíme nejaké číslo i, jeho najmenší nevyškrtnutý násobok je $i.i$, lebo menšie násobky už boli vyškrtnuté v predchádzajúcich krokoch. Ak je číslo $i \geq \sqrt{n}$, tak $i.i \geq n$, čiže všetky jeho násobky menšie ako n sú už vyškrtnuté. Preto stačí vyškrtnúť násobky prvočísel menších ako $\sqrt{n}$ a potom všetky nevyškrtnuté čísla sú prvočísla [6].

1.2. Čo sa dá vykonať paralelne?

Pri hľadaní prvočísel menších ako n môžeme postupovať tak, že najprv nájdeme všetky prvočísla menšie ako $\sqrt{n}$ a potom vyškrtneme ich násobky medzi $\sqrt{n}$ a n. Vyškrtávanie sa dá paralelizovať napríklad tak, že rozdelíme nájdené prvočísla jednotlivým procesorom a každý procesor vyškrtne zo zoznamu od $\sqrt{n}$ po n len násobky svojich prvočísel. Iný spôsob paralelizácie je taký, že rozdelíme zoznam čísel od $\sqrt{n}$ po n na úseky a pridelíme ich procesorom. Každý procesor potom vyškrtne násobky všetkých prvočísel v pridelenom úseku. V nasledujúcej časti podrobnejšie vysvetlíme oba uvedené spôsoby paralelizácie.

2. Paralelizácia

Ako sme už naznačili, pri paralelizácii sa výpočet rozdelí na časti, ktoré sa budú vykonávať súčasne viacerými procesormi. Po splnení úlohy procesory odovzdajú svoje výsledky na ďalšie spracovanie.

V našom príklade hľadáme prvočísla menšie ako 26. Našli sme všetky prvočísla od 2 po $\sqrt{26} \doteq 5$ (2, 3 a 5). Ostáva nájsť prvočísla od 6 po 25.

2.1. Rozdeľuj a počítaj!

Zo zoznamu čísel od 6 po 25 je potrebné vyškrtnúť násobky prvočísel 2, 3 a 5.

Prvý spôsob. Úlohu rozdelíme trom procesorom.

- Procesorom rozdelíme prvočísla: 2 *(proc0)*, 3 *(proc1)*, 5 *(proc2)*.
- Každý procesor vyškrtne len násobky svojich prvočísel:
  proc0: ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25
  proc1: ~~6~~ 7 8 ~~9~~ 10 11 ~~12~~ 13 14 ~~15~~ 16 17 ~~18~~ 19 20 ~~21~~ 22 23 ~~24~~ 25
  proc2: 6 7 8 9 ~~10~~ 11 12 13 14 ~~15~~ 16 17 18 19 ~~20~~ 21 22 23 24 ~~25~~
- Čísla, ktoré neboli vyškrtnuté žiadnym procesorom sú prvočísla: **7**, **11**, **13**, **17**, **19**, **23**.

Druhý spôsob. Úlohu rozdelíme taktiež trom procesorom.

- Procesorom rozdelíme čísla:
  od 6 po 12 *(proc0)*, od 13 po 19 *(proc1)*, od 20 po 25 *(proc2)*.
- Každý procesor vyškrtne násobky prvočísel 2, 3 a 5 len vo svojom úseku:
  proc0: ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~
  proc1: 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19
  proc2: ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~
- Nevyškrtnuté čísla sú prvočísla: **7**, **11**, **13**, **17**, **19**, **23**.

Pri paralelizácii môže každý procesor využívať svoju vlastnú pamäť oddelene od pamätí ostatných procesorov (paralelizmus s distribuovanou pamäťou). Vzájomnú výmenu dát medzi procesormi umožňuje technika známa ako *message passing*. Ďalej ukážeme, aký druh komunikácie medzi procesormi je vhodný pri oboch navrhnutých spôsoboch paralelizácie Eratostenovho sita.

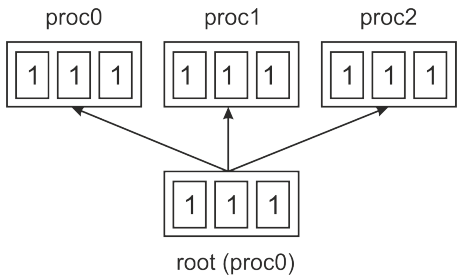
2.2. Komunikácia

V prípade paralelných výpočtov s distribuovanou pamäťou všetky kooperujúce procesory vykonávajú ten istý program so svojimi (odlišnými) parametrami. Procesor s číslom 0 (*root*) pošle dáta všetkým procesorom a po výpočte zas od nich prijme a spracuje výsledky.

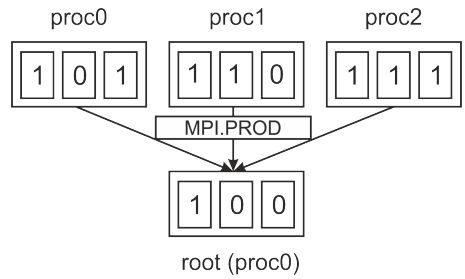
Jedným zo štandardov *message passing* je MPI (Message Passing Interface), ktorý definuje funkcie pre rôzne spôsoby posielania dát.

Funkcie Broadcast a Reduce. Pri prvom navrhnutom spôsobe má každý procesor vyškrtnúť zo zoznamu čísel od 6 po 25 násobky jemu pridelených prvočísel. Root musí poslať všetkým procesorom zoznam čísel od 6 po 25. To zabezpečíme použitím funkcie *Broadcast* (obr. 1).

Každý procesor teraz má svoj vlastný zoznam čísel a vykoná v ňom rovnaký program ako ostatné procesory (vyškrtne násobky prvočísel), ale so svojimi prvočíslami. Po dokončení svojej úlohy musia procesory poslať svoje zoznamy rootovi a ten ich spracuje (zistí, ktoré čísla neboli vyškrtnuté žiadnym procesorom). Na to je vhodné použiť funkciu *Reduce* (obr. 2).



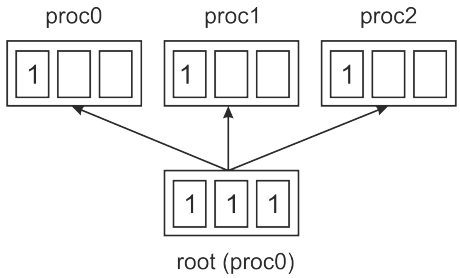
Obr. 1. Broadcast



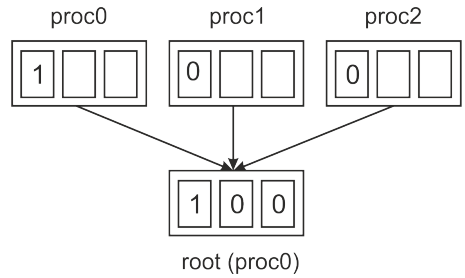
Obr. 2. Reduce

Funkcie Scatter a Gather. Druhý navrhnutý spôsob vyžaduje inú formu posielania dát. Root najprv rozdelí zoznam čísel od 6 po 25 na tri úseky a rozpošle ich procesorom – každý procesor dostane iný úsek. Takéto posielanie dát umožňuje funkcia *Scatter* (obr. 3).

Keď procesory vykonajú úlohu na svojich úsekoch (vyškrtnú v nich násobky prvočísel 2, 3 a 5), musia ich zaslať rootovi, aby ich opäť spojil do jedného zoznamu. Použijeme teda funkciu *Gather* (obr. 4).



Obr. 3. Scatter



Obr. 4. Gather

3. Implementácia

Naše riešenie implementujeme v programovacom jazyku Python [2] s použitím balíčka mpi4py [3]. Najprv inicializujeme komunikátor (`comm`), čiže logickú jednotku, ktorá definuje množinu procesorov, ktoré môžu medzi sebou komunikovať.

Počet procesorov je `size` a sú identifikované pomocou svojho čísla (`rank`), pričom sa číslujú od 0 po `size-1`. V Pythone zapíšeme:

```
from mpi4py import MPI
comm=MPI.COMM_WORLD
rank=comm.Get_rank()
size=comm.Get_size()
```

Zoznam čísel v Eratostenovom site budeme reprezentovať pomocou NumPy poľa `cisla`, pre ktorého prvky platí: `cisla[i]=0` ak je číslo `i` vyškrtnuté, v opačnom prípade je `cisla[i]=1`. Na začiatku sú všetky čísla počnúc 2 nevyškrtnuté [4]:

```
import numpy as np
cisla=np.ones(26,dtype=np.int8)
cisla[:2]=0
```

Analogicky aj lokálne dáta procesorov `svoje` a tiež výsledné dáta `dokopy` budú NumPy polia. Funkcie *Broadcast, Reduce, Scatter a Gather* implementujeme v Pythone nasledovne [1]:

Broadcast: Lokálne dáta procesora s číslom 0 `svoje` sú čísla od 2 po 25. Vykonaním funkcie `comm.Bcast(svoje, root=0)` pošle root (procesor 0) svoje lokálne dáta ostatným procesorom.

```
if rank==0:
  svoje=cisla
  comm.Bcast(svoje,root=0)
```

Reduce: Pomocou funkcie `comm.Reduce(svoje, dokopy, op=MPI.PROD, root=0)` pošle každý procesor svoje lokálne dáta `svoje` rootovi, root prijaté dáta spracuje pomocou operácie `op` (v našom prípade súčin `MPI.PROD`) a výsledkom budú dáta `dokopy`.

```
comm.Reduce(svoje,dokopy,op=MPI.PROD,root=0)
```

Scatter: Lokálne dáta procesora s číslom 0 `ostatne` sú čísla od 6 po 25. Funkcia `comm.Scatter(ostatne, svoje, root=0)` umožní rootovi rovnomerne rozdeliť svoje lokálne dáta `ostatne` všetkým procesorom, t. j. každý procesor prijme svoj úsek `svoje`. Pretože v našom prípade sa čísla od 6 po 25 nedajú rovnomerne rozdeliť 3 procesorom, použili sme funkciu `comm.Scatterv(.)`.

```
if rank==0:
  ostatne=cisla[6:]
else:
  ostatne=None
comm.Scatterv(ostatne,svoje,root=0)
```

Gather: Funkciou `comm.Gather(svoje, dokopy, root=0)` zas každý procesor pošle svoje lokálne dáta `svoje` rootovi a ten ich „spojí“ do jedného

výsledku `dokopy`. Keďže dĺžka `svoje` všetkých procesorov nie je rovnaká, museli sme použiť funkciu `comm.Gatherv(.)`.

```
comm.Gatherv(svoje,dokopy,root=0)
```

Program spustíme príkazom

```
mpiexec -n 3 ipython erato.py
```

Číslo 3 špecifikuje zvolený počet procesorov a `erato.py` je názov textového súboru s našim programom, ktorý je uložený v aktuálnom adresári.

4. Záver

Možnosti paralelizácie výpočtov sa rozšírili s príchodom viacjadrových procesorov, ale či bude ich potenciál využitý, závisí hlavne na programátoroch. Proces paralelizácie algoritmu sme priblížili na príklade Eratostenovho sita pre jeho pomerne malú náročnosť na čitateľove matematické znalosti a pritom sme predstavili použitie mpi4py – balíčka programovacieho jazyka Python pre paralelizáciu výpočtov.

Poďakovanie. Príspevok vznikol s prispením grantu KEGA 011–4/2014ŽU „Experimentálna matematika – zviditeľnenie neviditeľného“ podporeného Slovenskou kultúrno-edukačnou grantovou agentúrou.

Literatúra

[1] *A Python Introduction to Parallel Programming with MPI 1.0 documentation*, `http://materials.jeremybejarano.com/MPIwithPython/`.
[2] *Python 3.4.3 documentation*, `http://docs.python.org/3/`.
[3] *MPI for Python 2.0.0 documentation*, `http://pythonhosted.org/mpi4py/usrman/index.html`.
[4] *NumPy*, `http://www.numpy.org/`.
[5] `https://sk.wikipedia.org/wiki/Eratostenovo_sito`.
[6] WEEDEN, A.: *Parallelization:Sieve of Eratosthenes*, `http://www.shodor.org/media/content//petascale/materials/UPModules/sieveOfEratosthenes/module_document_pdf.pdf`.

Kontaktná adresa

RNDr. Zuzana Borčinová, Katedra matematických metód, Fakulta riadenia a informatiky, Žilinská univerzita, Univerzitná 8215/1, 010 26 Žilina, Slovenská republika,
E-mailová adresa: `zuzana.borcinova@fri.uniza.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 35–42

VIZUÁLNÍ PROGRAMOVÁNÍ V QGIS

ZDENA DOBEŠOVÁ (CZ)

Abstrakt. Vizuální programování je způsob jak jednoduchou grafickou formou navrhnout algoritmus, či postup zpracování dat. Je potěšující, že i Open Source softwary v GIS mají dnes k dispozici tuto komponentu pro vizuální programování. V programu QGIS je to Processing Modeler, v GRASS GIS je to Graphical Modeler. U každého vizuálního programovacího jazyka jsou důležité jak jeho funkční možnosti, tak je důležitý použitý vizuální slovník, resp. sada grafických symbolů a spojných čar. Na srozumitelnost diagramu má vliv jak podoba jednotlivých symbolů, jejich uspořádání, tak texty jako jsou názvy operací, dat atd. Vizuální slovník je důležitý z hlediska kognice jak při sestavování nového diagramu, tak při opakovaném použití diagramů stejným nebo jiným uživatelem.

Článek ukáže možnosti hodnocení vizuálního slovníku Processing Modeleru pro QGIS jak podle teoretických principů, tak možnosti hodnocení pomocí eye-tracking měření.

Klíčová slova. GIS, hodnocení, symbol, grafický editor, eye-tracking.

VISUAL PROGRAMMING IN QGIS

Abstract. Visual programming is a way how to simply design algorithm of data processing in a graphical form. Open Source software for GIS contains also this component for visual programming. In QGIS application, it is the Processing Modeler; in GRASS GIS it is the Graphical Modeler. For every visual programming language both its functional capabilities as well as the visual vocabulary used (set of symbols and line connectors) are important. The intelligibility of diagrams is influenced both by the shape of the individual symbols and by their configuration, as well as by the text as the names of operations, data, etc. The visual vocabulary is important from the cognition point of view both while designing a new diagram or during the repetitive use of a diagram by the same user or by another user.

The article shows the possibilities of assessment of the QGIS Processing Modeler visual vocabulary based on the theoretical principles as well as by eye-tracking measurement.

Keywords. Evaluation, symbol, graphical editor, eye-tracking.

Úvod

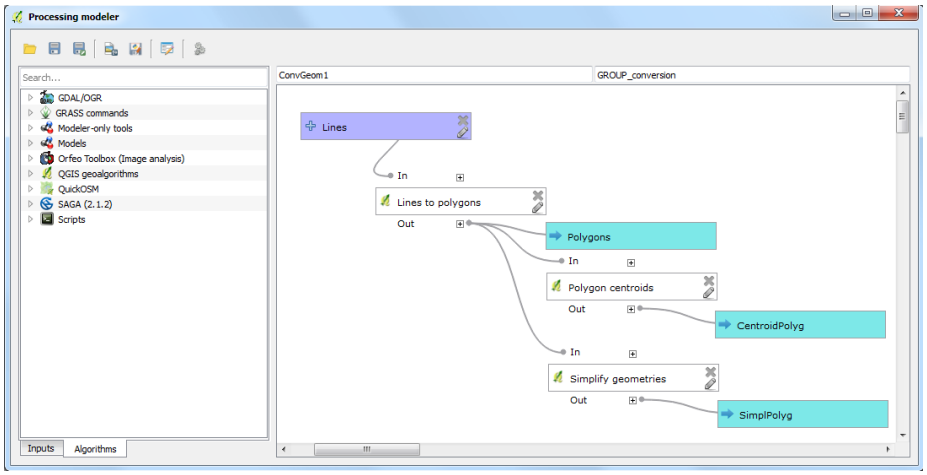
Vizuální programování je způsob zápisu algoritmu v grafické formě. Použitý jazyk je označován jako vizuální programovací jazyk (VPL). Vizuální reprezentace algoritmu resp. jednotlivých kroků postupu (tzv. workflow) je velice frekventované v informatice i v dalších oblastech. Stranou nezůstaly ani GIS softwary, kde uživatelé mají možnost graficky sestavovat postup zpracování prostorových dat. Z komerční sféry je dobře známá komponenta ModelBuilder z programu ArcGIS

for Desktop, dostupná ve verzích od roku 2004. Mezi další komerční zástupce patří Spatial Modeler v softwaru ERDAS Imagine, Macro Modeler v softwaru IDRISI anebo Workflow Designer v softwaru AutoCAD Map 3D (od roku 2010). Detailní popis komponent přináší přehled uvedený v článku [2].

Až v posledních letech začínají být uživatelům dostupné komponenty i v Open Source GIS softwarech. Od verze QGIS Dufour 2.0 vydané v 2013 je dostupná pro vizuální programování první verze komponenty Processing Modeler [6]. Od roku 2013 je také dostupná komponenta Graphical Modeler v GRASS GIS [4]. Diagramy znázorňující algoritmus jsou v obou těchto komponentách označovány jako modely. Oba termíny, model a diagram, lze pro potřeby VPL v GIS chápat jako synonyma. Dostupnost těchto komponent je výhodou pro uživatele Open Source GIS softwarů. Nespornou výhodou tvorby modelů je možnost navrhnout a zaznamenat postup dávkového zpracování dat tak, že výrazně automatizuje zpracování prostorových dat. Nezanedbatelná je i možnost opakovaného použití modelu pro jiná data, data z jiného území nebo aktualizovaná data, která byla již dříve zpracována. Poslední výhodou je vysoce samodokumentační funkce grafického zápisu algoritmu.

1. Interface a grafická notace Processing Modeler

Grafická komponenta Processing Modeler pracuje v samostatném dialogovém okně. Okno je rozděleno na levé užší okno, kde se přepíná mezi dvěma kartami Inputs a Algorithms. Pravé větší okno slouží jako prostor pro grafický návrh modelu (Obr. 1).

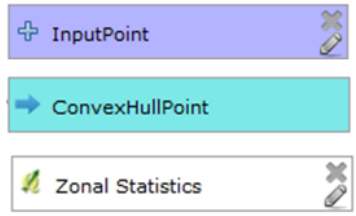


Obrázek 1. Rozhraní komponenty Processing Modeler s vytvořeným ukázkovým modelem.

Vlastní návrh se děje nejprve výběrem a vložením vstupních parametrů (dat) z karty Inputs. Zde se vybírá název a typ vstupních dat. Dále se metodou „táhni a pusť“ vkládají z karty Algorithms GIS operace do modelu. Ve skupině Algorithms jsou dostupné operace z různých knihoven softwarů – QDAL/ORG, GRASS, Orfeo, QGIS, Saga a je možné vkládat i různé Python skripty nebo jiné, již existující modely. Spojovací oblé čáry se vykreslí automaticky ihned po nastavení vstupních parametrů do právě vložené operace. To jsou spojné čáry mezi vstupními daty a operacemi. Čáry uživatel nikdy nekreslí ručně pomocí myši, taková možnost ani není dostupná jako funkční ikona nebo volba z menu. Obdélník reprezentující výstupní data je přidán do modelu automaticky ihned po vložení operace, která data produkuje spolu s propojovací čárou mezi operací a výstupními daty.

Vizuální slovník se skládá ze tří obdélníkových symbolů, rozměrově shodných (Obr. 2). Obdélník s fialovou výplní reprezentuje vstupní data. Modrý obdélník reprezentuje výstupní data. Bílý obdélník reprezentuje operace. Zajímavé je použití vnitřních ikon. Použití ikon je obecně v rozhraní programů hojně využívané a je tak zajímavá aplikace ikon v tomto VPL.

Ikony použité v symbolech lze rozdělit do dvou skupin. První skupina jsou významové ikony, druhá skupina jsou operační ikony. Mezi významové ikony patří ikona PLUS u vstupních dat a ikona modré vodorovné šipky u výstupních dat. Obě ikony se nachází u levého okraje symbolů. Jejich použití lze pokládat za užitečné z hlediska zvýšení transparentnosti symbolů.



Obrázek 2. Základní symboly Processing Modeler (shora dolů): vstupní data, výstupní data, operace z knihovny QGIS.

Do druhé skupiny operačních ikon patří ikony u symbolu pro data a operace, které jsou úplně vpravo uvnitř obdélníku. Je to ikona křížku a tužky, které jsou umístěné pod sebou. Tyto ikony nemají za účel posílit vypovídající schopnost významu symbolu, ale slouží pro tvorbu modelu. Po kliknutí na ikonu křížku se symbol z modelu odstraní, tužka slouží pro editaci symbolu, resp. u funkce pro nastavení parametrů. Použití těchto ikon se jeví jako zbytečné, protože editace (nastavení parametrů) se obvykle v grafických programech děje po dvojkliku myší kdekoliv na symbol (to zde není implementováno). Funkce odstranění prvku lze provést alternativně z kontextového menu po kliku myší na prvek. Běžně je i zažitá možnost výběru symbolu a stisknutí klávesy Delete (ale to zde není také

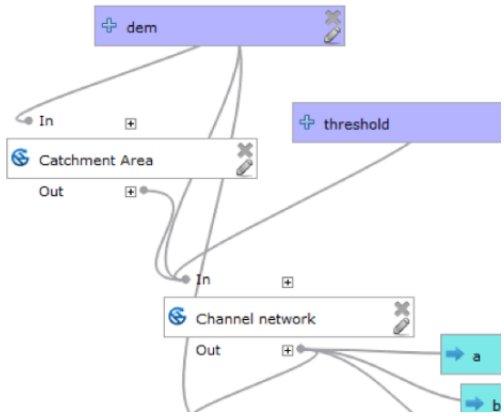
implementováno). Odstranění ikon vpravo ze symbolu by nabídlo větší prázdný prostoru uvnitř obdélníku, který pak lze dobře využít pro delší textový popis symbolu a tím zvýšení čitelnosti buď názvu dat nebo názvu operace, který je dost často automatizovaně zkracován v rozhraní.

Bílé obdélníky operace obsahují vlevo také významovou ikonu a to ikonu zdrojové knihovny. Na Obr. 2 je to ikona programu QGIS, protože funkce Zonal Statistics je použita z programu QGIS. Ikona tedy není stále stejná, ale závisí na zdrojové knihovně.

Specifická ikona je určena pro Python skript anebo jiný vložený model – podprogram (Obr. 3). Možnost vložit jiný existující model jako podprogram je poměrně nová funkce v Processing Modeleru, která byla implementována ve verzi QGIS 2.6. Brighton.

Obrázek 3. Ukázka symbolů funkcí s ikonou pro vnořený model (vlevo) a Python skript (vpravo).

Spojovací čáry patří nedílně do vizuálního slovníku. Spojovací čáry naznačují směr toku dat. V Processing Modeler jsou celkem netradičně použity oblé čáry. Čáry vychází vždy z delšího dolního okraje obdélníku vstupních dat. Čáry nejsou zakončeny šipkou u vstupu do operace, ale jsou zakončeny bodovým portem popsaným slovem In. Spojovací čára, která spojuje operaci a výstupní data, tak ta vstupuje do kratší levé strany obdélníku výstupních dat. Ani v jednom případě není u dat použit port.



Obrázek 4. Ukázka nevhodného křížení a překryvu spojných čar.

Při změně umístění symbolů v ploše modelu a jejich přeuspořádání se čáry automatizovaně překreslí. Dochází však dost často k překryvu oblých čar symboly, čáry se navíc kříží a je obtížné vysledovat, které symboly jsou spolu spojeny navzájem (Obr. 4). Navíc oblé čáry zabírají hodně místa v ploše modelu.

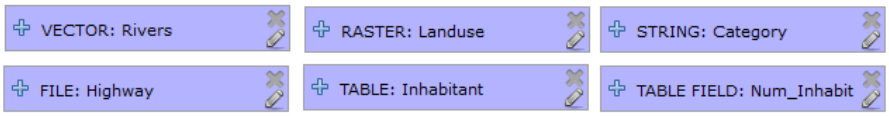
Správné čtení a vůbec sestavování diagramu usnadňuje, pokud je diagram zarovnán k mřížce a jednotlivé prvky mají mezi sebou stejné vzdálenosti. Bohužel, Processing Modeler nemá implementovanou funkci automatického zarovnání diagramu, tak jak je dostupná v jiných grafických editorech VPL pro GIS (např. ModelBuilder). Zarovnání diagramu závisí na ruční práci autora diagramu a jeho snaze, aby diagram byl maximálně přehledný. S přehledností souvisí i to, že ve vizuálním slovníku chybí symbol začátku a konce diagramu, tak jak je známý symbol terminátoru (Start, Konec) z klasického vývojového diagramu.

2. Hodnocení podle principů Physics of Notations

Nástrojem pro objektivní hodnocení vizuální notace poskytuje teorie Dr. Daniela Moodyho s názvem Physics of Notations [5]. Tato teorie stanovuje celkem devět principů. Jsou to principy: princip sémiotické čistoty, princip perceptuální rozlišitelnosti, princip vizuální expresivnosti, princip duálního kódování, princip sémantické transparentnosti, princip ekonomie grafiky, princip řízení složitosti, princip kognitivní integrace, princip kognitivní vhodnosti. Definice a uplatnění každého principu umožňuje stanovit úroveň grafické notace VPL z hlediska kognitivně efektivní notace. Snaha o kognitivně efektivní notaci znamená cíl disponovat s notací, která umožňuje vytvářet diagram, který je rychle a bezchybně čitelný, pochopitelný a zapamatovatelný. Podle této teorie lze hodnotit i různé vizuální programovací jazyky v GIS. Ukázka hodnocení pro ModelBuilder lze nalézt v článku [1].

Při hodnocení podle jednotlivých principů bylo zjištěno přetížení obou symbolů pro vstupní a výstupní data. Toto je výsledek aplikace principu Sémiotické čistoty, kdy není splněn princip shody 1:1 mezi prvky sémantického modelu a grafickými symboly syntaxe. Oba symboly dat slouží pro jakýkoliv typ dat. Nelze v modelu při čtení rozlišit, jaký typ dat symboly představují (vektor, rastr, boolean, table, point, line, polygon, string,...), i když již při vkládání dat se musí nejprve určit o jaký typ vstupních dat se jedná. Řešení jsou možná dvě. První možnost je rozlišit jednotlivý druh dat barvou výplně obdélníku. Toto řešení však vede k navýšení celkového počtu symbolů na 12 symbolů vstupních dat a jeden symbol pro operace. To však odporuje dalšímu z principů a to principu Ekonomie grafiky, který říká, že lidská mysl je schopná rozlišit kolem sedmi symbolů (plus mínus dva). Původní notace se třemi symboly tomuto principu vyhovuje. Druhé řešení je doplnění textu o druhu dat před vlastní název parametru. Momentálně toto řešení může aplikovat uživatel ručně zadáním textu před vlastní název parametru. Samozřejmě by bylo lepší, kdyby se toto doplňovalo automaticky po volbě

druhu vstupních dat. Ukázka je na Obr. 5. Doplnění textu vyhovuje jednomu z dalších principů a to principu Duálního kódování, kdy doplňující text dodává další informaci (dualita je zde ve smyslu doplnění barvy jako základního významu symbolu o duální text).



Obrázek 5. Ukázka úpravy názvu dat o text specifikující typ dat (velkými písmeny).

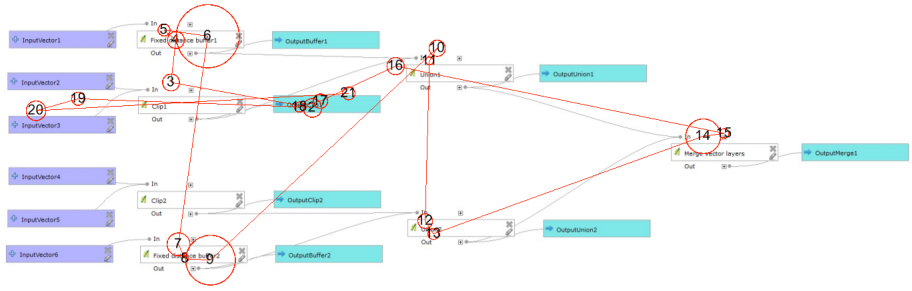
Princip Vizuální expresivnosti měří vizuální rozdílnost všech grafických symbolů v notaci. Notace Processing Modeleru používá pouze jednu vizuální proměnnou k zakódování informace, a to je barva výplně obdélníkového symbolu [3]. Barva je jednou z nejvíce kognitivně efektivních vizuálních proměnných a je správné, že je použita. Lidský zrak je velmi citlivý na změnu barvy. Změny barev jsou zjišťovány třikrát rychleji než změny tvaru a také jsou často snadněji zapamatovatelné. Tvar je druhou možnou vizuální proměnnou, není však tak kognitivně efektivní proměnnou. Z hlediska principu Vizuální expresivnosti se jedná o jednorozměrnou notaci (použita pouze barva, nikoliv tvar, orientace, velikost, textura, sytost).

Dalším principem je princip Perceptuální rozlišitelnosti, který nabádá využít maximum vizuálních proměnných. Rozlišitelnost je primárně stanovena vizuální vzdáleností mezi jednotlivými grafickými symboly. Ta je měřena podle počtu vizuálních proměnných, které jsou u symbolů použity, a také pomocí velikostí vizuálních vzdáleností mezi jednotlivými grafickými symboly. Obecně platí, že čím větší je vizuální vzdálenost mezi grafickými symboly, tím rychleji a přesněji jsou grafické symboly rozeznávány. Ze všech vizuálních proměnných má v rozlišování grafických symbolů speciální roli tvar, který představuje základní proměnnou, pomocí které identifikujeme běžné objekty v reálném světě. U symbolů Processing Modeleru je použita ze všech vizuálních proměnných pouze barva výplně. Problematická je bílá barva symbolu operace. Zde je špatná rozlišitelnost od bílé barvy pozadí plochy, kde se kreslí model. Zlepšení by přinesla jiná barevná výplň symbolu, např. žlutá barva. Byl by stále zachován dobrý kontrast černého textu na žlutém pozadí. Princip Řízení složitosti poukazuje na špatnou orientaci autora nebo čtenáře v rozsáhlém modelu s velkým počtem symbolů. Pro efektivnější kognici složitých modelů musí grafická notace poskytovat mechanismy pro modularizaci (dělení na části - podprogramy) a hierarchickou strukturalizaci. Modularizace je zde možná díky možnosti vložení jiného modelu jako podprogramu. Hierarchické členění není dost dobře možné, není implementováno.

3. Eye-tracking testování

Katedra geoinformatiky disponuje eye-tracking laboratoří pro snímání pohybu očí. Laboratoř je vybavena zařízením SMI RED 250 s programem SMI Experiment Suite 360 pro sestavení experimentu. Bylo nachystáno 11 modelů z Processing Modeleru pro část volného prohlížení a 23 modelů, u kterých respondenti řešili úkol. Jednalo se převážně o označování prvků v modelu. Testováno bylo 22 studentů z navazujícího magisterského studia Geoinformatika.

Software SMI BeGaze dodávaný s eye-trackerem umožňuje po otestování respondentů následovné přehrání a zpracování zaznamenaných pohybů očí. Naše zpracování bylo prováděné v Open-Source programu OGAMA. Základním výstupem z eye-tracking měření je zobrazení pohybu očí ve formě fixací oka na stimulu a pohybů očí mezi fixacemi (sakády), dohromady se jedná o tzv. scan-path. Doba fixace je zobrazena kružnicí, jejíž poloměr znázorňuje délku fixace. Číslo fixace znázorňuje pořadí fixace. Takto lze zobrazit každého respondenta individuálně (Obr. 6). Z eye-tracking měření se zjistilo, že u části volného čtení je způsob čtení diagramu převážně zleva doprava podle orientace toku dat.



Obrázek 6. Model s fixacemi a sakádami z eye-tracking testování od jednoho respondenta.

Druhou možností grafického vyhodnocení je zobrazení tzv. mapy pozornosti. Toto zobrazení umožňuje zobrazit více respondentů zároveň.

Dále je možné provést statistické vyhodnocení dat z eye-tracking metrik. Jedná se zejména o vyhodnocení celkového času, jak dlouho každý respondent řešil zadaný úkol nad modelem. Dále jsou to počty fixací nad stimulem, délka scanpath, dále čas průměrné délky fixace atd. Při vyhodnocení těchto metrik pro 22 respondentů lze již porovnávat různé uspořádání diagramů a vnímání jednotlivých prvků. Prvním indikátorem u úkolů je počet správných a špatných odpovědí. Při nezarovnaném uspořádání byl vyšší počet špatných odpovědí na otázky z důvodu překryvu spojných čar. Při statistickém vyhodnocení pomocí Kruskal-Wallisova testu byl testován celkový čas řešení úloh s vertikálním, horizontálním, diagonálním a chaotickým uspořádáním. U metriky celkového času nevyšel signifikantní rozdíl mezi časy pro různá uspořádání diagramů. Signifikantní rozdíl vyšel pouze

u metrik Scanpath Lenght a Fixation Count, kde horizontální uspořádání má nižší hodnoty. Na základě tohoto zjištění lze říci, že pro čtení a kognici diagramů je lepší horizontální uspořádání diagramu než vertikální uspořádání.

4. Závěr

Tento článek upozorňuje na některé přednosti a možnosti vylepšení grafického slovníku vizuálního programovacího jazyka komponenty Processing Modeler v software QGIS. Slovník obsahuje pouze tři symboly, dva pro data a jeden pro operace. U symbolu vstupních dat a operace by bylo možné odstranit ze symbolů ikonu tužky a křížku, které slouží pouze v režimu návrhu. Ušetřené místo lze využít pro delší textový popis, zejména typu dat. Tím by se vyřešila i přetíženost symbolu dat. Návrh vyplývající z hodnocení pomocí teorie Physics of Notation otvírá i diskuzi, zda nepoužít pro operace jiný tvar symbolu než je obdélník, neboť vizuální proměnná tvar není vůbec využita. Bílá výplň obdélníku symbolu operace navíc splývá s bílou barvou pozadí. Jako vhodný návrh se jeví světle žlutá barevná výplň.

Z hlediska orientace diagramu se jeví lepší horizontální orientace.

Poděkování. Příspěvek byl podpořen projektem CZ.1.07/2.3.00/20.0170.

Reference

[1] Dobešová, Z.: *Using the Physics of notation to analyse ModelBuilder diagrams*, 13th International Multidisciplinary Scientific GeoConference, Albena, Bulgaria, 2013, 595–602 s., DOI: 10.5593/SGEM2013/BB2.V1/S08.039.

[2] DOBEŠOVÁ, Z.: *Přehled grafických notací diagramů toků dat v GIS a metody hodnocení*, Sborník sympozia GIS Ostrava 2015, 2015, VŠB-TU, Ostrava, ISBN 978-1-4799-7738-3.

[3] HRIC, F.: *Hodnocení QGIS Processing Modeler podle principů fyzické notace*, bakalářská práce, Univerzita Palackého, Olomouc, 2015.

[4] LANDA, M. – FURKEVIČOVÁ, L.: *wxGUI Graphical Modeler*, `https://grass.osgeo.org/grass70/manuals/wxGUI.gmodeler.html`.

[5] MOODY, D. L.: *The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering*, IEEE Transactions on Software Engineering, 2009, **35**(6), 756–779, DOI:10.1109/tse.2009.67.

[6] QGIS: *Changelog for QGIS 2.0*, 2013, `http://www.qgis.org/en/site/forusers/visualchangelog200/index.html#feature-processing-modeller`.

Kontaktní adresa

Ing. Zdena Dobešová, PhD., Katedra geoinformatiky, Přírodovědecká fakulta, Univerzita Palackého, 17. listopadu 50, 779 00 Olomouc, Česká republika,
E-mailová adresa: `zdena.dobesova@upol.cz`, `http://dobesova.upol.cz`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 43–60

TABULKY V CONTEXTU: PŘÍSTUPY, MOŽNOSTI, ALGORITMY

TOMÁŠ HÁLA (CZ)

Abstrakt. Sazba tabulek trvale patří mezi obtížnější prvky při zpracování publikací. Příspěvek přináší přehled možností sazby tabulek v systému ConTEXt. Zabývá se srovnáním dostupných způsobů, starších i současných, konkrétně se jedná o prostředí `table`, `tabulate`, `TABLE`, `xtables`, srovnává jejich možnosti vzájemně i s „konkurenčním“ LaTeXem.

Tabulky mohou i generovány z dat v jiném formátu, velmi často z formátu CSV. Proto příspěvek se dále zabývá i možnými přístupy tzv. databázového zpracování.

Dále budou předvedeny některé jednoduché algoritmy, kterými lze celkem snadno rozšířit stávající možnosti. Algoritmy budou prezentovány v jazyce Lua, jenž je součástí systému ConTEXt MkIV.

Klíčová slova. ConTEXt, LaTeX, srovnání, sazba.

TABLES IN CONTEXT: WAYS, POSSIBILITIES, ALGORITHMS

Abstract. In publication process, typesetting of tables belongs to the more complicated tasks. The paper reviews older as well as current possibilities of typesetting of tables in ConTEXt (environments `table`, `tabulate`, `TABLE`, `xtables`), and compares them mutually and with the „rival“ LaTeX.

Tables might be generated from other formats, one very frequently used is CSV. Therefore, the paper deals also with the database processing.

Finally, some simple algorithms for easy extensions of repertoir are presented.

Keywords. ConTEXt, LaTeX, comparison, typesetting.

Úvod

Srovnáme-li nástroje sazby tabulek napříč implementacemi TEXu, uvidíme značně odlišné přístupy. Základní plain nedisponuje specializovanými prostředky pro sazbu tabulek, což ostatně vyplývá i ze skutečnosti, že „bible“ TEXu [10] se samotnou sazbou tabulek zabývá až v kapitole 22 nazvané Alignment, a to v souvislosti s obecnou problematikou zarovnání sazebního materiálu.

Způsob zápisu v plainu – i přesto, že je ve své podstatě přímočarý a zcela logický – patrně nikdy nebude prohlášen za uživatelsky přívětivý, neboť vyžaduje poněkud hlubší znalost systému a jeho chování.

Požadavek na tvorbu jednoduššího uživatelského prostředí vyústil v 80. letech minulého století vznikem nadstavby LaTeX, která pro tabulkově orientovanou

sazbu používá dvě základní prostředí – `tabbing` napodobující systém tabelačních zarážek psacího stroje, jež je vhodné pro pořadovou sazbu, a `tabular` pro běžné tabulky.

Od 90. let minulého století je k dispozici další nadstavba – ConTEXt. Jedná se, stejně jako u TEXu samotného, o volně šiřitelný software určený pro sazbu dokumentů s požadavkem vysoké kvality výstupního produktu. Základní vlastnosti verze MkIV této nadstavby byly v představeny již dříve ([8, 9]).

ConTEXt: základní tabulky – `table`

Nejstarším způsobem sazby tabulek je prostředí `table`. Ve srovnání s LATEXem má prostředí `table` nejblíže ke známému prostředí `tabular`. Namísto znaku `&`, který v LATEXu slouží k oddělování sloupců, zde příkazem `\NC` zahajujeme sazbu nového sloupce (NC = new column).

Definice sloupců

K určení vlastností jednotlivých sloupců jsou k dispozici kódy podobné v prostředí tabular, pro sloupec lze však použít i více než jeden kód. Aby nedošlo k chybné interpretaci, svislá čára, v LATEXu používaná k aktivaci svislých linek, zde slouží pouze k oddělení definic jednotlivých sloupců. Kromě obdobných kódů jako v LATEXu (c, l, r, p) lze zadat i informaci o vyrovnání sloupců (s0, s1, s2, s3), kde číslo je faktorem velikosti mezisloupcových mezer. Zde uvedené příklady se vyrovnáním sloupců v tomto prostředí nezabývají.

Prostředí `table` je historicky nejstarším, avšak dosud funkčním způsobem sazby tabulek. Četnost jeho užití je dnes již nižší, vyspělejší prostředky (`TABLE`, `xtables`) nabízejí pohodlnější správu tabulek i s přístupem k jednotlivým buňkám.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LATEX:	tabbing
	tabular + halda balíčků
ConTEXt:	tabulate
	table, TABLE, xtables:
Lua:	neomezené možnosti:

Obrázek 1. Tabulka vytvořená prostředím `table`.

```
\starttable[|r|p(0.6\textwidth)|]\HL
\NC Podle úpravy:
       \NC pořadově sázené, otevřené, uzavřené, s členícími linkami\,\dots
              \NC\FR
\NC Podle účelu:
       \NC formuláře, bilance, knižní, časopisecké\,\dots
              \MR\HL
\NC  Podle nástroje:\NC ~\NC\FR\HL[2]
\NC \LaTeX: \NC tabbing\NC\MR
\NC \NC tabular + halda balíčků\NC\MR\HL
\NC \ConTeXt: \NC tabulate\NC\MR
\NC \NC table, TABLE, xtables:\NC\MR\HL
\NC Lua:\NC neomezené možnosti:\NC\MR\HL[3]
\stoptable
```

Obrázek 2. Zdrojový text tabulky vytvořené prostředím `table`.

Vyrovnávání řádků

Uvedená ukázka nemá správně vyrovnán první řádek, neboť všechny řádky jsou vyrovnávány příkazem `\MR`, sloužícím jako pro horní, tak pro dolní vyrovnání běžného řádku. Jiné způsoby vyvoláme příkazy `\SR` – rovněž horní i dolní, ale s využitím u oddělovacích řádků, `\FR` pouze pro horní vyrovnání, `\LR` pouze pro dolní vyrovnání a `\NR` bez vyrovnání ([15]).

V uvedené ukázce provedeme dvě změny ve vyrovnání řádků.

```
\starttable[|r|p(0.6\textwidth)|]
\NC Podle úpravy:
       \NC pořadově sázené, otevřené, uzavřené, s členícími linkami\,\dots
              \NC\FR  %%% Zde \FR
\NC Podle účelu:
       \NC formuláře, bilance, knižní, časopisecké\,\dots
              \VL\SR\HL  %%% Zde \SR
\NC  Podle nástroje\NC\VL\MR\HL[2]
\NC \LaTeX \NC tabbing\VL\MR
\NC \NC tabular + halda balíčků\VL\MR\HL
\NC \ConTeXt: \NC tabulate\VL\MR
\NC \NC table, TABLE, xtables\NC\MR\HL
\NC Lua:\VL neomezené možnosti\MR\HL[3]
\stoptable
```

Obrázek 3. Prostředí `table` – změna vyrovnání řádků.

Vyznačování sloupců, svislé linky

Pokud chceme, aby v daném místě byl sloupec ohraničen svislou linkou, použijeme namísto příkazu `\NC` příkaz `\VL` (vertical line). Tento způsob je však

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje	
LaTeX	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

Obrázek 4. Tabulka (`table`) se změnou vyrovnání řádků.

poněkud nepraktický, pokud sazeč nemá předem rozmyšleno, kde se budou linky nacházet a kde nikoliv.

Vodorovné linky

V předchozí ukázce je použit dosud nezmíněný příkaz `\HL` vytvářející vodorovné linky. Nepovinný parametr slouží k určení stupně tloušťky čáry. Přípustné hodnoty jsou 1, 2, 3. I přesto, že tento způsob je dosti omezující, pro běžnou práci postačuje.

ConTeXt: pořadová sazba – `tabulate`

Prostředí `tabulate` pro pořadovou sazbu vychází z prostředí `table` – s tabulkou pracujeme úplně stejně (způsob definice sloupců, ohraničení sloupců atd.), máme však k dispozici širší repertoár formátovacích povelů a parametrů. Toto prostředí lze využít pro tabelaci. Podrobněji viz například [15]. Další inspiraci lze nalézt například v [12] a v [13].

ConTeXt: Natural Tables – `TABLE`

Přirozené tabulky vznikly o něco později a autor ConTeXtu Hans Hagen se inspiroval jazykem HTML. Z následujícího příkladu je patrné, jak se jednotlivé značky používají.

```
\bTABLE
\bTR\bTD Česká republika\eTD\bTD CZ \eTD\bTD 10 500 000\eTD\eTR
\bTR\bTD Slovenská republika \eTD\bTD SK \eTD\bTD 5 410 000\eTD\eTR
\bTR\bTD Polská republika\eTD\bTD PL \eTD\bTD 38 500 000\eTD\eTR
\eTABLE
```

Obrázek 5. Zdrojový kód tabulky.

Česká republika	CZ	10 500 000
Slovenská republika	SK	5 410 000
Polská republika	PL	38 500 000

Obrázek 6. Výsledný tvar tabulky s implicitním nastavením.

Vidíme zde náhradu slov `start` a `stop` za `b` a `e` oproti běžným zvyklostem v ConTEXtu, ale i přesto je k dispozici mocný příkaz `\setupTABLE`, kterým lze elegantně nastavit řadu vlastností tabulce jako celku i jejím jednotlivým částem. Implicitní nastavení obsahuje mj. zarovnání k levému okraji a nestejné vnitřní okraje.

TABLE – nastavení vlastností

Příkaz `\setupTABLE` má velmi pružnou syntaxi. Můžeme jím nastavovat vlastnosti celé tabulky, jednotlivých řádků, sloupců či buněk. Zápis je poměrně pohodlný a především se jedná o koncepční přístup při definici vlastností sazby.

Následující ukázky postupně předvedou vybrané vlastnosti z velmi rozsáhlé množiny. Pro lepší pochopení uveďme, že prvním parametrem udáváme zda definujeme sloupce (c) či řádky (r), druhý parametrem číslo řádku, případně můžeme použít označení first, last a třetí parametr pak obsahuje nastavení vlastností. Speciální případem je situace, kdy první parametr obsahuje číslo. Tím říkáme, že určujeme vlastnosti konkrétní buňky či buněk. První parametr je pak chápan jako sloupcová souřadnice, druhý jako řádková.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTEXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[c][2][align=middle]
```

Obrázek 7. Tabulka s centrovaným druhým sloupcem.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[r][odd][offset=-2dd]
```

Obrázek 8. Tabulka se záporným vnitřním okrajem.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[r][1,last][background=color,
                       backgroundcolor=yellow]
```

Obrázek 9. Tabulka s barevným pozadím prvního a posledního řádku.

Zarovnání čísel v prostředí `TABLE`

S požadavkem na zarovnání hodnot na desetinnou čárku (nebo tečku či jiný znak) se sazeči setkávají poměrně často. Nejedná o zcela triviální problém, avšak prostředí `TABLE` nabízí relativně jednoduché řešení, jak ukazují poslední dva příkazy v následující ukázce kódu. Zbývající příkazy zajišťují vodorovné i svislé linky.

Další příklady lze nalézt například v [2].

ConTeXt: Extreme Tables – `xtables`

Jedná se o nejnovější a doporučovaný nástroj, který rozšiřuje možnosti tabulkové sazby o další nástroje [3]. Koncept strukturních značek odpovídá logice prostředí `table`. Rozsah tohoto příspěvku bohužel neumožňuje představit zde všechny zajímavé vlastnosti.

Podle úpravy:	pořadově sázené, otevřené, uzavřené, s členícími linkami ...
Podle účelu:	formuláře, bilance, knižní, časopisecké ...
Podle nástroje:	
LaTeX:	tabbing
	tabular + halda balíčků
ConTeXt:	tabulate
	table, TABLE, xtables
Lua:	neomezené možnosti

```
\setupTABLE[1][4,6][align=left,
                    background=color,backgroundco]
```

Obrázek 10. Tabulka s barevným pozadím u vybraných buněk.

Málo:	Hodně
10.000 :	1,0
−10.00 :	10,000000
1000.0000:	−1000,000

```
\setupTABLE[frame=off]
\setupTABLE[column][first][leftframe=on]
\setupTABLE[column][last][rightframe=on]
\setupTABLE[row][first][topframe=on]
\setupTABLE[row][first,last][bottomframe=on]
\setupTABLE[column][1][alignmentcharacter={.},
                      aligncharacter=yes,align=middle]
\setupTABLE[column][2][alignmentcharacter={,},
                      aligncharacter=yes,align=middle]
```

Obrázek 11. Tabulka se čísly zarovnanými podle požadavku.

```
\startxtable[offset=1cm]
 \startxrow
  \startxcell one
  \stopxcell
  \startxcell two
  \stopxcell
 \stopxrow
 \startxrow
  \startxcell alpha \stopxcell
  \startxcell beta \stopxcell
 \stopxrow
\stopxtable
```

Obrázek 12. Ukázka jednoduché tabulky – prostředí xtables (Převzato z: [3]).

Zpracování CSV

Ve všech dosud uvedených příkladech jsme předpokládali, že data jsou součástí zdrojového textu sázeného v jazyce ConTEXt, případně vložená příkazem `\input`

z jiného souboru. Často se však lze setkat se situací, kdy máme data ve formátu CSV a nechce se nám tato data ručně ani automatizovaně značkovat.

Formát CSV

CSV (comma separated values) je formát uložení dat v souboru, při kterém řádky představují záznamy a v rámci řádků jsou jednotlivé hodnoty jednotně odděleny čárkou jakožto dohodnutým oddělovačem Formát CSV není popsán žádnou normou, existuje k němu však specifikace ([17]). Dnes se název formátu zobecňuje na všechna podobná data nezávisle na zvoleném oddělovacím znaku. Někteří autoři (např. [11, 16]) však používají jiná označení, např. obecné DSV (delimiter separated values) nebo konkrétnější TSV (tabulator separated values).

Modul `database`

Modul database (autor Hans Hagen, soubor `m-database`) je dostupný například v distribuci TEXlive nebo na stránkách ConTEXtu ([1]). Dostupná verze z roku 2011 nese označení 1.001 a není známo, zda je modul dále vyvíjen či nikoliv. Lze jej použít, avšak s nepříliš podrobnou dokumentací.

Miklavec ([14]) uvádí dosud neřešený rozpor se specifikací CSV [17], podle níž může buňka, je-li správně označena uvozovkami, vést přes více řádků. Toto však není implementováno a výsledná tabulka je chybně formátována.

Připojení modulu, definice čtení a zobrazení tabulky

Modul `database` připojíme v preambuli, a to obvyklým způsobem:

```
\usemodule[database]
```

Poté je potřeba uvést, jak se mají data správně načíst a jak má být tabulka sázena na výstupu. Obojí definujeme parametry příkazu `\defineseparatedlist`, jehož prvním parametrem pojmenujeme zapsané vlastnosti a zároveň, jak ukážeme dále, vytváříme speciální příkazy, kterými ohraničujeme oblast formátu CSV.

```
\defineseparatedlist
  [CSV]
  [separator=;,
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
   left=\bTD,right=\eTD]
```

Obrázek 13. Běžná definice pro konverzi formátu CSV do prostředí `TABLE`.

Nyní předpokládejme data ve formátu CSV se čtyřmi řádky a pěti sloupci (obr. 14).

a	b	c	d	e	
a	"b"	"c"	"d"	"	e"
A	B	C	D	E	
A	"B"	"C"	"D"	"	E"

```
a;b;c;d;e
a;"b";"c";"d";";e"
A;B;C;D;E
A;"B";"C";"D";";E"
```

Obrázek 14. Data ve formátu CSV se čtyřmi řádky a pěti sloupci.

Uvedená ukázka však není dokonalá, uvozovky, které ohraničují řetězce, nebyly vzaty v potaz. Závadu napravíme doplněním parametru `quotechar` v příkazu `\defineseparatedlist` (obr. 15):

a	b	c	d	e
a	b	c	d	;e
A	B	C	D	E
A	B	C	D	;E

```
\usemodule[database]
\defineseparatedlist
  [CSV]
  [separator=;,
   quotechar=",
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,
   left=\bTD,right=\eTD]
```

Obrázek 15. Data ve formátu CSV se čtyřmi řádky a pěti sloupci – náprava.

To, že jsme v definici uvedli příkazy z prostředí `TABLE`, nám nyní umožňuje tabulky dále formátovat a graficky upravit způsobem, který byl již zmíněn.

Dva způsoby psaní příkazu

Jak již bylo zmíněno, příkazem `\defineseparatedlist[CSV][...]` definujeme chování prostředí pro sazbu tabulky dat ve formátu CSV, které se uplatní mezi `\startseparatedlist[CSV]` a `\stopseparatedlist`. Můžeme však také použít stručnější způsob:

```
\startCSV
a;b;c;d;e
a;"b";"c";"d";";e"
A;B;C;D;E
A;"B";"C";"D";";E"
\stopCSV
```

Konverze LaTeXové tabulky do ConTEXtu

Před více než 15 lety autor tohoto příspěvku sázel v LaTeXu. Jedním z vysázených děl byla učebnice o radioaktivitě [7], jejíž autor nyní připravuje nové vydání. Je potřeba s co nejmenší námahou tyto LaTeXové tabulky převést, resp. použít v ConTEXtu.

```
\begTAB{Přehled nuklidů transfermiových prvků.}{trfe}{%
\begin{center}\setlength{\tabcolsep}{1mm}\def\arraystretch{1.2}
\let\hp=\hphantom
\small\sf\begin{tabular}{|c|l|l|c|l|} \hline
%--
\podpěra Z
    & známé              & izotop       & T (s)               & reakce\\
    & izotopy (A)        & s-nejdelším T&                     &       \\[2.5mm]\hline\hline
101 & 248--259           & \(^{258}\)Md & 55 dní              & \(^{255}\)Es(\(\alpha\),n)\\\hline
102 & 250--259           & \(^{255}\)No & 185\hp{,00000}      & \(^{244}\)Pu(\(^{16}\)O,-5n)\\\hline
103 & 252--262           & \(^{256}\)Lr & \hp{0}45\hp{,00000} & \(^{243}\)Am(\(^{18}\)O,-5n)\\\hline
104 & 253--262           & \(^{261}\)Rf & \hp{0}65\hp{,00000} & \(^{248}\)Cm(\(^{18}\)O,-5n)\\\hline
105 & 255--258, 260--263 & \(^{262}\)Db & \hp{0}34\hp{,00000} & \(^{249}\)Bk(\(^{18}\)O,-5n)\\\hline
106 & 258--261, 263      & \(^{263}\)Sg & \hp{00}0,9\hp{0000} & \(^{249}\)Cf(\(^{18}\)O,-4n)\\\hline
107 & 261, 262, 264      & \(^{262}\)Bh & \hp{00}0,0061\hp{0} & \(^{209}\)Bi(\(^{54}\)Cr,-2n)\\\hline
108 & 264, 265, 267, 269 & \(^{269}\)Hs & \hp{0}19,7\hp{0000} & produkt \(\alpha\)
                                                              přeměny \(^{273}\)110\\\hline
109 & 266, 268           & \(^{266}\)Mt & \hp{00}0,0034\hp{0} & \(^{209}\)Bi(\(^{59}\)Fe,-n)\\\hline
110 & 269, 271--273      & \(^{269}\)110& \hp{00}0,0027\hp{0} & \(^{208}\)Pb(\(^{62}\)Ni,-n)\\\hline
111 & 272                & \(^{272}\)111& \hp{00}0,0015\hp{0} & \(^{209}\)Bi(\(^{64}\)Ni,-n)\\\hline
112 & 272                & \(^{272}\)112& \hp{00}0,00028      & \(^{208}\)Pb(\(^{70}\)Zn,-n)\\\hline
%-+
\end{tabular}\end{center}}{}
\endTAB
```

Obrázek 16. Ukázka zdrojového textu původní tabulky.

Na tabulku zapsanou v LaTeXu může určitým způsobem pohlížet také jako na data typu CSV. Oddělovačem sloupců je znak `&` a pokud fyzicky tabulku uspořádáme po řádcích, nic nebraní použití modulu `database` pro zpracování dat tabulky.

```
\defineseparatedlist[LTX][separator=&,
   before=\bTABLE,after=\eTABLE,
   first=\bTR,last=\eTR,left=\bTD,right=\eTD,
   setups=unix
]
```

Obrázek 17. Definice vstupních a výstupních vlastností pro konverzi.

V původním vydání bylo matematické prostředí sázeno dvojicí příkazů `\(` a `\)`, tyto příkazy je totiž možno v LaTeXu předefinovat podle vlastních představ. V ConTeXtu však tato dvojice příkazů není implementována. Abychom nemuseli tyto příkazy ručně nebo jinak nahrazovat za tradiční `$`, vytvoříme si jejich vlastní definici, v níž, za pomoci jazyka Lua, vysázíme „dolary“.

```
\def\hline{}
\def\\{}
\def\({\startluacode context("$") \stopluacode}
\def\){\startluacode context("$") \stopluacode}
```

Obrázek 18. Předefinování LaTeXových příkazů.

Zároveň s tím se elegantně zbavíme nežádoucích LaTeXových příkazů, kterými jsou data v tabulkách hojně opatřena.

```
%\begTAB{Přehled nuklidů transfermiových prvků.}{trfe}{%
%\begin{center}\setlength{\tabcolsep}{1mm}\def\arraystretch{1.2}
%\small\sf\begin{tabular}{|c|l|l|c|l|} \hline
%~-
\setupTABLE[column][each][align={low,middle}]
\startLTX
 Z   & známé       & izotop  & T (s)  & reakce\\
     & izotopy (A) & s~nejdelším T&&\\[2.5mm] \hline\hline
101 & 248--259          & \(^{258}\)Md & 55 dní
       &             \(^{255}\)Es(\(\alpha\),n)\\ \hline
102 & 250--259          & \(^{255}\)No & .185\hphantom{,00000}
       &             \(^{244}\)Pu(\(^{16}\)O,~5n)\\ \hline
...
\stopLTX
%~+
%\end{tabular}\end{center}}{}
%\endTAB
```

Obrázek 19. Zdrojový text tabulky nuklidů transfermiových prvků.

Nevyhnuli jsme se ruční editaci, neboť je potřeba ještě deaktivovat použitá LaTeXová prostředí a naopak aktivovat prostředí ConTEXtové. I přesto, že je možné i toto řešit programově, jeví se ruční editace zatím i při desítkách tabulek rychlejší.

Z	známé	izotop	T (s)	reakce
	izotopy (A)	s nejdelším T		[2.5mm]
101	248–259	^{258}Md	55 dní	^{255}Es(α,n)
102	250–259	^{255}No	185	^{244}Pu(^{16}O, 5n)
103	252–262	^{256}Lr	45	^{243}Am(^{18}O, 5n)
104	253–262	^{261}Rf	65	^{248}Cm(^{18}O, 5n)
105	255–258, 260–263	^{262}Db	34	^{249}Bk(^{18}O, 5n)
106	258–261, 263	^{263}Sg	0,9	^{249}Cf(^{18}O, 4n)
107	261, 262, 264	^{262}Bh	0,0061	^{209}Bi(^{54}Cr, 2n)
108	264, 265, 267, 269	^{269}Hs	19,7	produkt α přeměny ^{273}110
109	266, 268	^{266}Mt	0,0034	^{209}Bi(^{59}Fe, n)
110	269, 271–273	^{269}110	0,0027	^{208}Pb(^{62}Ni, n)
111	272	^{272}111	0,0015	^{209}Bi(^{64}Ni, n)
112	272	^{272}112	0,00028	^{208}Pb(^{70}Zn, n)

Obrázek 20. Přehled nuklidů transfermiových prvků po vysázení ConTEXtem bez dalších typografických úprav.

Zpracování dat z externího souboru

Dosavadní ukázky vycházely z předpokladu, že data jsou součástí zdrojového textu. Získat data z externího souboru je však také možné:

```
\processseparatedfile[TSV][filename]
```

ScanCSV a HandleCSV

Problematikou zpracování souborů ve formátu CSV, zejména pro vlastní školské potřeby, se zabývá Jaroslav Hajtmar. Na konferencích TEXperience v r. 2012 a ConTEXt Meeting v r. 2013 prezentoval svou knihovnu ScanCSV ([6], soubor `t-scancsv.lua`). Dnes však tuto knihovnu sám autor již považuje za zastaralou a nahradil ji novější verzí nazvanou HandleCSV, která se skládá ze dvou souborů `t-handlecsv.lua` a `t-handlecsv-tools.lua`. Na jejím současném vývoji se podílí dále Pablo Rodriguez. Zatím se jedná však spíše o soukromý projekt, který není veřejně prezentován a není zveřejněna ani dokumentace.

Na jednoduchém příkladu se můžeme podívat, jak lze s knihovnou HandleCSV můžeme pracovat.

"Name";"Surname";"Birthdate" "John";"Smith";10/03/02 "Jane";"Newman";03/03/92	John Smith was born on 10/03/02. Jane Newman was born on 03/03/92.

Obrázek 21. Vstupní data a požadovaná výstupní sestava.

```
\usemodule[handlecsv]
\setheader
\opencsvfile{a.csv}
\starttext
  \startbuffer[loop]
    \cA\ \cB\ was born on \cC.\crlf
  \stopbuffer
   \doloop{\getbuffer[loop]%
          \nextrow\ifEOF\exitloop\fi}
\stoptext
```

Obrázek 22. Zdrojový kód k ukázce použití knihovny `HandleCSV`.

Tabulkový procesor – modul `spreadsheet`

Tento modul (autor Hans Hagen, soubor `m-spreadsheet`) umožňuje podle očekávání provádět jednoduché výpočty. Je dostupný v distribuci TEXlive i na

ConTEXt Garden. Modul je udržován, ale jeho další vývoj neprobíhá. Dokumentace je aktualizována, ale delší dobu nebyla zařazována do distribuce pro drobnou technickou závadu ([5]).

Příkazy `\startcell` a `\stopcell` ohraničují prostředí buňky, do níž se nepíše text k sazbě v jazyce TEX, ale podle syntaxe jazyka Lua, v němž jsou příslušné výpočty vyhodnocovány. Jedná se o velmi zajímavé a elegantní řešení, avšak méně pohodlné, pokud sázená tabulka obsahuje větší množství „TEXového“ textu a méně výpočtů, neboť všechny texty je potřeba opatřit uvozovkami. Autor modulu sám doporučuje jeho použití jen pro menší výpočty, například faktury [4]. Jeho příklad v drobné úpravě uvádíme i zde.

```
\startspreadsheettable[test][frame=off]
 \startrow
  \startcell[align=flushleft,width=8cm] "První položka" \stopcell
  \startcell[align=flushright,width=3cm] @ "0.2f EUR" 3.50 \stopcell
 \stoprow
 \startrow
  \startcell[align=flushleft] "Druhá položka" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" 8.45 \stopcell
 \stoprow
 \startrow
  \startcell[align=flushleft] "Třetí položka" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" 5.90 \stopcell
 \stoprow
 \startrow[topframe=on]
  \startcell[align=flushleft] "VAT 21\percent" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" 0.21 * sum(B) \stopcell
 \stoprow
 \startrow[topframe=on]
  \startcell[align=flushleft] "\bf Celkem" \stopcell
  \startcell[align=flushright] @ "0.2f EUR" sum(B) \stopcell
 \stoprow
\stopspreadsheettable
```

Obrázek 23. Ukázka použití modulu `spreadsheet`.

První položka	
Druhá položka	8.45 EUR
Třetí položka	5.90 EUR
VAT 21%	3.75 EUR
Celkem	21.60 EUR

Obrázek 24. Výsledná tabulka obsahující část faktury.

Vlastní řešení

Nyní zde předvedeme speciální řešení zhotovení pro jednu zakázku obsahující ekonomická data, která mají být prezentována jednak tabulkou, jednak grafem.

Zákazník však zpočátku nebyl schopen přesně formulovat požadavky, nebylo jasné, jakou formou se budou data prezentovat, ani jak dlouhá časová řada má být prezentována. K tomu přistoupily i požadavky na snadnou správu, konkrétně obecnou datovou strukturu, s pružným přístupem, indexovatelnou a také editace zdrojového textu musí být jednoduchá.

```
\tabulka[][title="Pracoviště A",
data=
RS 2517000 2515000 2386000 1954000
PU 4.57 4.62 4.65 4.05
PKL 361 491 392 268
PK 6699 5508 9475 6012
]
```

Obrázek 25. Příklad dat.

Původní úvaha spočívala v načítání dat do dvourozměrného pole s oddělovačem řádků `<lf>`. Pokud jsou však data předána ke zpracování ve formě atributu v parametrech, jsou součástí zdrojového kódu, a tudíž znak `<lf>` se změní na mezeru. Pro zjednodušení implementace bylo načítání doplněno „natvrdo“ o konstantu udávající počet sloupců, toto řešení bylo možno použít proto, že všechny tabulky měly stejný počet sloupců.

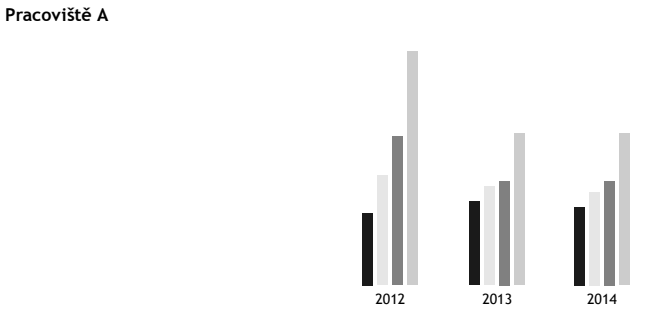
```
\def\tabulka[#1][#2]{
  \ctxlua{userdata.tabulka('#1','#2')}}
```

Obrázek 26. Ukázka propojení ConTEXtového makra a funkcí napsanou v jazyce Lua.

```
\def\grafwidth{0.5}

\def\graf#1#2{%
  \def\koef{\csname#1koef\endcsname}
  \startcolor[#1color]
  \vrule width\grafwidth cc height #2dd\
  \stopcolor
}
```

Obrázek 27. Definice potřebné pro vykreslení sloupcového grafu.

Zdrojový kód celé funkce je uveden na konci příspěvku. Protože se jedná o starší projekt, naleznete v kódu makra pro generování tabulky starším způsobem, tj. pomocí prostředí `table`.



Rozpočet služby	1 786 000	2 100 000	1 954 000
Počet úvazků	5,65	5,05	4,75
Počet klientů	392	268	268
Počet úkonů	9 475	6 012	6 012

Obrázek 28. Výsledná tabulka i se sloupcovým grafem.

Projekt CTX-CSV

Další specifické potřeby v oblasti zpracování dat ve formátu CSV vedly autora tohoto příspěvku k zahájení vývoje vlastních nástrojů, zejména s výstupem ve formě grafů, pro což není v ConTEXtu zatím kompaktní a uživatelsky přívětivá podpora. Vzhledem k probíhajícímu vývoji není projekt zatím zveřejněn.

Skládá se ze tří částí – `csv-tools.lua`, obsahující vstupně výstupní operace pro zpracování souboru ve formátu CSV, `csv-stat.lua`, zabývající se grafickou reprezentací načtených dat, a soubor `t-csv.mkiv`, který slouží jako modul. Grafy jsou vykreslovány MetaPostem, výpočty zajišťují úseky kódy v jazyce Lua.

```
\CSVRead[comma2dot][data=\soubor,separator=|]
\CSVWrite[dot2comma][output=pomout.csv,enclosechar="]
\CSVTABLE[dot2comma][]
```

Obrázek 29. Základní vstupně-výstupní operace se souborem ve formátu CSV.

Závěr

ConTEXt nabízí zajímavé možnosti sazby tabulek. Představené ukázky vycházejí jednak z řešení čistě „TEXových“, tj. ze sady strukturních značek, jednak z možností, jež přináší jazyk Lua.

I přesto, že máme k dispozici vícero možností, základní vlastnosti jsou integrovány do jednoho celku, což zbavuje uživatele nutnosti připojovat různé balíčky, jako je tomu v LATEXu, z nichž některé nejsou ani vzájemně kompatibilní, což bylo dokumentováno v [18].

```
function vypocet(t,c)
  keyword_options = utilities.parsers.settings_to_array(keywords)
  named_values = utilities.parsers.settings_to_hash(keyvals)
  ...
  if t[r][c+3]>0 then t[r][c+10]=t[r][c+3] end
  if t[r][c+3]>1 then t[r][c+11]=t[r][c+3] end
  if t[r][c+3]>2 then t[r][c+12]=t[r][c+3] end
  if t[r][c+3]>3 then t[r][c+13]=t[r][c+3] end
  ...
 return t
end
```

Obrázek 30. Ukázka části funkce, která reprezentuje tabulkové výpočty v načtených datech a která je užita v příkazu \CSVNewColumn.

```
\CSVNewColumn[][table=1,column=23,expr=vypocet]

\CSVComputeColumn[][table=1,column=23,startrow=2,meze={0,13}]
\CSVDrawDistribution[][table=1,column=23,maxparts=12]

\CSVComputeCorrelation[][table=1,columns={23,18},startrow=2]%,meze={0,60}]
\CSVDrawCluster[][table=1,xdraw=19.2,columns={23,18},startrow=2]%,groupby=2]
```

Obrázek 31. Ukázka příkazů, sloužících k výpočtům a k vykreslení grafů.

V oblasti zpracování formátu CSV však různé přístupy různých autorů ukazují zatím spíše na tříštění sil než na hledání cesty pro vývoj nějakého univerzálního prostředku. Tato databázová problematika je jedním z témat, která se v současnosti diskutují v ConTEXtové konferenci, a bude i jedním z témat nadcházející konference ConTEXt Meeting.

Reference

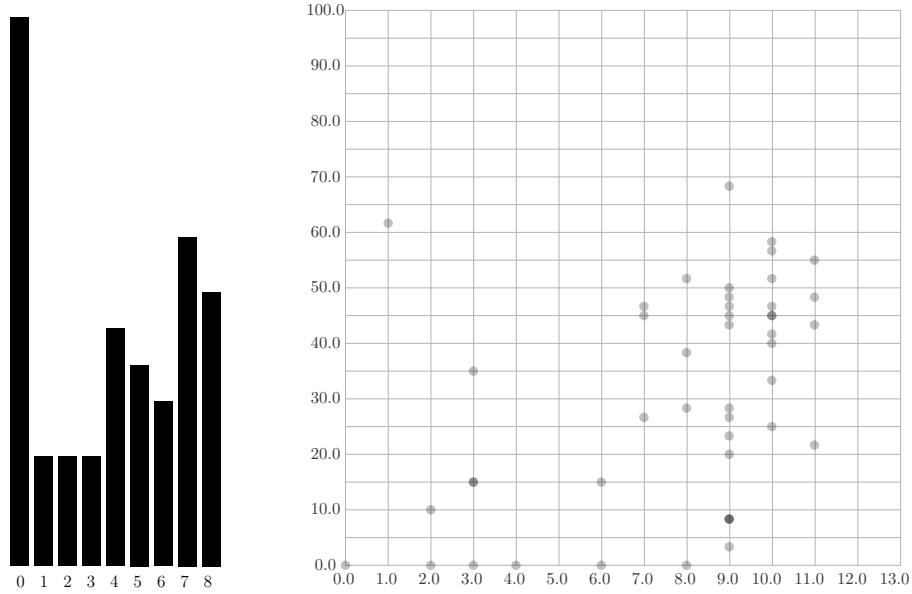
[1] ConTEXt Garden: *M-database*, http://wiki.contextgarden.net/M-database [cit. 1. 6. 2016].

[2] EGGER, W.: *Use of the natural table environment MyWay*, July 16, 2003, 1–9 pp., http://dl.contextgarden.net/myway/NaturalTables.pdf [cit. 21. 6. 2016].

[3] HAGEN, H.: *Extreme Tables: ConTEXt MkIV*, 24 s., http://www.pragma-ade.com/general/manuals/xtables-mkiv.pdf [cit. 21. 6. 2016].

[4] HAGEN, H.: *Simple Spreadsheets: ConTEXt MkIV*, 11 s., https://www.google.sk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=0ahUKEwjhxdyo-uHRAhULuRQKHRYKBxIQFggYMAA&url=http%3A%2F%2Fwww.pragma-ade.nl%2Fgeneral%2Fmanuals%2Fspreadsheets-mkiv.pdf&usg=AFQjCNHOP1vdOyacic-3vttozXQfKBDWuA&bvm=bv.145063293,d.d24&cad=rja [cit. 20. 7. 2016].

[5] HAGEN, H.: *[NTG-context] How can I create a table from a CSV file?*, https://mailman.ntg.nl/pipermail/ntg-context/2016/085789.html [cit. 24. 7. 2016].



Obrázek 32. Ukázka vykreslených grafů.

[6] HAJTMAR, J.: *ScanCSV – Lua knihovna pro zpracování CSV souborů ConTEXtem a LuaLATEXem*, Zpravodaj CSTUG, roč. 22, č. 2, 2012, 76–90 s., ISSN 1211-6661, DOI 10.5300/2012-2/76.

[7] HÁLA, J.: *Radioaktivita, ionizující záření, jaderná energie*, Brno: Konvoj, 1999, 311 s., ISBN 80-85615-56-8.

[8] HÁLA, T.: *LATEX, nebo ConTEXt? První zkušenosti se sazbou ConTEXtem*, Zpravodaj CSTUG, roč. 23, č. 1, 2013, 57–64 s., ISSN 1211-6661, DOI 10.5300/2013-1/57.

[9] HÁLA, T.: *Proč jsem zkusil ConTEXt*, Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach, Zborník príspevkov medzinárodnej konferencie OSSConf 2015, Žilina, Žilinská univerzita v Žilině, 2015, 37–40 s., ISBN 978-80-970457-7-7.

[10] KNUTH, D. E.: *The TEXbook*, Reading (MA, USA): Addison-Wesley, x+483 pp., ISBN 0-201-13448-9.

[11] KORPELA, J.: *Tab Separated Values (TSV): a format for tabular data exchange*, `https://www.cs.tut.fi/~jkorpela/TSV.html` [cit. 6. 5. 2016].

[12] MAHAJAN, A.: *ConTEXt basics for users: Table macros*, TUGboat, Vol. 28, 2007, No. 3, 372–374 pp.

[13] MAHAJAN, A.: *ConTEXt basics for users: Table macros II*, TUGboat, Vol. 29, 2008, No. 1, 219–222 pp.

[14] MIKLAVEC, M.: *Creating tables using CSV (comma–separated values)*, My Way, July 26, 2006, 1–7 pp., `http://dl.contextgarden.net/myway/csv.pdf` [cit. 12. 5. 2016].

[15] OTTEN, T. – HAGEN, H.: *Exkurze do ConTEXtu*, Zpravodaj CSTUG, roč. 16, č. 2–4, 2006, 57–224 s„ ISSN 1211-6661, DOI 10.5300/2006-2-4/1.

[16] RAYMOND, E. S.: *The Art of Unix Programming*, Chapter 5, Textuality: Data File Metaformats, `http://www.catb.org/~esr/writings/taoup/html/ch05s02.html` [cit. 6. 5. 2016].

```
function userdata.tabulka(keywords, keyvals)
   keyword_options = utilities.parsers.settings_to_array(keywords)
   named_values = utilities.parsers.settings_to_hash(keyvals)
   context.pracoviste(string.unquoted(named_values["title"]))
   local roky = { 2010, 2011, 2012, 2013 }
   roku,kolikroku,delim = #roky,3,','
   t = string.split(named_values["data"]," ")
   context("\\bgroup\\setupbodyfont[8dd]\\starttable[s2|p(15cc)|s3w(4cc)c|w(4cc)c|w(4cc)c|]")
   context("\\NC")
   for r=roku-2,roku do
     context("\\NC")
     for i=r,#t-1,roku do  -- 1., ., 11. ... údaj
       if t[i]~="x" then
          context("\\graf{"..t[i-r+1].."}{\\compute{"..t[i].."/
           \\csname "..t[i-r+1].."koef\\endcsname}}%")
       end
     end
   end
   context("\\AR\\NC")
   for i=roku-kolikroku+1,roku do context("\\NC "..roky[i]) end
   context(" \\FR\\HL[2]")
   for i=1,#t-1 do
     s = i % (roku + 1)
     t[i]=string.strip(t[i])
     if s == 1 then
       if utf.len(t[i]) < 6 then context("\\NC\\ctverec{"..t[i].."color}\\ \\csname "..t[i].."\\endcsname")
         else context("\\NC",t[i])
       end
     end
     if s > kolikroku or s == 0 then
       if delim==',' then
        cislo,pocet= string.gsub(t[i],"[.]",",")
        cislo,pocet= string.gsub(cislo,"(%d)(%d%d%d)(%d%d%d)$","%1\\,\\,%2\\,\\,%3")
        cislo,pocet= string.gsub(cislo,"(%d)(%d%d%d)$","%1\\,\\,%2")
       else
        cislo=t[i]
        cislo,pocet= string.gsub(cislo,"(%d)(%d%d%d)(%d%d%d)$","%1,%2,%3")
        cislo,pocet= string.gsub(cislo,"(%d)(%d%d%d)$","%1,%2")
       end
       context("\\NC "..cislo)
     end
     if s == 0 then context("\\FR\\HL") end
   end
   context("[2]\\stoptable\\egroup\\par");
end
```

Obrázek 33. Zdrojový kód funkce `tabulka`.

[17] SHAFRANOVICH, Y.: *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, `https://tools.ietf.org/html/rfc4180` [cit. 6. 5. 2016].

[18] TALANDOVÁ, P.: *Možnosti tabulkové sazby*, Zpravodaj CSTUG, roč. 18, č. 3, 2008, 151–160 s., ISSN 1211-6661, DOI 10.5300/2008-3/151.

Kontaktní adresa

RNDr. Tomáš Hála, Ph.D., Mendelova univerzita, Provozně ekonomická fakulta, ústav informatiky, Zemědělská 1, 613 00 Brno, Česká republika,
E-mailová adresa: `thala@mendelu.cz`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 61–68

ADAPTIVE SPARSE DISTRIBUTED MEMORY AS FUNCTION APPROXIMATOR

MICHAL CHOVANEC (SK) AND PETER ŠARAFÍN (SK)

Abstract. In this paper, we describe modified sparse distributed memory (SDM) for function approximation. Original Kenerva solution was improved to achieve lower error, no sparse data values are required to approximate.

Key words and phrases. Sparse distributed memory, function aproximation, machine learning, neural network.

ADAPTÍVNA DISTRIBUOVANÁ RIEDKA PAMÄŤ AKO APROXIMÁTOR FUNKCIE

Abstrakt. V článku je popísaná modifikácia SDM pamäte slúžiaca ako aproximátor funkcie. Pôvodné Kenervove riešenie bolo vylepšené aby bolo možné dosiahnúť malú chybu aj v prípade, že dáta nemajú riedku povahu.

Kľúčové slová. Riedka distribuovaná pamäť, aproximácie funkcie, neurónová sieť, strojové učenie.

Introduction

Consider sets of N vectors $x(n) \in X$ and $y(n) \in Y$ and function

$$y(n) = h(x(n)),$$

where $x(n) = (x_0(n), \ldots, x_I(n))$, $y_i = (y_0(n), \ldots, y_J(n))$, and I, J and N are more than 10 (usually hundreds or thousands elements). Vectors $x(n)$ and $y(n)$ represent sparse space, and represents to approximate function $h(x(n))$.

This can be found in many applications:

1. Image and voice recognition.
2. Q-values storing in reinforcement learning.
3. Time series prediction.
4. Classification problems.

Neural networks are commonly used for these problems. Conceptions using McCulloch-Pitts neuron model have problematic learning process (local minima, vanishing gradient...), and is almost impossible to learn discontinuous functions.

In 1988 Kenerva published concept of sparse distributed memory [1,2,3]. Original SDM is using binary encoding, and can be rewritten to work with real numbers. The basic idea is shown on figure 1.

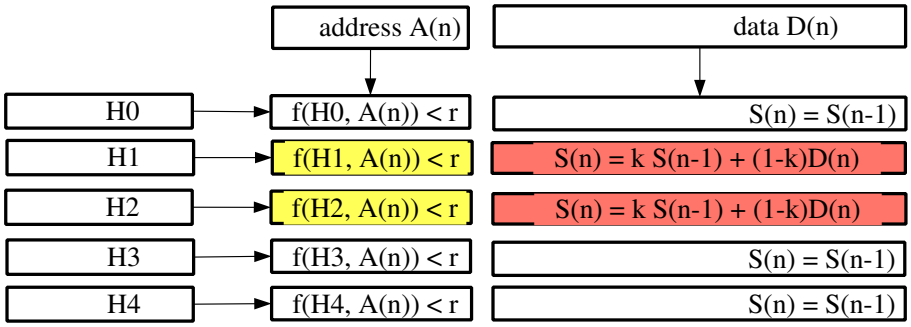


Figure 1. Sparse distributed memory

Inputs, called address $A(n)$ (in function approximation problem $x(n)$), and memory data $D(n)$ (in function approximation problem $y(n)$) are stored into more than one place

$$D(n) = h(A(n)).$$

Saving process is based on storing data in more than one set of places, and they may overlap. There are hard locations places in memory $H_0 \dots H_k$ which are vectors with the same size an address. There is also "distance" function $f(H_i, A(n))$ which measure difference between hard location H_i and address $A(n)$. If it's value is less than radius r data $D(n)$ are stored in corresponding places, using equation

$$S_i(n) = \begin{cases} kS_i(n-1) + (1-k)D(n), & \text{if } f(H_i, A(n)) < r, \\ S_i(n-1), & \text{otherwise,} \end{cases}$$

where $k \in (0, 1)$, and $S_i(n)$ represents stored data.

Data reading process is similar: first the address is presented $a(n)$ and places to read are selecting according $f(H_i, A(n))$. From selected places $S_i(n)$ is calculated arithmetic mean and is presented as output $D'(n)$-approximated value of $D(n)$.

Memory storing ability depends on $A(n)$ and $D(n)$ variance, count of hard locations and used function $f(H_i, A(n))$. For small dimensional spaces (less than 10 (see N-sphere volume)) can be as $f(H_i, A(n)$ used Euclidean distance. For huge dimensional spaces works well cosine angle of vectors H_i, $A(n)$ and examples of tested functions are

$$f_1(H_i, A(n)) = 1 - \left(\frac{H_i \cdot A(n)}{\|H_i(n)\| \cdot \|A(n)\|}\right)^2, \quad f_2(H_i, A(n)) = \mathrm{e}^{-k\left(\frac{H_i \cdot A(n)}{\|H_i(n)\| \cdot \|A(n)\|}\right)^2} \quad (1)$$

values of functions f_1 and f_2 are in range $\langle 0, 1\rangle$, where values close to zero are for similar vectors. Values close to one are for unrelated vectors.

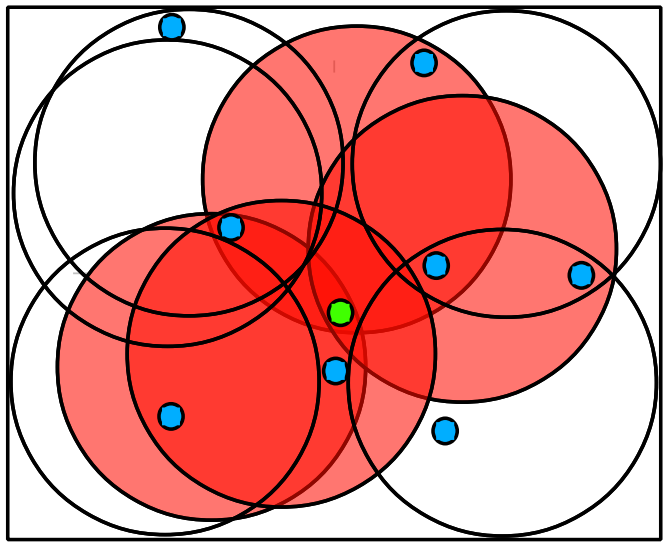


Figure 2. Stored data for huge variance

1. Adaptive sparse distributed memory

First consider that $A(n)$ and $D(n)$ have huge variance — f. e. uniform distribution. Situation is demonstrated in the figure 2. Circles represent radius r and their center's represent hard locations positions. Each hard location has stored corresponding data. Blue dots represent data stored previously, green dot represent some new item to be stored. Red circles represent hard locations where new data is stored.

In situation, when $A(n)$ has low variance and $Y(n)$ has huge variance (Figure 3), addresses $A(n)$ in meaning of equation (1) are very close. In this case, radically different data $D(n)$ (because of its uniform distribution) is stored in same locations, because of low variance of $A(n)$. This yields to huge storage error, and memory can't be used in this situation.

To avoid this situation, we need to define quality of stored data, as the storage error

$$e_i(n) = \alpha e_i(n-1) + (1-\alpha)\,\|d(n) - s_i(n)\|,$$

where $\alpha \in (0, 1)$ is constant, usually close to 1 and $e_i(n)$ is smoothed difference between required data to be stored $d(n)$ and actually stored data into memory $s(n)$ — equation represents complementary filter.

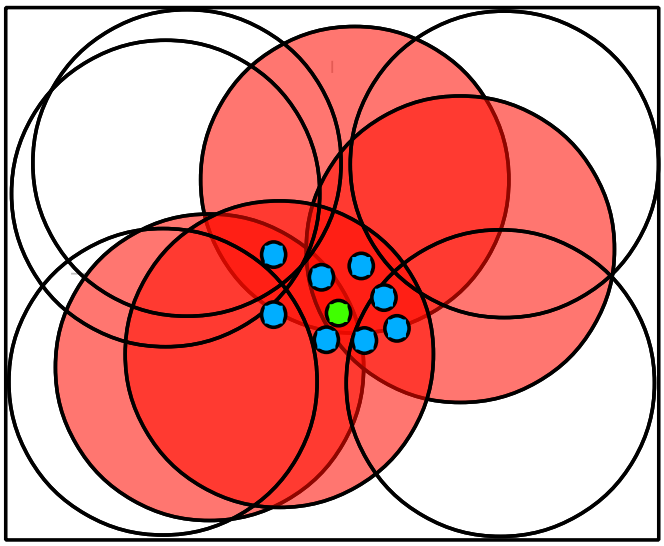


Figure 3. Stored data for low variance

After each storing process, the error is calculated and when reach defined treshold e_t (constant during operation), few new hard locations are added using following equation

$$H_{new}(n) = \eta x(n) + (1-\eta)r(n), \quad s_{new}(n) = ks_{new}(n-1) + (1-k)d(n),$$

where $r(n)$ is random noise vector (in experiments with uniform distribution) and $\eta \in (0, 1)$ is constant, usually close to 1.

2. Experimental results

Three basic experiments have been processed to test this modification, compared to the original solution:

1. Function approximation.
2. Time series prediction.
3. Mnist database image recognition (accuracy 93%–96%).

In this paper, we present function approximation results. Input vector $x(n)$ size was 50 and output vector size $y(n)$ was set to 10. Thousand input vectors have been randomly generated with Gaussian distribution in few groups — to test similar inputs separation. Output vectors have been generated with uniform random distribution. We chose 5000 hard locations count.

Figure 4 shows 3D cut of 50 dimensional space of positions of input vectors as green dots, and hard locations as purple. Hard locations are completly uniformly distributed, but input data is in few groups.

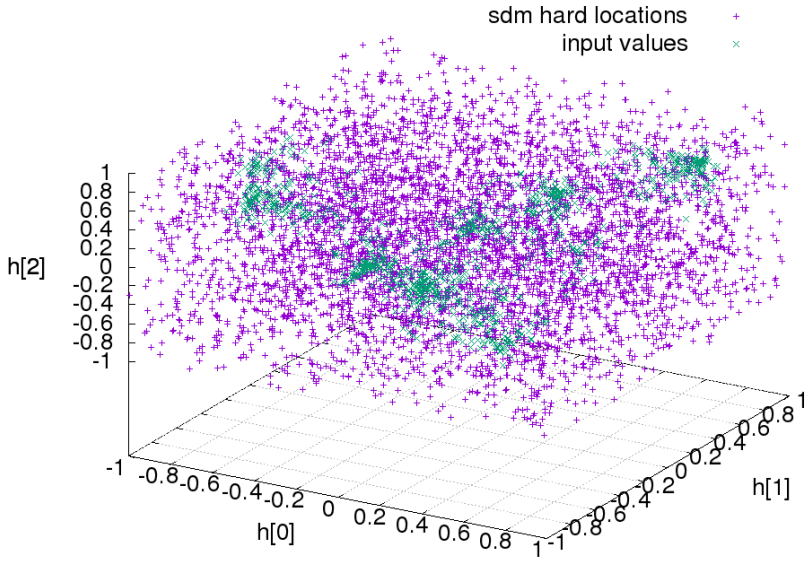


Figure 4. Hard locations positions for original solution

Figure 5 shows situation for improved memory. Hard locations have higher density in positions indeticaly with input vectors. This solution better cover input space.

Input data has been generated using following equations

$$t(n) = \begin{cases} \text{uniform}(-1,1), & \text{if uniform}(0,1) \geq \text{similarity}, \\ t(n-1) & \text{otherwise}, \end{cases}$$

$$d(n) = (1-k)t(n) + k\,\text{uniform}(-1,1).$$

For similarity equal to 1 the data is generated in one group, for similarity close to 0 the data is generated with uniform distribution. Parameter $k \in (0,1)$ is responsible for data dispersion – if close to 1 data have huge dispersion around point $t(n)$. If close to 0 data have small dispersion around $t(n)$. In fact, this parameter effects entropy around $t(n)$.

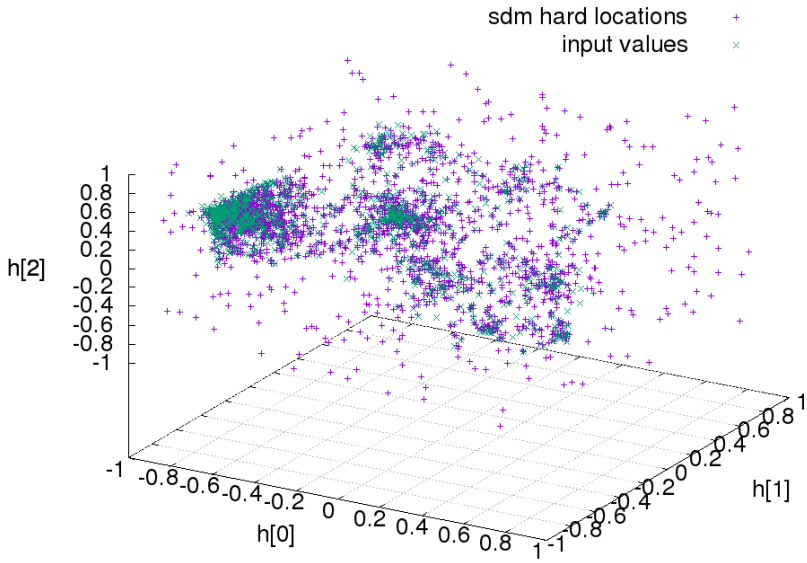


Figure 5. Hard locations positions for improved solution

For comparison error between required and stored data has been estimated as

$$E = \sum_{n=1}^{N} \frac{(\|d(n)-s(n)\|)^2}{NJ},$$

where N is vector count (1000), and J is output vector dimension (10).

Results are in the figure 6. On the X axis there is similarity parameter. This shows how input data character affect the resulting error. Original solution has better results when data is in uniforom-like distribution (small similarity). For high similarity parameter, group error increses. On the other side, similarity doesn't radicaly change improved solution results – error is close to 0.

In figure 7 we can see memory locations utilization – used hard locations to store data. For values around 0.3 it is 30% (1500 locations). In original solution for similar data exists only few locations to store them. In improved solution, adtional locations are created, and memory utilization is better.

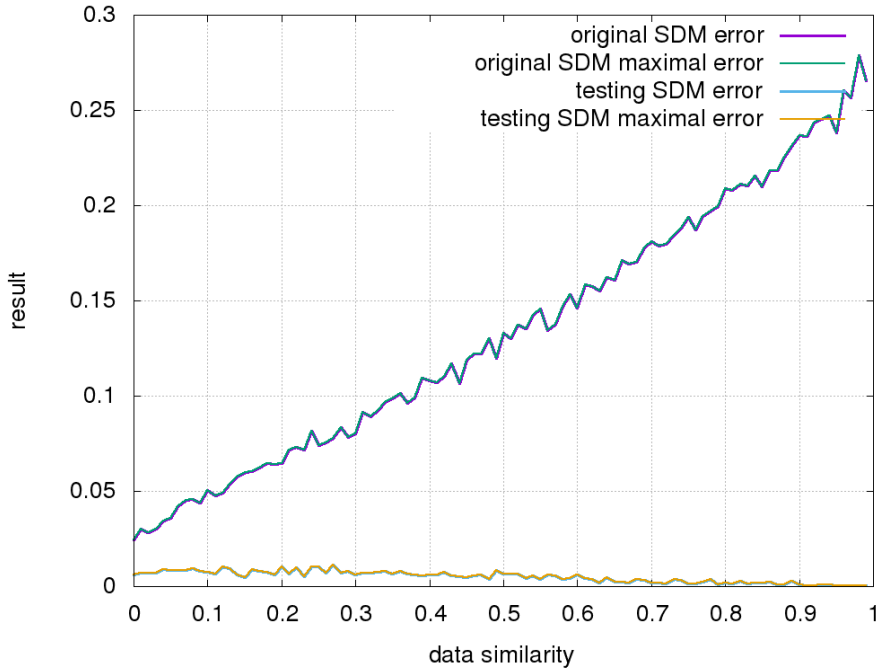


Figure 6. Approximation error result

3. Conclusion

In this paper, we presented improved sparse distributed memory. Memory was tested on function approximation problem, with different input vector's distribution. On dataset with low variance, results were significantly better than on original solution. Further experiments are also required, to test approximation of differentiable functions, and dynamical systems prediction. Further application can be found in [4]. Sources of memory are available at [5]

References

[1] ROGERS, D.: *Kanerva's sparse distributed memory: An associative memory algorithm well-suited to the connection machine*, 1988, `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19890017031.pdf`.

[2] FLYNN, M. J. – KANERVA, P. – BHADKAMKAR, N.: `http://infolab.stanford.edu/pub/cstr/reports/csl/tr/89/400/CSL-TR-89-400.pdf`.

[3] GREBENÍČEK, F.: *Self-Organized Sparse Distributed Memory – an Application*, `http://www.fit.vutbr.cz/~grebenic/Publikace/Asis99/`.

[4] MENDES, M. – CRISOSTOMO, M. – COIMBRA, A. P.: *Robot navigation using a Sparse Distributed Memory*, Conference: 2008 IEEE International Conference

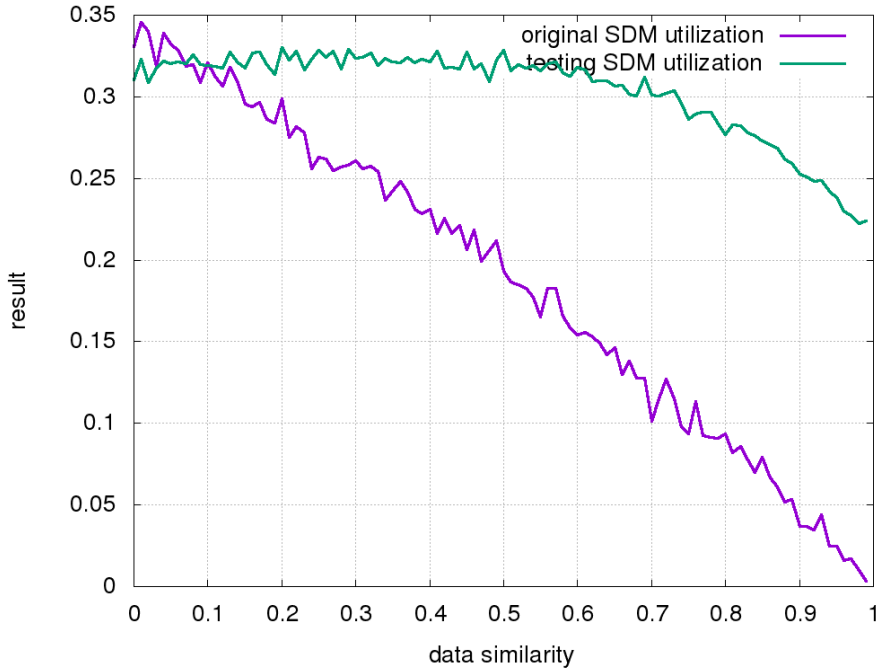


Figure 7. Memory utilization

on Robotics and Automation, ICRA 2008, May 19–23, 2008, Pasadena, California, USA, https://www.researchgate.net/publication/221069833_Robot_navigation_using_a_Sparse_Distributed_Memory, DOI: 10.1109/ROBOT.2008.4543186.

[5] CHOVANEC, M.: *Sources of apdative sparse distributed memory*, https://www.researchgate.net/publication/221069833_Robot_navigation_using_a_Sparse_Distributed_Memory.

Contact addresses

Ing. Michal Chovanec, Department of Technical Cybernetics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: michal.chovanec@yandex.com

Ing. Peter Šarafín, Department of Technical Cybernetics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: peter.sarafin@fri.uniza.sk

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf
OSSConf 2016: 69–76

AKO SOM OBJAVIL KNIŽNICU `tikzDevice`

ALEŠ KOZUBÍK (SK)

Abstrakt. Grafické prvky sú dôležitou súčasťou tlačených dokumentov. Jedným z významných atribútov grafických súborov, vkladaných do dokumentov je ich zosúladenie s formátovaním ostatného textu. To nemusí byť vždy jednoduché, ako ukazuje aj spolupráca medzi prostredím R a typografickým systémom LaTeX. Tento príspevok prináša informáciu o balíčku `tikzDevice`, ktorý umožňuje používať syntax LaTeX-u v rámci grafických výstupov v prostredí R.

Kľúčové slová. R, Ti*k*Z, tikzDevice.

HOW DID I DISCOVER THE `TIKZDEVICE` LIBRARY

Abstract. The graphical objects represent an important part of the printed documents. A significant attribute of the inserted graphics files is the unification of their formatting with the surrounding text. It need not be always easy, how we can see on the cooperation between the R environment and the typographic system LaTeX. The present paper brings some information about the `tikzDevice` package that enables using the LaTeX syntax in the graphical outputs of the R environment.

Keywords. R, Ti*k*Z, tikzDevice.

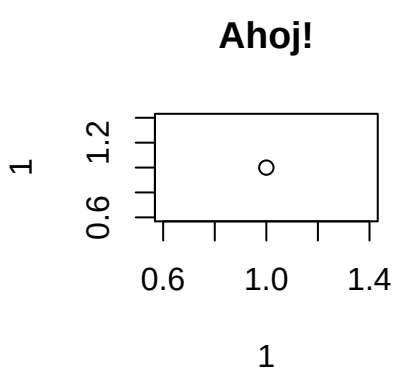
Úvod

Poznáme to všetci. Pri úvodnom zoznámení sa s novým nástrojom sa tešíme z každého dosiahnutého výsledku. To platí aj o grafických výstupoch, ktoré nás obdarúvajú radostnými pocitmi zo získania požadovaného výsledku. Ešte väčšiu radosť máme, keď sa nám vytúžený obrázok podarí vložiť do iného, napr. TeX-ového dokumentu. Až v „druhom kole“ si začíname uvedomovať, že náš obrázok nie je celkom bez chýb, začína nám prekážať, že fonty použité v obrázku sa nie celkom zhodujú s tými, ktoré sú použité v dokumente atď. Navyše, je tu ešte kolo tretie, kedy nás už sústavné opakovanie tých istých postupov začína unavovať a hľadáme možnosti celý proces nejakým spôsobom zautomatizovať.

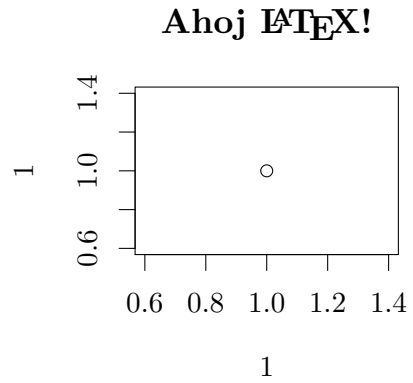
Neinak tomu bolo aj v mojom prípade, kedy som sa pokúšal preniesť grafické výstupy z prostredia R do dokumentov sádzaných v sytéme LaTeX. Po vytriezvení z prvých radostných pocitov z vygenerovania grafov vybraných rozdelení pravdepodobností mi začalo prekážať, že popisy grafu, legenda a pod. sú odlišné od okolitého textu. Ako prvé riešenie som „vygúglil“ knižnicu `latex2exp`. Po niekoľkohodinovom márnom boji s touto knižnicou, bez významnejšieho priblíženia sa ku vytúženému výsledku som dospel k záveru, že možno rýchlejšie nájdem iné

riešenie, než dobojujem tento, zdanlivo vopred prehratý, súboj. Môžem potvrdiť, že intuícia ma v tomto prípade nesklamala a pomerne rýchlo som vypátral knižnicu `tikzDevice`, ktorá plne uspokojila moje požiadavky na prepojenie grafiky prostredia R so systémom LaTeX resp. pdfLaTeX.

V nasledujúcich odstavcoch teda stručne predstavím základy činnosti knižnice `tikzDevice` a jej spoluprácu so systémom pdfLaTeX. V tomto príspevku sa zameriavam na spoluprácu so sytémom LaTeX, ale knižnica spolupracuje aj s XeLaTeX-om a LuaLaTeX-om, nemám s nimi však praktické skúsenosti. Ukážku toho, ako môže vyzerať výstup bez použitia a s použitím tejto knižnice si ilustrujeme na obrázkoch 1 a 2. Na prvý pohľad je zrejmé, že výstup na obrázku 2 korešponduje s okolitým textom lepšie, než obrázok 1.



Obr. 1. Ukážka „neželaného“ výstupu grafiky z prostredia R



Obr. 2. Ukážka žiadúceho výstupu grafiky z prostredia R

1. Získanie a použitie knižnice `tikzDevice`

1.1. Inštalácia

Balíček `tikzDevice` predstavuje knižnicu prostredia R, ktorá umožňuje ukladať grafické výstupy z prostredia R vo formáte, ktorý je vhodný pre použitie v typografických systémoch, založených na TeX-u. Pri ukladaní dochádza ku konverzii kresliacich príkazov jazyka R do blokov LaTeX-ového kódu, ktorý je interpretovaný pomocou balíčka `TikZ`, ktorý sme si predstavili pred niekoľkými rokmi (pozri napr. [3] alebo manuál [6]).

Stabilná verzia tejto knižnice je súčasťou repozitára CRAN. Je ju teda možné inštalovať štandardným spôsobom, pomocou príkazu:

```
install.packages('tikzDevice')
```

Pripomeňme si, že systém je case sensitive a veľké „D“ v názve balíčka je teda podstatné. Pre tých, ktorí túžia po experimentovaní s vývojovou verziou balíčka odporúčam inštalovať aj s uvedením konkrétneho repozitára na RForge, teda:

```
install.packages('tikzDevice',repos='http://rforge.net',type='source')
```

1.2. Načítanie balíčka

Funkcie balíčka `tikzDevice` sa v prostredí R sprístupnia štandardným načítaním pomocou:

```
library(tikzDevice)
```

Po načítaní balíčeк vyhľadáva niektorý z LaTeX-ových kompilátorov:

- pdfLaTeX,
- XeLaTeX,
- LuaLaTeX.

V prípade, že sa nenájde funkčný pdfLaTeX kompilátor, dôjde ku zlyhaniu balíčka `tikzDevice` a zobrazí sa chybové hlásenie.

Správanie balíčka je ovplyvňované viacerými voliteľnými argumentmi, ktoré je možné buď lokálne načítať v rámci jednotlivých R-skriptov alebo na R konzole. Inou možnosťou je ich globálne nastavenie v súbore `.Rprofile`. Tieto voľby sa nastavujú pomocou príkazu:

```
options(<voľba>=<hodnota>)
```

Pre zjednodušenie návratu ku pôvodným default nastaveniam je k dispozícii funkcia `setTikzDefaults()`.

Postupne si uvedieme aspoň niektoré z možných voliteľných nastavení. V prvom rade je to výber implicitného LaTeX kompilátora. Ku tomuto účelu slúži voľba `tikzDefaultEngine`, ktorá môže nadobúdať jednu z troch hodnôt `pdftex`, `xetex` alebo `luatex`. Príslušné príkazy potom vyzerajú napr. takto:

```
options(tikzDefaultEngine='pdftex')
```

pre implicitný prekladač pdfLaTeX, resp.

```
options(tikzDefaultEngine='xetex')
options(tikzDefaultEngine='luatex')
```

pre ostatné dve alternatívy.

1.3. Prvý obrázok

Po načítaní knižnice `tikzDevice` sme už pripravení vygenerovať kód, pre získanie obrázku 2. Na konzole prostredia R stačí zadať nasledujúce príkazy:

```
tikz('tikz-example.tex',
width = 3.25, height = 3.25)
plot(1, 1, main = 'Ahoj \\TeX !')
dev.off()
```

a na požadované miesto v zdrojovom súbore vložiť načítanie výstupného súboru `tikz-example.tex`, teda:

```
\input{tikz-example}
```

Na uvedenom príklade si všimnime, že pre zadávanie znaku opačného lomítka \ je potrebné použiť tento symbol dvakrát, teda \\, inak dôjde pri generovaní LaTeX-ového kódu balíčkom `tikzDevice` ku chybe.

2. Funkcia `tikz`

Rozhodujúcim prvkom činnosti celého balíčka `tikzDevice` je funkcia `tikz`. Jej činnosťou je vlastne otvorenie grafického zariadenia prostredia R, ktoré zaznamenáva celý grafický výstup pomocou príkazov Ti*k*Z-u. Toto grafické zariadenie podporuje viacero možností výstupu, počínajúc `standalone` dokumentmi LaTeX-u a končiac pri súbore príkazov Ti*k*Z-u, ktorý je potrebné vložiť do zdrojového kódu pomocou príkazov `\input{}` resp. `\include{}`.

2.1. Argumenty funkcie `tikz`

Funkciu `tikz` je možné použiť s viacerými argumentmi. Uveďme aspoň tie najvýznamnejšie:[1]

`file`: Určenie názvu výstupného súboru. Argumentom je reťazec znakov, udávajúci celú cestu ku výstupnému súboru. Odporúča sa, aby súbor mal príponu `.tex`, nie je to však nevyhnutné.

`width`: Nastavenie šírky obrázka, údaj je v palcoch.

`height`: Nastavenie výšky obrázka, údaj je v palcoch.

`bg`: Špecifikuje farbu pozadia obrázku.

`fg`: Špecifikuje začiatočnú farbu vykreslenia grafu.

`standAlone`: logická hodnota, ktorá špecifikuje, či má byť výstupný súbor samostatne kompilovateľným zdrojovým textom pre LaTeX.

`bareBones`: Logická hodnota, ktorá udáva, či majú byť príkazy Ti*k*Z-u umiestnené do samostatného prostredia `tikzpicture`.

Najvýznamnejšou vlastnosťou funkcie `tikz` je, že nám dáva možnosť umiestniť akúkoľvek sekvenciu príkazov LaTeXu do všetkých kresliacich nástrojov funkcií prostredia R. Treba však mať na pamäti, že tak ako je tomu u viacerých programovacích jazykov, aj tu má spätné lomítko \ špeciálny význam. Preto pre zápis opačného lomítka je potrebné použiť zdvojený symbol \\.

Ukážme si to na príklade zložitejšieho výstupu grafu z prostredia R. Konkrétne si ilustrujeme výpočet a grafickú prezentáciu lineárnej regresie v prostredí R. Nakoľko nie je naším cieľom výklad jazyka R, sústredíme sa v zdrojovom kóde iba na výklad riadkov súvisiacich s balíčkom `tikzDevice`.

[1] Pre úplný zoznam treba pozri manuál [5].

Základom je samozrejme načítanie knižnice na prvom riadku výpisu. Ďalším, z nášho pohľadu významným riadkom je riadok č. 4, kde sa spúšťa grafické zariadenie prostredia R pre zápis Ti*k*Z-ového kódu. Potom už nasledujú štandardné príkazy jazyka R, s tým, že všetky texty sa konvertujú do LaTeX-u, symbol $ ako obyčajne začína a ukončuje matematický režim sadzby a všetky kontrolné slová LaTeX-u je potrebné začínať zdvojeným symbolom spätného lomítka, ako sme si už spomenuli. Výsledok si môžeme pozrieť na obrázku 3. Tu pozorný čitateľ postrehne istú nedokonalosť v znamienku absolútneho člena vo výsledku lineárnej regresie. To by bolo možné odstrániť pomocou vetvenia `if` v jazyku R, to však nie je účelom tohto príspevku.

```
1  library(tikzDevice)
2  x<-rnorm(20)
3  y<-x+rnorm(5,sd=1)
4  tikz('regresia.tex',width=4,height=4)
5  plot(x, y,las=1, main='Nadpis \\LaTeX!')
6  model<- lm(y ~ x)
7  rsq<-summary(model)$r.squared
8  rsq<-signif(rsq,4)
9  abline(model,col='red')
10 mtext(paste("Linear model:$R^{2}=",rsq, "$"), line
       =0.5)
11 legend('topleft',legend =paste("$y=",round(coef(
       model)[2],3), 'x +',
12 ...round(coef(model)[1],3), '$ ', sep = ''), bty =
       'n')
13 dev.off()
```

3. Veľká vec – diakritika

V záverečnej sekcii sa dostávam ku obrázkom, ktorými to vlastne všetko začalo. Pre potreby teórie pravdepodobnosti som potreboval vygenerovať obrázky konkrétnych rozdelení a získať popisy zhodné s okolitým textom, teda nie napríklad Hustota f(x), ale Hustota $f(x)$. Samozrejme, v nadpise sa v pomenovaní rozdelenia začína objavovať aj text s diakritickými znamienkami, ktorý balíček `tikZdevice` nedokáže spracovať „sám od seba“. Naopak, pomocou voliteľných `options` pre vloženie LaTeX-ových balíčkov týkajúcich sa kódovania, fontov a metrík fontov

je potrebné prostredie R pripraviť na korektnú transformáciu výstupu v kódovaní UTF-8.

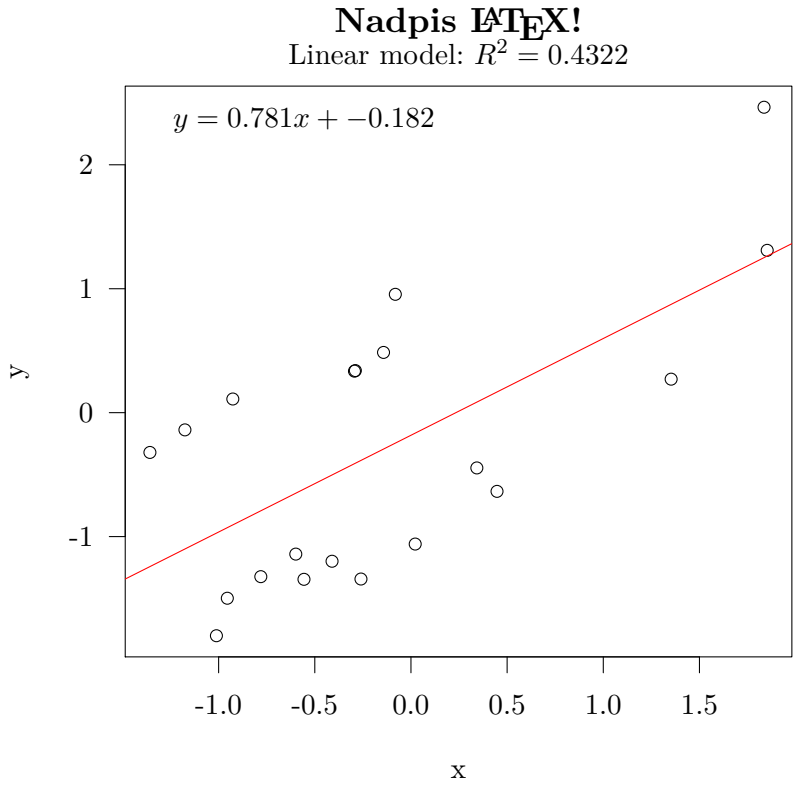
Na začiatok R-kového skriptu som vložil nasledujúce riadky:

```
library('tikzDevice')
options(tikzDefaultEngine = "pdftex")

options(tikzLatexPackages = c(
    "\\usepackage{amsmath,amssymb,amsfonts}\n",
    "\\usepackage{tikz}\n",
    "\\usepackage[utf8]{inputenc}\n",
    "\\usepackage[T1]{fontenc}\n",
    "\\usepackage{lmodern}\n",
    "\\usetikzlibrary{calc}\n",
    "\\usepackage{standalone}"
))
options(tikzMetricsDictionary="~/R/tikzMetrics")

options(tikzMetricPackages = c(
    "\\usepackage[utf8]{inputenc}\n",
    "\\usepackage[T1]{fontenc}\n",
    "\\usetikzlibrary{calc}"
))
```

za ktorými už nasledovali príkazy jazyka R pre vykreslenie funkcií hustoty pravdepodobnosti exponenciálneho rozdelenia pre rôzne hodnoty parametra λ. Výstup sa ani tentoraz neodohral bez problémov a pri použití som dostal nasledujúce varovné hlásenia:

```
Warning message:
In (function(texString,cex = 1,face = 1,
                          engine = getOption("tikzDefaultEngine"),:
Attempting to calculate the width of
                          a Unicode stringusing the pdftex engine.
This may fail!
See the Unicodesection of ?tikzDevice for more information.
```

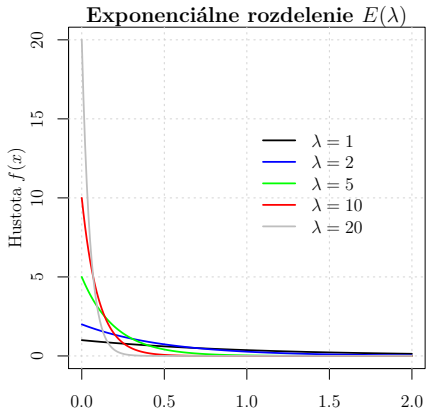
Avšak kompilácia výstupu prebehla bez ťažkostí a získal som výstup v podobe obrázku 4. Podobným postupom bol vytvorený aj graf zodpovedajúcich distribučných funkcií na obrázku 5.



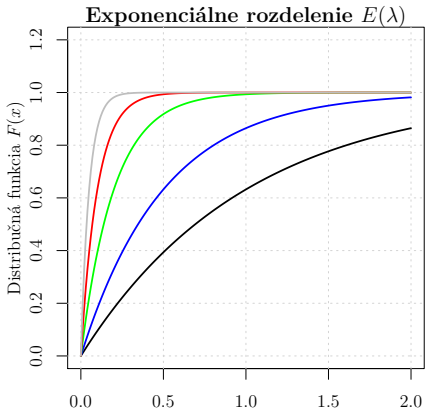
Obr. 3. Grafická prezentácia lineárnej regresie v prostredí R

4. Záver

Balíček `tikzDevice` je určite užitočným rozšírením možností vývojového prostredia R. Umožňuje účinnú spoluprácu tohto štatistického nástroja s typografickým systémom LaTeX a jeho grafickou nadstavbou v podobe balíčka `tikz`. Tak sa stáva vítanou pomôckou pre vytváranie kvalitných výstupov pre prezentáciu výsledkov získaných pri použití tohto prostredia. V manuáli [5] je možné nájsť aj detailný výklad pre spoluprácu s ostatnými alternatívami TeX-u a to menovite XeLaTeX-u a LuaTeX-u. Autor však nemá praktické skúsenosti s použitím týchto nástrojov, preto im v článku nevenuje pozornosť. O možnej spolupráci so systémom ConTeXt sa autorovi nepodarilo zistiť vôbec nič.



Obr. 4. Grafy hustoty exponenciálneho rozdelenia, získané z prostredia R



Obr. 5. Grafy distribučnej funkcie exponenciálneho rozdelenia, získané z prostredia R

Poďakovanie. Tento príspevok vznikol s láskavým prispením grantu KEGA–011ŽU–4/2014 „Experimentálna matematika – zviditeľnenie neviditeľného“ podporeného Slovenskou kultúrno-edukačnou grantovou agentúrou.

Literatúra

[1] BLAŠKO, R.: *Nebojme sa obrázkov v LaTeX-u*, Zborník príspevkov medzinárodnej konferencie OSSConf 2011, 2.–4. júla 2012, Žilina, str. 79–85. ISBN 978-80-970457-2-2.

[2] CRAWLEY, M. J.: *The R book*, Chichester, John Wiley & Sons Ltd. 2007, ISBN 978-0-470-51024-7.

[3] KOZUBÍK, A.: *Naučím vás kresliť alebo predstavenie balíčka TikZ*, Zborník príspevkov medzinárodnej konferencie OSSConf 2011, 2.–4. júla 2012, Žilina, str. 91–96. ISBN 978-80-970457-2-2.

[4] RYBIČKA, J.: *LaTeX pro začátečníky*, Brno, KONVOJ 2003, ISBN 80-7302-049-1.

[5] SHARPSTEEN, Ch. – BRACKEN, C.: *LaTeX Graphics for R*, Public domain dokument, 2016, `https://cran.r-project.org/web/packages/tikzDevice/tikzDevice.pdf`.

[6] TANTAU, T.: *The TikZ and PGF Packages Manual for version 3.0.1a*, `http://mirrors.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`.

Kontaktná adresa

RNDr. Aleš Kozubík, PhD., Katedra matematických metód a operačnej analýzy, Fakulta riadenia a informatiky, Žilinská univerzita, Univerzitná 8215/1, 010 26 Žilina, Slovensko,
E-mailová adresa: `alesko@frcatel.fri.uniza.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf
OSSConf 2016: 77–84

INVESTIGATION OF ERRORS IN VIRTUAL MODEL OF HUMAN BODY USING BLENDER

MIROSLAV KVAŠŠAY (SK)

Abstract. One of the principal issues in development of computer-based tools for training of complex medical procedures, e.g. regional anaesthesia, is availability of high accurate virtual anatomical models of a human body. There exist several high-detailed models that could be used for this task, but the question is whether they have the required quality. In this paper, a new approach for quantification of errors in such models is presented. The approach is based on the assumption that the human body is modelled as a triangle mesh, and the basic idea behind it is quantification of the overlapping elements of the model. The presented approach was implemented in open source 3D modelling tool Blender and was used to investigate a real anatomical model of a human body.

Key words and phrases. Virtual human body, 3D modelling, Blender.

ANALÝZA MODELU ĽUDSKÉHO TELA S VYUŽITÍM NÁSTROJA BLENDER

Abstrakt. Jedným z hlavných problémov vo vývoji nástrojov pre nácvik komplexných medicínskych úkonov (napr. úkony súvisiace s lokálnou anestéziou) s využitím virtuálnej reality je dostupnosť kvalitných anatomických modelov ľudského tela. V súčasnosti existuje niekoľko veľmi detailných modelov, ktoré by mohli byť použité k tomuto účelu, avšak otázkou zostáva, či je ich kvalita dostatočná. V tomto článku je popísaný nový prístup pre kvantifikáciu chýb v takýchto modeloch. Prezentovaný prístup vychádza z predpokladu, že ľudské telo je modelované ako trojuholníková sieť, pričom základná myšlienka spočíva v identifikácií a kvantifikácií pretínajúcich sa elementov skúmaného modelu. Prístup popísaný v tomto článku bol implementovaný v open source nástroji Blender a bol použitý pre analýzu jedného z existujúcich modelov.

Kľúčové slová. Virtuálny model ľudského tela, 3D modelovanie, Blender.

Introduction

One of the current trends in modern medicine is development of high accurate computer-based tools for training of complex medical procedures. One of the projects that deal with this task is RASimAs.

RASimAs (Regional Anaesthesia Simulator and Assistant) [1, 2] is a research project founded by the European Union's 7th Framework Program, whose goal is to increase the application, effectiveness and success rates of regional anaesthesia

procedures. It aims at developing two independent but complementary computer-based systems: RASim (Regional Anaesthesia Simulator) and RAAs (Regional Anaesthesia Assistant).

Figure 1. Prototype of RAAs (Source: [2])

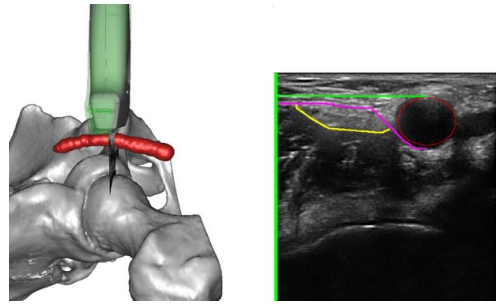


Figure 2. Recognition of artery (red), fascia (purple), and nerve (yellow) using RAAs (Source: [2])

The RAAs (Fig. 1) will assist the physicians to localize the nerve during the procedure. The RAAs represents a complex system whose principal parts are perfect ultrasound scanner, powerful computer, and software for real-time detection of the key anatomical structures (artery, fascia, nerve, etc.) in the ultrasound frames (Fig. 2).

Figure 3. Prototype of RASim

The RASim (Fig. 3) is an augmented reality-oriented system that will provide training in performing regional anaesthesia. To achieve this goal, the RASim has to demonstrate the key features of the regional anaesthesia and, therefore, it includes patient and ultrasound views, virtual ultrasound nerve blocking, and haptic guided needle insertion procedure. Development and use of such tool require high accurate anatomical model of human body. One of the possible models that have been considered for this task is Zygote [3]. However, during the development of the RASim prototype, it has been found that the purchased

version of Zygote has some shortcomings, e.g., missing structures or overlapping layers (Fig. 4) that penalize its use. So, the question is how reliable is this model. To answer this question, we tried to quantify how many overlapping structures exist in Zygote. For this task, we used open source 3D modelling tool Blender version 2.76 [4].

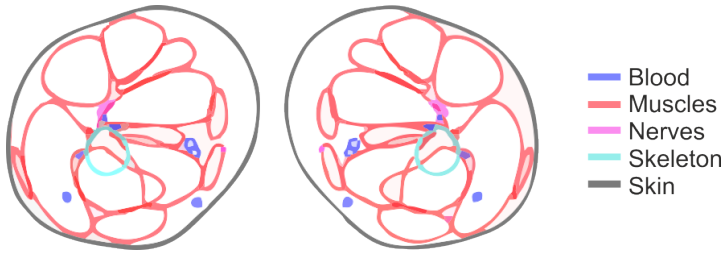


Figure 4. Overlapping layers in Zygote

1. Virtual Model of Human Body

Zygote represents a virtual model of human body. It consists of several layers that represent individual systems of human body. In this paper, we will consider only the model of a male, and we will focus on 6 layers: skeletal, muscular, connective (cartilages around joints), lymphatic, circulatory, and nervous (Fig. 5 and 6).

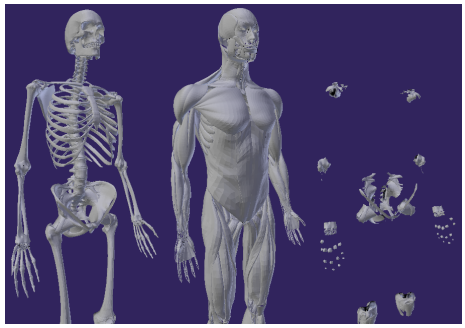


Figure 5. Zygote – skeletal, muscular, and connective layers

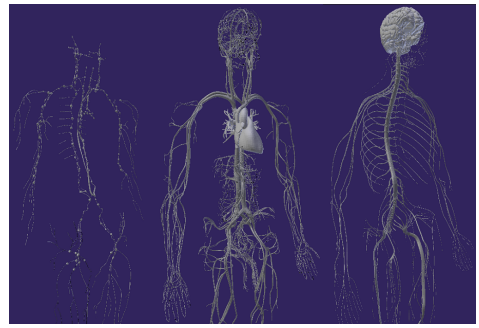


Figure 6. Zygote – lymphatic, circulatory, and nervous layers

Every layer is described by a triangle mesh that consists of vertices, edges, and triangles. This mesh defines surface of the objects that the layer is composed of (Fig. 7). For example, the muscular layer contains 296 735 vertices, 886 997 edges, and 591 342 triangles. The similar data about other layers are presented in Table 1. According to this table, the most complex layers are circulatory, nervous, and muscular.

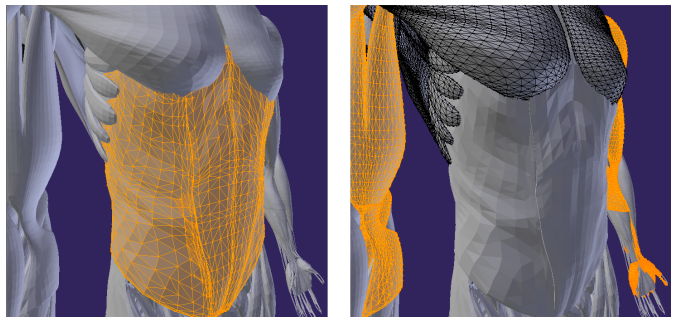


Figure 7. Examples of triangle meshes modelling muscular layer

In Fig. 7, we can see that the triangle mesh representing a layer is not connected, i.e. it is composed of a lot of components. Every component is a connected triangle mesh that represents the surface of one element of the layer. Such element can agree with a bone (skeletal layer), muscle (muscular layer), nerve (nervous layer), etc. This implies that before identifying the overlapping elements, the triangle meshes representing individual layers of the model have to be split into components. For this task, Blender and its tool known as 'Separate' (shortcut 'P' in 'Edit Mode') can be used. After applying this tool, for example, 563 objects were identified in the muscular layer. The numbers for other layers are presented in Table 2.

Layer	**Vertices**	**Edges**	**Triangles**
Skeletal	196 787	589 231	392 822
Muscular	296 735	886 997	591 342
Connective	158 672	473 301	315 424
Lymphatic	53 245	157 441	104 952
Circulatory	401 468	1 201 191	800 747
Nervous	367 849	1 101 881	734 501

Table 1. Basic statistics about Zygote

2. Intersection Analysis

After identifying individual objects of the layers, the overlapping analysis can be performed. One of the possible ways of how this can be done is finding the intersection between each pair of objects. If a model is correct, there should be no or minimal intersections between individual bones, muscles, bones and muscles, etc. On the other hand, if the intersections are large, then it implies that the model is probably not very accurate.

Layer	Objects	Non-manifolds		Manifolds with boundaries	
Skeletal	246	1	(0.41%)	0	(0.00%)
Muscular	563	12	(2.13%)	505	(89.70%)
Connective	415	2	(0.48%)	202	(48.67%)
Lymphatic	381	0	(0.00%)	82	(21.52%)
Circulatory	688	21	(3.05%)	182	(26.45%)
Nervous	275	12	(4.36%)	27	(9.82%)

Table 2. Basic analysis of individual layers of Zygote

Computation of the intersection between two 3D objects modelled by triangle meshes has one principal issue – detection whether a point is inside an object or outside. Such decision can be done if and only if the triangle mesh of the object is a manifold without boundary.

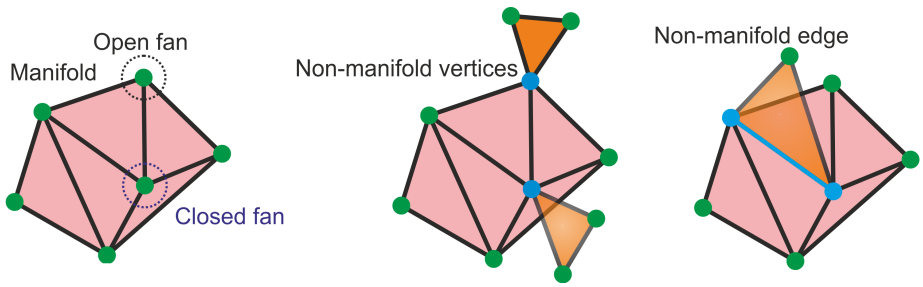


Figure 8. Examples of triangle meshes - manifold, a mesh with non-manifold vertices, a mesh with non-manifold edge and vertices

According to [5], a triangle mesh is manifold if a) each edge is incident to only one or two triangles, and b) the triangles incident to a vertex form a closed or an open fan (Fig. 8). A manifold has no boundary if it contains only vertices that create closed fans. This implies that the first task in the intersection analysis is identification of objects that are not manifolds. For this task, another tool in Blender can be used. This tool is referred to as 'Select Non Manifold' (shortcut 'Shift+Ctrl+Alt+M' in 'Edit Mode') and it allows finding non-manifold vertices and edges and boundary edges in the selected mesh. Using this tool, we analysed individual objects of the layers, and the results are also shown in Table 2. As we can see, most of the objects are correct from the point of view of manifold/non-manifold. However, a problem arises in case of manifolds with boundaries, i.e. almost all muscles and a lot of connective, circulatory, and lymphatic objects are modelled as manifolds with boundaries (Fig. 9). This implies that the intersection analysis cannot be performed directly on these objects. To solve this problem, we tried to close the objects that are modelled as manifolds with boundaries. For this

task, another tool of Blender was used. This tool is referred to as 'Fill' (shortcut 'Alt+F'), and it fills a selected edge loop with triangles. So, for every manifold with boundaries, we selected boundary edges that created a loop and using this tool we filled it with triangles (Fig. 10). This ensured that all analysed meshes have no boundaries. After this, the intersection analysis could be performed.

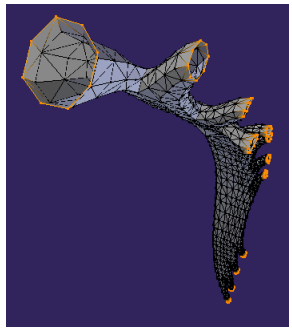


Figure 9. A muscle modelled as a manifold with boundary

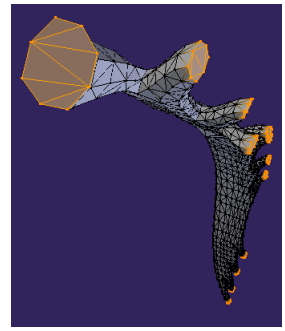


Figure 10. The muscle after the filling procedure

The intersection analysis can be done in several ways. If we have two objects and if we want to quantify their intersection, then we can compute volume of the intersection and compare it with the volumes of the investigated objects or identify how many vertices/edges/triangles of one object are inside another. In this paper, we focus on identifying number of vertices. For this task, we need to decide if a given vertex is inside a triangle mesh (which has the properties of a manifold without boundaries) or not. We solved this task by adding a function proposed in [6] into Blender. This function has two parameters from which the first one agrees with the investigated point and the second one with the mesh. Furthermore, it assumes that all triangles of the mesh are oriented outside:

```
def isPointInsideMeshObject(point, meshObject):
    pointOnMesh, normalVector, faceIndex =
        meshObject.closest_point_on_mesh(point)
    vector = pointOnMesh - vertex
    dotProduct = vector.dot(normalVector)
    return not (dotProduct < 0.0)
```

Please note that 'closest_point_on_mesh' is a method of a mesh object that returns a point from the mesh that is closest to the point given in the argument of this method. This method is a part of Blender's API.

2.1. Results of Intersection Analysis

Using the approach presented above, we were able to identify how many vertices of objects of one layer are inside other layers. The results are presented in Table 3.

The cell in the i-th row and the j-th column agrees with a proportion of vertices from the i-th layer that are inside objects from the j-th layer. For example, in case of the 1-st row, value 3.01% means that about 5,920 from 196,787 vertices that are in skeletal layer (Table 1) are inside objects that skeletal layer is composed of. Similarly, number 23.19% means that about 45,635 from 196,787 vertices that creates skeletal layer are inside muscles. In case of the 2-nd row, the number 12.70% indicates that about 3,768 from 296,735 vertices that muscular layer is composed of (Table 1) are situated within objects of skeletal layer.

Layer	**Skeletal**	**Muscular**	**Connec.**	**Lymph.**	**Circul.**	**Nervous**
Skeletal	3.01%	23.19%	24.83%	0.06%	0.71%	0.65%
Muscular	12.70%	18.30%	4.57%	0.03%	1.05%	0.52%
Connective	20.97%	17.23%	24.59%	0.01%	0.80%	0.09%
Lymphatic	1.94%	16.75%	0.47%	12.18%	3.56%	0.57%
Circulatory	17.18%	11.00%	1.24%	0.14%	9.22%	2.26%
Nervous	22.73%	11.81%	0.31%	0.03%	2.97%	3.64%

Table 3. Intersection analysis of Zygote

As we can see in Table 3, several numbers are really high. This indicates that a lot of overlapping objects exist in the analysed model. To explain these numbers, we performed a preliminary qualitative analysis that showed that some of the major issues are:

- existence of objects without thickness (for example, the skull is modelled as one solid object and, therefore, all elements inside it (e.g., the brain) are recognized as overlapping objects);
- inaccurate location of objects (for example, many muscles are terminated inside bones and not on their surface);
- filling procedure – there exist situations where application of these procedure is not appropriate (for example, the pericardium is modelled as a manifold with boundaries, therefore, it has to be closed; however, this results in degeneration of the pericardium, and all elements inside it will be recognized as overlapping elements (Fig. 11); more reasonable way would be to define some thickness of the pericardium).

3. Conclusion

In this paper, the approach for investigation of quality of a virtual model of a human body was presented. The presented approach was implemented in open source software Blender and was used to investigate one of the existing models. The analysis showed that a lot of intersections exist in this model. Some of them result from inaccuracies in the model, but others are caused by inappropriate use of filling procedure. In the future work, we will focus on solving this issue.

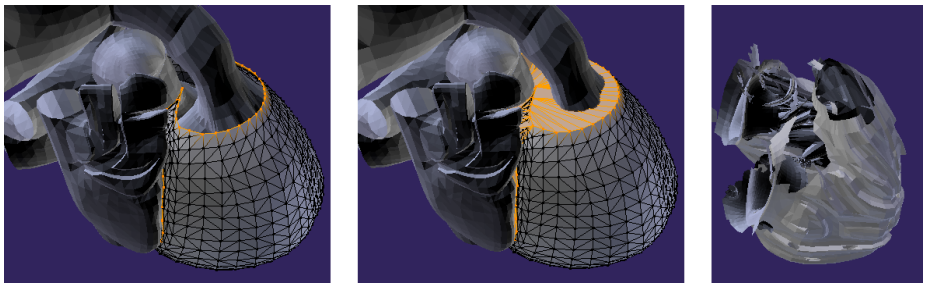


Figure 11. Pericardium and filling problem

Acknowledgment. The RASimAs project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610425.

References

[1] DESERNO, T. M. – Oliveira, J. E. E. – Grottke, O.: *Regional Anaesthesia Simulator and Assistant (RASimAs): medical image processing supporting anaesthesiologists in training and performance of local blocks*, 28th IEEE International Symposium on Computer-Based Medical Systems, Sao Carlos – Ribeirao Preto, June 22–25, 2015, pp. 348–351.
[2] RASimAs project: `http://www.rasimas.eu/`.
[3] Zygote: `http://www.zygote.com/`.
[4] Blender: `https://www.blender.org/`.
[5] MARSCHNER, S. – SHIRLEY, P.: *Fundamentals of Computer Graphics*, 4th ed., Boca Raton, CRC Press, 2015, ISBN 1-4822-2939-0.
[6] Function for deciding if a point is inside a mesh or not: `https://github.com/nortikin/sverchok/issues/660`.

Contact address

Ing. Miroslav Kvaššay, PhD., Department of Informatics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: `Miroslav.Kvassay@fri.uniza.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 85–92

TVORBA VLASTNÝCH APLIKÁCIÍ PRE INTERAKTÍVNE TABULE

DOMINIKA MARŠÁLEKOVÁ (SK), MIROSLAV BIŇAS (SK)
A MARTIN SARNOVSKÝ (SK)

Abstrakt. Modernizácia vyučovacieho procesu so sebou priniesla zavedenie interaktívnych tabúľ do škôl. Dodávaný softvér však neumožňuje vytvárať vhodné aplikácie pre kurzy venované programovaniu. V tomto článku sa preto venujeme téme tvorby vlastných aplikácií pre interaktívne tabule pomocou otvorených technológií.

Kľúčové slová. Interaktívna tabuľa, interaktívne aplikácie, programovanie.

DEVELOPMENT OF CUSTOM APPLICATIONS FOR INTERACTIVE WHITEBOARDS

Abstract. Modernization of educational process brought the implementation of interactive whiteboards to schools. The supplied software does not allow development of applications suitable for programming courses. In this article we focus on development of custom applications for interactive whiteboards with open technologies.

Keywords. Interactive whiteboard, interactive applications, programming.

Úvod

Interaktívna výučba nie je vo vzdelávaní žiadnou novinkou. Učitelia sa stále snažia nájsť spôsoby, ktorými by mohli preberanú problematiku podať študentom atraktívnejšou formou a tak študentov zaujať. Nie vždy je pritom nutné využiť súčasné technologické trendy, ale niekedy stačí len zmeniť zaužívaný spôsob výučby a zaujať študentov spoločnou aktivitou, počas ktorej sa môže prejaviť ich názor alebo kreativita.

V posledných rokoch sa vďaka mnohým projektom školy dovybavili rôznymi novými informačnými a komunikačnými technologickými pomôckami. Tie prácu učiteľom uľahčujú, čím sa aj výučba stáva pre žiakov príťažlivejšou a pestrejšou. No aj napriek tomu môže byť rola žiaka viac-menej pasívna a jeho aktívne zapojenie sa do procesu výučby môže byť nanajvýš v príprave vlastnej prezentácie alebo popisovaní premietaných obrázkov.

Medzi dodávanými pomôckami do škôl sú aj interaktívne tabule, ktoré by už z názvu mali do procesu výučby priniesť interakciu. Spolu s nimi je dodávaný aj softvér, ktorý učiteľom pomáha v tvorbe interaktívnych učebných materiálov. Pri

ich používaní sa pedagóg aj žiak môžu aktívne zúčastňovať výučby, ovplyvňovať ju a prispôsobovať svojim potrebám.

Nie všetky ponúkané aplikácie však dokážu svojimi možnosťami pokryť potreby každého kurzu. Do tejto kategórie spadajú hlavne kurzy venované programovaniu, ktoré majú špecifické požiadavky a aplikácie typu doplň kľúčové slovo nie sú vždy ideálne.

Nájsť vhodné príklady, ako sa dá interaktívna tabuľa využiť práve v kurzoch venovaných programovaniu, je ťažké aj v súčasnosti, keď máme k dispozícii obrovské množstvo kurzov dostupných prostredníctvom *MOOC* serverov, a keď vyhľadávanie v zdrojoch dostupných na internete nám uľahčujú internetové vyhľadávače. V tomto článku sa venujeme práve problematike tvorby vlastných aplikácií pre interaktívne tabule pomocou vhodných a hlavne otvorených technológií. Inšpiráciou pre ich vznik ako aj pre túto prácu sa stal kurz *CS50*[1], ktorý je úvodným kurzom do programovania na *Harvardskej univerzite*.

1. Príklady existujúcich aplikácií

Ako sme uviedli v úvode, využitie interaktívnej tabule v kurzoch venovaných programovaniu je možné vidieť v kurze *CS50*, ktorý svojim študentom ponúka *Harvardská univerzita*. V nasledujúcom texte sa pozrieme na niektoré aplikácie a technológie, ktoré je možné vo výučbe predmetov venovaných programovaniu použiť.

1.1. Prostredie *Scratch*

Pomocou prostredia *Scratch*[2] je možné programovať interaktívne príbehy, animácie, hry a prezentácie. Programy sú vytvárané jednoduchým skladaním farebných blokov, ktoré nahrádzajú kľúčové slová programovacieho jazyka. Posledná verzia prostredia *Scratch* je vytvorená pomocou technológie *Flash*.

Scratch je určený mládeži vo veku od 8 do 16 rokov, ale kvôli svojim vlastnostiam ho používajú ľudia všetkých vekových kategórií. Je populárny aj u nás, čomu svedčí aj preklad publikácie [2] do slovenčiny [1] alebo rozličné súťaže, ako napr. *Scratch Cup*[3].

1.2. Prostredie *Snap!*

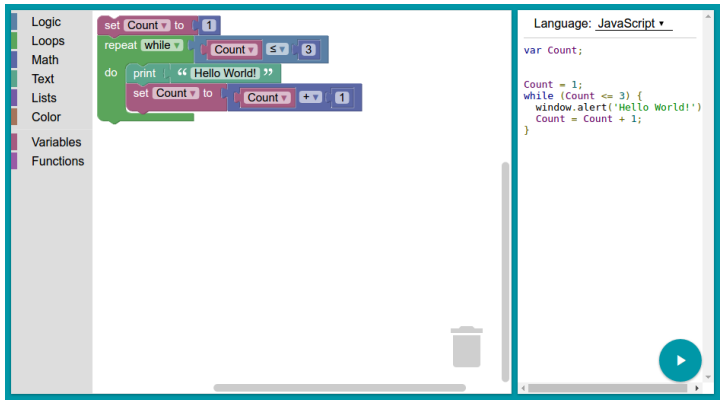
Prostredie *Snap!*[4] predstavuje alternatívnu implementáciu prostredia *Scratch*. Toto prostredie je vytvorené v jazyku *JavaScript* a umožňuje vytvárať vlastné bloky.

[1] `https://cs50.harvard.edu`
[2] `https://scratch.mit.edu`
[3] `http://edi.fmph.uniba.sk/~tomcsanyiova/ScratchCup/`
[4] `http://snap.berkeley.edu/`

1.3. Knižnica Blockly

Knižnica *Blockly*[5] je vytvorená spoločnosťou *Google* v jazyku *JavaScript*. Poskytuje vizuálny editor inšpirovaný prostredím *Scratch*, ktorý je možné umiestniť do vlastných (vzdelávacích) webových aplikácií.

Na obrázku 1 je zobrazená ukážka využitia tejto knižnice priamo na domovskej stránke projektu. Medzi ďalšie ukážky využitia tejto knižnice je možné zaradiť server `https://code.org/`, ktorý sa snaží popularizovať programovanie svojou iniciatívou *Hour of Code*. Všetky ponúkané prostredia tejto aktivity boli vytvorené práve pomocou knižnice *Blockly*.



Obr. 1. Ukážka využitia knižnice *Blockly*

1.4. Zhrnutie

Všetky uvedené riešenia sú ideálne na výučbu základov algoritmizácie a programovania pomocou interaktívnych tabúľ. Nie sú však vhodné na plnú výučbu programovania v univerzitnom prostredí.

2. Interaktívna tabuľa

J. Dostál definoval interaktívnu tabuľu v článku [3] ako *"dotykovú plochu, prostredníctvom ktorej prebieha vzájomná aktívna interakcia medzi používateľom a počítačom s cieľom zaistiť maximálnu možnú mieru názornosti zobrazovaného obsahu. Z technického hľadiska je možné interaktívnu tabuľu chápať ako elektronické zariadenie, ktoré je obvykle spojené s počítačom a dataprojektorom."*

Na Slovensku boli interaktívne tabule dodané do škôl prostredníctvom rôznych projektov, ktoré výrazne prispeli k modernizácii výučby. Medzi ne patrí napríklad národný projekt *Elektronizácia vzdelávacieho systému regionálneho školstva*,

[5] `https://developers.google.com/blockly/`

známy ako *Digiškola*[6], projekt *Planéta vedomostí* [7] či projekt *Digitálne vzdelávanie*[8].

Na tvorbu interaktívnych prezentácií a aplikácií sa používajú aplikácie, ktoré využívajú pedagógovia po celom svete. Medzi tie známejšie patria *SMART Notebook*[9], *Flow!Works*[10] a *Open-Sankoré*[11]. Všetky z nich dokážu vytvárať interaktívne prezentácie a mini aplikácie, vďaka čomu dokážu výučbu zatraktívniť.

3. Technológie pre tvorbu aplikácií pre interaktívne tabule

Každý pedagóg, ktorý vyučuje predmet venovaný programovaniu, má iné zvyky. Niektorí si na prednášku prinesú svoj vlastný notebook, v ktorom majú pripravenú prednášku spolu s prostredím, v ktorom bude pedagóg počas prednášky programovať. Nedá sa totiž spoľahnúť na to, že na počítači, na ktorom bude mať prednášať, bude mať nainštalované všetko potrebné. Niektorí zasa na prednáškach neprogramujú, ale len prezentujú. Tým stačí si na prednášku priniesť USB kľúč so svojou prezentáciou.

Predtým, ako začneme zvažovať výber vhodných technológií pre tvorbu aplikácií pre interaktívne tabule, by bolo dobré na základe stereotypov učiteľov stanoviť niekoľko kritérií:

1. Zvolená technológia by mala byť multiplatformná. Ak totiž prednášajúci používa OS Linux, na ktorom si pripraví interaktívne aplikácie, nemusí ich spustiť na prezentačnom počítači, ktorý používa iný operačný systém.
2. Aplikácie by mali byť vytvorené pomocou technológií a v jazyku, ktorý sa neviaže na platformu. Toto kritérium úzko súvisí s predchádzajúcim, ale môže sa týkať výslednej aplikácie, ktorá využíva napr. vlastnosti konkrétneho súborového systému alebo operačného systému.
3. Aplikácie by primárne nemali byť desktopovými aplikáciami. S nimi totiž súvisí konkrétna technológia a okrem samotnej aplikácie sa v kóde môže objaviť priveľa sprievodného kódu (napr. vytvorenie okna, ošetrenie jeho ukončenia a pod.). Rovnako tak môže byť problematická údržba takto vytvorených aplikácií, keď nemusí byť zaručené, že aplikácia vytvorená vo verzii X danej technológie bude podporovaná aj na cieľovom počítači používanom na prezentáciu.
4. Krivka učenia zvolených technológií by mala byť čo najstrmšia. Aj keď je možné predpokladať, že učitelia programovania sú zbehlí v programovaní, nemusia byť zbehlí v každej technológii alebo jazyku.

[6] `http://www.digiskola.sk/`
[7] `http://www.planetavedomosti.sk/`
[8] `http://www.digitalnevzdelavanie.sk/`
[9] `http://smarttech.com/`
[10] `http://qomo.com`
[11] `http://open-sankore.org/`

Ak začneme tieto kritériá aplikovať na súčasné jazyky a technológie, môžeme z nich vyradiť aj jazyky ako *Java* alebo platformu *.NET*. Naopak, ako vhodná technológia spĺňajúca uvedené kritériá sa zasa javí *HTML5*, pretože jediný predpoklad, ktorý je potrebné splniť, je mať na cieľovom počítači nainštalovaný prehliadač s povoleným jazykom *JavaScript*. Na vývoj aplikácií stačí ľubovoľný textový editor a webový prehliadač, čo je tiež značnou výhodou.

4. Tvorba aplikácií

Pri tvorbe aplikácií sme si stanovili jednu dôležitú podmienku: vytvárané aplikácie pre interaktívne tabule musia vyžadovať zadávanie vstupných informácií v čo najmenšej miere. Ak teda bude aplikácia vyžadovať napr. vyplnenie formulára, nie je pre použitie na interaktívnej tabuli najvhodnejšia. Pod interakciou si totiž predstavujeme niečo úplne iné, ako je zadávanie vstupných textov z klávesnice zobrazenej na tabuli.

Samotný jazyk *JavaScript* je vo svojej podstate veľmi surový. Preto sme hľadali tiež vhodné rozšírenia, ktoré by nám tvorbu vlastných aplikácií uľahčili. Pre naše experimenty sme používali knižnicu *jQuery*[12].

Na tvorbu prezentácií sme používali *HTML5* rámec *reveal.js*[13], vďaka čomu sa vytvárané aplikácie stali priamou súčasťou týchto prezentácii.

V nasledujúcom texte budú predstavené vytvorené aplikácie a predmety, na ktorých boli použité.

4.1. *Základy algoritmizácie a programovania* a *Programovanie*

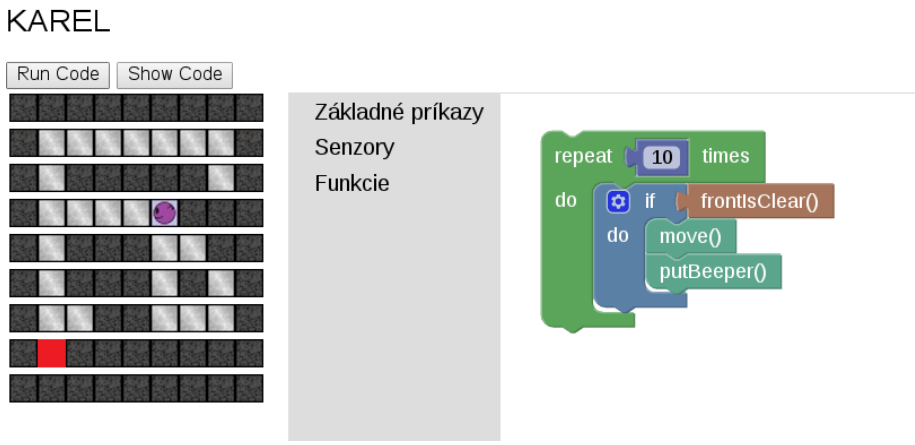
Predmet *Programovanie* bol prvý, na ktorom sme začali aplikácie pre interaktívnu tabuľu vytvárať. Na začiatku sa však jednalo len o implementácie aplikácií, ktoré sme videli v kurze *CS50*. Jednalo sa o dve aplikácie, ktoré prezentovali operáciu vyhľadávania na neutriedenom a utriedenom zozname, kde úlohou dobrovoľníka bolo práve nájsť požadovaný údaj. To nám však stačilo pre overenie zvolených technológií.

Po zahrnutí *regulárnych výrazov* do osnov predmetu vznikol nápad vytvoriť jednoduchý validátor, ktorý by sa stal súčasťou prezentácie a slúžil by na okamžité overenie regulárneho výrazu. Tento typ aplikácie však porušuje nami stanovenú podmienku, pretože si vyžaduje zadať ako vstupný text, tak aj regulárny výraz. To sme z časti vyriešili priamym zadaním daného vstupného textu. Úlohou študentov bolo napísať regulárny výraz. Výhodou tohto prístupu je aj to, že študenti si môžu otestovať regulárny výraz sami, pretože prezentácia s aplikáciou je voľne dostupná.

[12] `https://jquery.com/`

[13] `http://lab.hakim.se/reveal-js/`

Ďalšiu inšpiráciu sme našli v spomínanej aktivite *Hour of Code*, ktorú sme si vyskúšali priamo počas prednášky z predmetu *Základy algoritmizácie a programovania*. Požiadali sme dobrovoľníka z publika, s ktorým sme skúsili priamo na prednáške vyriešiť dva scenáre zvolenej hry. V jednom prípade sme však interaktívnu tabuľu k dispozícii nemali a museli sme si vystačiť s dotykovou obrazovkou dostupnou na katedre. Tá však nebola dostatočne citlivá, ako tomu bolo v prípade interaktívnej tabule. Na základe skúsenosti s touto aktivitou sme pomocou knižnice *Blockly* vytvorili jednoduché prostredie pre jazyk *Robot Karel* (viď obr. 2), ktorý používame v začiatočných hodinách tohto kurzu.

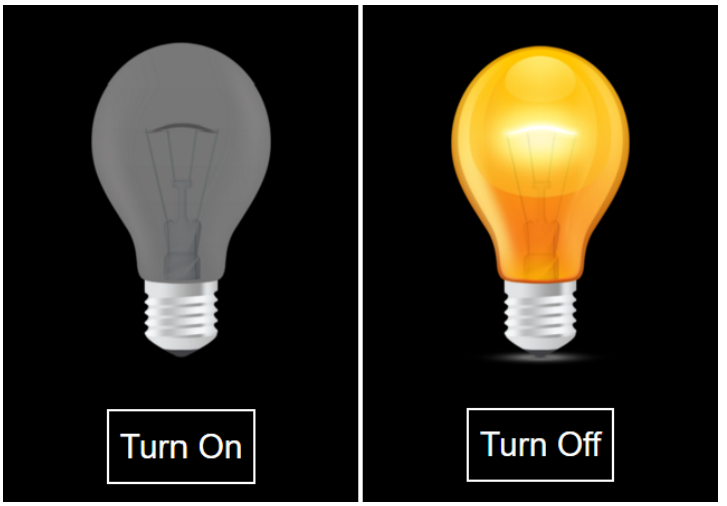


Obr. 2. Vývojové prostredie pre jazyk *Robot Karel*

4.2. *Vývoj aplikácií pre chytré zariadenia*

Tento predmet sa venoval vývoju aplikácií pre *OS Android*. Vznikol však nápad, že aplikácie, ktoré budeme vytvárať počas prednášok, je možné vyrobiť vo forme prototypu ako webové aplikácie. Tak vznikli dve aplikácie: aplikácia *Torch* a aplikácia *MrIlko*. Aplikácia *Torch* (na obr. 3) bola prvou aplikáciou vytvorenou počas prednášok. Prezentovala rozhranie aplikácie baterky. Aplikácia *MrIlko* predstavovala jednoduchého klienta pre službu `openweathermap.org` a poskytovala päťdňovú predpoveď počasia.

Výhoda zvolených technológií sa naplno prejavila v oboch prípadoch. V prvom prípade sa žiarovka v aplikácii rozsvecovala a zhasínala po stlačení príslušného tlačidla. V druhom prípade sa jednalo o plne funkčnú aplikáciu, ktorá vytvárala HTTP požiadavky na REST API príslušnej služby a zobrazovala skutočné údaje.



Obr. 3. Aplikácia *Torch* (baterka)

4.3. *Vývoj počítačových hier*

Súčasťou prvej prednášky tohto predmetu bolo vytvorenie a overenie vytvorenej hry jej zahratím spolu so študentom, ktorý sa dobrovoľne zúčastnil tohto duelu. Aby bol proces ťahov transparentný, vytvorili sme hraciu kocku (viď obr. 4), ktorou sme striedavo so študentom hádzali. Jediným problémom bolo, že v miestnosti sa nenachádzala interaktívna tabuľa, takže sme s kockou hádzali v prehliadači na prezentačnom počítači.

Obr. 4. Hracia kocka

5. Záver

Tvorba aplikácií pre interaktívne tabule pomocou *HTML5* technológií je nenáročná a je možné si ju rýchlo osvojiť. Prednášky, na ktorých sme ku tabuli zavolali študenta, boli vždy atraktívnejšie, čo pozitívne zhodnotili aj samotní študenti v dotazníku ku práci [4]. Rovnako pozitívne sa vyjadrili aj ďalší pedagógovia, ktorým sme výsledky práce prezentovali.

Samotné aplikácie sú často rozsahovo nenáročné. Výhodou tiež je, že nemusia byť nutne súčasťou prezentácie vytvorenej v jazyku *HTML*, ale prezentácia sa môže na aplikáciu len odkazovať. Tým je možné zabezpečiť aj údržbu aplikácií, kedy sa do prezentácie dostane vždy najaktuálnejšia verzia aplikácie bez akéhokoľvek sťahovania. Tento prístup môže viesť až k vytvoreniu galérie ako je napr. `http://www.texample.net/`.

Na tejto práci sa aktívne podieľala študentka 3. ročníka bakalárskeho štúdia odboru *Hospodárska informatika* Dominika Maršáleková, ktorá svoje výsledky spracovala v bakalárskej práci [4]. Tým, že sama tento typ výučby zažila v 1. a 2. ročníku, vedela vo výsledkoch svojej práce zúročiť práve svoje skúsenosti. A keďže s uvedenými technológiami nemala predchádzajúcu skúsenosť, je tiež dôkazom strmej krivky učenia sa týchto technológií.

Literatúra

[1] *Programovanie pre deti, Naučte sa programovať pri tvorbe skvelých hier*, Computer Press, 2014, ISBN 9788025141120.

[2] *Super scratch programming adventure!: learn to program by making cool games*, No Starch Press, 2014, San Francisco, CA, ISBN 1593275315.

[3] DOSTÁL, J.: *Interaktivní tabule ve výuce*, Journal of Technology and Information Education, 2009, ISSN 1803-537X

[4] MARŠÁLEKOVÁ, D.: *Tvorba aplikácií pre interaktívne tabule*, Technická univerzita v Košiciach, 2016, Letná 9, Košice, Bakalárska práca.

Kontaktné adresy

Dominika Maršáleková, Katedra kybernetiky a umelej inteligencie, Fakulta elektrotechniky a informatiky, Technická univerzita v Košiciach, Letná 9, 042 00 Košice, Slovensko,
E-mailová adresa: `dominika.marsalekova@student.tuke.sk`

Ing. Miroslav Biňas, PhD., Katedra počítačov a informatiky, Fakulta elektrotechniky a informatiky, Technická univerzita v Košiciach, Letná 9, 042 00 Košice, Slovensko,
E-mailová adresa: `miroslav.binas@tuke.sk`

Ing. Martin Sarnovský, PhD., Katedra kybernetiky a umelej inteligencie, Fakulta elektrotechniky a informatiky, Technická univerzita v Košiciach, Letná 9, 042 00 Košice, Slovensko,
E-mailová adresa: `martin.sarnovsky@tuke.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 93–98

GNU RADIO TOOLKIT AND ITS APPLICATION IN THE AREA OF RF SIGNAL PROCESSING

REMIGIUSZ OLEJNIK (PL)

Abstract. The article presents GNU Radio toolkit and RTL-SDR tuner hardware. It is also shown how to build simple FM radio receiver in GNU Radio with RTL-SDR used as a radio signal source.

Key words and phrases. RTL-SDR, GNU Radio, FM radio receiver.

GNU RADIO TOOLKIT A JEHO APLIKÁCIA V OBLASTI SPRACOVANIA RF SIGNÁLOV

Abstrakt. Článok prezentuje GNU Radio toolkit a RTL-SDR tuner hardvér. Je tiež ukázané, ako vytvoriť jednoduchý FM rádio prijímač v GNURadio s RTL-SDR použitým ako zdroj rádiového signálu.

Kľúčové slová. RTL-SDR, GNU Radio, FM rádio prijímač.

Introduction

Although the idea of a programmable radio is not new, recent years have brought progress within the area of using low-cost hardware alternatives. At the same time GNU Radio appeared, which is a versatile environment for radio frequency (RF) signals processing. This article presents the GNU Radio environment, RTL-SDR tuner and an example of the construction of the FM radio receiver.

The first part of the article answers to the question: what is GNU Radio? The second part provides information on the RTL-SDR tuner hardware and its potential applications. The third part shows how to build FM radio receiver in GNU Radio environment that will use RTL-SDR hardware. The article concludes with a brief summary.

1. What is GNU Radio?

GNU Radio [1] is an open source, free software development toolkit which provides signal processing blocks to implement software-defined radios and signal processing systems. It can read input data from external RF hardware to create software-defined radios (SDR), or without any additional hardware in a simulation-like environment. It's main audience are users from hobbyist, academic, and commercial environments since it is helpful both in real-world radio

systems and wireless communications research. The application used by GNU Radio is named as *flowgraph* and is composed of the series of signal processing blocks connected together, describing a data flow. These applications can be written in either Python or C++. GNU Radio Companion (**GRC**) is a GUI used to develop GNU Radio applications [3], similar to Simulink known from MATLAB environment. GRC is bundled with the GNU Radio package distribution.

GRC is a Python code-generation tool. After *flowgraph* compilation in GRC, Python code is generated. It creates the desired GUI windows and widgets, and creates and connects the blocks in the *flowgraph*. Reconfigurability is a key feature, important especially when connecting to different external RF signal inputs. It means that a general purpose device can be used as a radio front-end while GNU Radio will process incoming signals according to the configured purpose.

GNU Radio performs all the signal processing using *blocks*: channel codes, decoders, demodulators, equalizers, filters and many more.

GNU Radio provides many data plotting and visualization features, including Fast Fourier Transform (FFT) displays, symbol constellation diagrams, and scope displays, which are widely used both for debugging radio applications and as the user-interface to a final application.

GNU Radio system was published in 2001 for the first time, with Eric Blossom as original author sponsored by American philanthropist John Gilmore, which donated US$ 320000 for code creation and project management duties. Currently it is maintained by GNU Project with Ben Hilburn as the Project Lead and Johnathan Corgan as the Chief Architect. It is written entirely in C++, while many user tools are written in Python. It is licensed under GNU General Public License (GPL), and most of the project code is copyrighted by the Free Software Foundation [2] Latest version of GNU Radio is v3.7.9.2 and was published on 12th of April, 2016.

1.1. What is Software Defined Radio (SDR)?

Software Defined Radio (SDR) is an idea of substituting a radio communication system components that have been typically hardware components (filters, amplifiers, modulators/demodulators, detectors) by software implementations running on a personal computer or embedded system.

In recent years low-cost DVB-T USB dongles based on the Realtek RTL2832U controller and the Elonics E4000 or Rafael Micro R820T tuner are used as wide band SDR receivers. They can be used also as input devices for GNU Radio toolkit, what will be shown later.

1.2. How to install GNU Radio

GNU Radio is available through package management installer command in most of today Linux distributions — name of the package is simply **gnuradio**. However

it sometimes doesn't work after default installation and the compilation from sources is needed. Another way for starting using GNU Radio is to use bootable DVD with GNU Radio pre-installed [4].

1.3. Bootable DVD with GNU Radio pre-installed

As mentioned above, the easiest way to use GNU Radio is to boot a PC from DVD with GNU Radio pre-installed [4]. The DVD is based on Ubuntu Linux 14.04.4 LTS, 64-bit edition, and has additional software installed beyond the defaults supplied by Ubuntu. GNU Radio Release 3.7.9.2 is installed, along with the drivers for most common GNU Radio-compatible SDR devices, like RTL-SDR.

Pre-installed GNU Radio applications. More than 20 packages that highlight some of the diverse capabilities of GNU Radio are pre-installed [4], and the most interesting are:

- **gr-osmosdr**, a set of blocks supporting various hardware SDRs and applications, and related dependencies,
- **gr-fosphor**, a GPU-accelerated real-time spectrum analyzer display,
- **gqrx**, a narrow band SDR AM, FM and SSB receiver application,
- **GNSS-SDR**, a fully software-based GPS, Galileo, and GLONASS real time receiver,
- **gr-specest**, an implementation of several spectral estimation algorithms,
- **OpenLTE**, an open source 3GPP LTE implementation,
- **gr-nacl**, wrapper blocks around NaCL encryption library,
- **gr-adsb**, an ADS-B modem,
- **gr-ais**, an AIS modem,
- **gr-burst**, a burst PSK modem,
- **gr-cdma**, a CDMA phy,
- **gr-gsm**, blocks and tools for receiving GSM transmissions,
- **gr-ieee-80211**, an IEEE 802.11a/g/p transceiver,
- **gr-lte**, an implementation of a 3GPP LTE receiver,
- **gr-radar**, the GNU Radio Radar Toolbox,
- **gr-rds**, an implementation of broadcast FM radio RDS reception.

Many tutorials related to installation and using of GNU Radio are available on-line [5].

2. RTL-SDR tuner

RTL-SDR is a SDR approach based on a relatively cheap (< US$10) USB dongle DVB-T tuner based on Realtek RTL2832U chipset – see Figure 1. Depending on particular tuner chip used (most common are Elonics E4000 and Rafael Micro R820T), it provides receive capability at least in 52–1766 MHz range. RTL2832U chipset ADC native resolution is 8 bits, but the Effective Number of Bits (ENOB)

Figure 1. RTL-SDR hardware: USB DVB-T tuner based on RTL2832U chipset, made in China

is estimated at ≈ 7. Nominal input impedance is 75 Ω, since these tuners are intended for digital TV reception. Using standard 50 Ω antennas and cables will introduce small mismatch resulting in small loss (around 0.177 dB). Standard antenna shipped together with RTL tuners is very inefficient broadband vertically polarized whips. It is suggested to use better antennas, in order to get more signals with higher strength — for radiomonitoring purposes quite good solution can be Discone type antenna. It is also possible to enable reception of LF and HF signals using an external up-converter — using a converter with 100 MHz oscillator all the signals will be shifted by 100 MHz. RTL-SDR used along with adequate software can be powerful SDR solution intended for VHF/UHF bands monitoring.

There are many well-documented possible applications of the RTL-SDR [6]:

- listening to unencrypted Police/Ambulance/Fire conversations,
- listening to aircraft traffic control (ATC) conversations,
- listening to VHF amateur radio,
- listening to FM radio, and decoding RDS information,
- listening to DAB broadcast radio,
- listening to satellites and the ISS,
- watching digital (DVB-T) and analogue broadcast TV,
- receiving GPS signals and decoding them,
- receiving NOAA weather satellite images,
- receiving wireless temperature sensors and wireless power meter sensors,
- sniffing GSM signals,
- scanning for cordless phones and baby monitors,
- scanning trunking radio conversations,
- decoding ham radio APRS packets,
- decoding aircraft ACARS short messages,

- decoding unencrypted digital voice transmissions,
- decoding POCSAG/FLEX pager traffic,
- decoding taxi mobile data terminal signals,
- tracking aircraft positions like a radar with ADSB decoding,
- tracking maritime boat positions like a radar with AIS decoding,
- tracking and receiving meteorological agency launched weather balloon data,
- using RTL-SDR on an Android device as a portable radio scanner,
- using RTL-SDR as a spectrum analyzer,
- using RTL-SDR as a panadapter for a traditional hardware radio,
- using RTL-SDR as a high quality entropy source for random number generation,
- using RTL-SDR as a noise figure indicator,
- radio astronomy,
- reverse engineering unknown protocols,
- triangulating the source of a signal,
- searching for RF noise sources,
- characterizing RF filters and measuring antenna SWR.

RTL2832U based tuner can be in an easy way used with GNU Radio environment. Next part of the article presents example of RTL-SDR based GNU Radio application for FM radio reception.

3. RTL-SDR based GNU Radio FM radio receiver

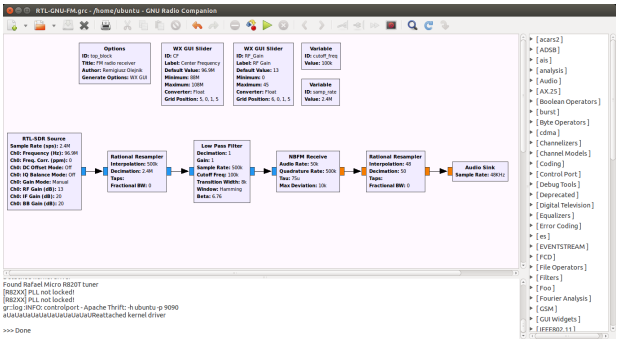


Figure 2. *Flowgraph* of RTL-SDR based GNU Radio FM radio receiver

The goal of the project that is shown in Figure 2 is to demodulate one of the ubiquitous FM broadcast radio stations. RTL2832U-based tuner is an input device that acts as an RF signal source — **RTL-SDR source** block. The signal is then passed to the **Low Pass Filter** and **NBFM**[1] **Receive** demodulator. Final stage is **Audio Sink** — PC's audio card. Since the input signal sample

[1]Narrow Band Frequency Modulation

rate is 2.4 Msps (***samp_rate*** variable) and the audio card needs just 48 ksps, **Resampler** blocks are used, in order to down-sample the signal. With WX GUI sliders, RF Gain — ***RF_Gain*** variable — and center frequency — ***CF*** variable — can be set. Figure 3 shows resulting GUI view.

During the OSSConf 2016 conference, functioning of the FM radio receiver will be presented in real-time.

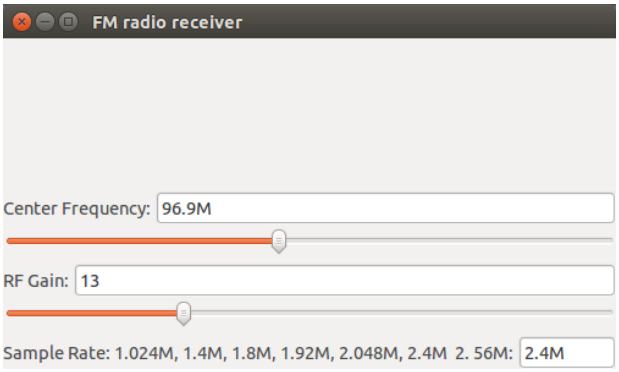


Figure 3. GUI of RTL-SDR based GNU Radio FM radio receiver

4. Conclusions

The article presents GNU Radio environment along with its features. RTL-SDR tuner is also shown with a description of its possible applications. The construction of FM radio receiver is an example of mutual cooperation of GNU Radio and RTL-SDR hardware.

References

[1] `http://gnuradio.org/`.
[2] `http://gnuradio.org/redmine/projects/gnuradio/wiki/WhatIsGR#How-is-GNU-Radio-licensed`.
[3] `http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion`.
[4] `http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioLiveDVD`.
[5] `http://gnuradio.org/redmine/projects/gnuradio/wiki`.
[6] `http://www.rtl-sdr.com/about-rtl-sdr/`.

Contact address

Dr inż. Remigiusz Olejnik, Department of Computer Architecture and Telecommunication, Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, Szczecin, ul. Żołnierska 49, 71-210 Szczecin, Poland,
E-mail address: `rolejnik@zut.edu.pl`, `http://rolejnik.zut.edu.pl`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 99–104

USNADNĚNÍ TVORBY DOKUMENTŮ V APLIKACI TEXONWEB

JAN PŘICHYSTAL (CZ)

Abstrakt. Článek se zabývá možností využití webové aplikace TEXonWeb ve výukové oblasti. Diskutuje problémy, které se objevují při použití tohoto nástroje ve výuce předmětu Zpracování textů na počítači. Na základě identifikovaných problémů hledá jejich řešení a navrhuje konkrétní vylepšení a rozšíření aplikace mezi které patří využití našeptávače značek a sdílení dokumentů. Na závěr shrnuje dosažené výsledky a srovnává je s konkurenčními řešeními. Naznačuje také další možnosti rozšiřování a vylepšování vytvořeného řešení.

Klíčová slova. Našeptávač, sdílení, TEXonWeb, TEX/LATEX, výuka, webová aplikace.

FACILITATION OF THE DOCUMENT CREATION IN TEXONWEB APPLICATION

Abstract. The article deals with the possibility of using the web application TEXonWeb in the educational field. Discusses the problems that occur when you use this tool in the course Computer typesetting. Based on the identified problems looks for solutions and suggests specific improvements and expansion of the application including the use of auto-complete and document sharing. In conclusion summarizes the results and compares them with competing solutions. It also suggests other possibilities for expansion and improvement of created solution.

Keywords. Autocomplete, sharing, TEXonWeb, TEX/LATEX, education, web application.

Úvod

V současné době je aplikace TEXonWeb využívána především studenty Mendelovy univerzity při výuce předmětu Zpracování textů na počítači. S touto aplikací pracují zejména na cvičeních, kdy se seznamují se způsobem tvorby jednotlivých částí dokumentu pomocí systému TEX a jeho nadstavby LATEX. Aplikaci využívá pravidelně každý semestr téměř dvě stě studentů. Aplikace nabízí studentům mnoho výhod oproti běžným editorům a je tak optimální výukovou pomůckou. Mezi hlavní výhody patří například možnost využívat editor bez nutnosti instalace a konfigurace systému TEX, možnost práce odkudkoliv prostřednictvím webového prohlížeče, využívání předdefinovaných šablon dokumentů či pomocníků tvorby komplikovanějších částí dokumentu. TEXonWeb je při výuce tohoto předmětu využíván řadu let a neustále se vylepšuje, tak aby odpovídal měnícím se požadavkům na výuku i na způsob používání webových aplikací. I z tohoto

důvodu byla například implementována podpora pro mobilní zařízení. Podrobněji se jednotlivým rozšířením věnuje [3].

Současný stav

Při výuce předmětu Zpracování textů na počítači na Provozně ekonomické fakultě Mendelovy univerzity v Brně se vyučující často potýkají s řadou problémů, které studentům na cvičeních komplikují tvorbu dokumentů v editoru TEXonWeb. Mezi nejpalčivější problémy patří například neschopnost správně zapsat TEXové či LATEXové struktury. Dochází k tomu, že studenti nesprávně zapíší název makra, zaměňují různé typy závorek, nesprávně strukturují zanořené konstrukce či zapomínají otevřené závorky uzavírat. Spousta z těchto chyb pramení z obtížného opisování textů z tabule či projektoru, kde vyučující vysvětluje a demonstruje probíranou látku. Studenti sice mají k dispozici i materiály v PDF formátu obsahující osnovu cvičení včetně použitých značek, ty ale často nevyužívají a navíc tyto osnovy nepokrývají vše, o čem se na cvičení hovoří. Z tohoto důvodu obsahuje editor TEXonWeb, jako výuková pomůcka, řadu vlastností, která mají tyto problémy odstranit. Mezi ně patří například zvýrazňování syntaxe maker a zobrazování párových závorek. Nicméně ani toto podle dlouhodobého pozorování nestačí. Studenti při zápisu maker stále chybují a vyučující stráví spoustu času obcházením jednotlivých počítačů a řešením obdobných a přitom zbytečných chyb.

Analýza problému

Pro řešení výše zmíněných problémů se nejprve podíváme na jiné editory a to nejenom pro tvorbu dokumentů, ale i programové editory, protože tvorba TEXového dokumentu spočívá v podstatě v zápisu kódu.

Pokud se zaměříme na to, jakým způsobem usnadňují programové editory uživatelům tvorbu kódu, najdeme zde důležitou funkcionalitu, kterou označujeme jako našeptávač či doplňovač kódu. Ten uživateli napovídá správný název příkazu programovacího jazyka nebo libovolné jiné struktury. Tím zrychluje zápis neboť není nutné psát celý identifikátor, ale tento je editorem automaticky po stisknutí definované klávesové zkratky doplněn. Navíc, a to je pro nás důležitá vlastnost, editor doplní kód struktury správně bez překlepů. Tuto vlastnost nabízí i editor Ace, který je integrován do aplikace TEXonWeb. Jak uvádí dokumentace [1], lze využít vlastnosti `enableBasicAutocompletion` a našeptávač zapnout. Následně je nezbytné definovat vlastní množinu značek, protože pro TEX a LATEX není v editoru Ace definována.

Pokud se podíváme, jaké možnosti usnadnění tvorby dokumentů nabízí online editory dokumentů, mezi které TEXonWeb patří, nalezneme zde zajímavou vlastnost umožňující skupině uživatelů spolupracovat na tvorbě jednoho dokumentu.

Tuto možnost nabízí například Google Drive, Microsoft Office 365 a další podobné služby. Na první pohled nemusí být použitelnost této vlastnosti ve výuce patrná. Nicméně, pokud se zamyslíme, vyučující a studenti vlastně vytváří jeden stejný dokument, avšak každý na svém počítači. Studenti si opisují to, co vidí na tabuli nebo projektoru. A právě ve fázi opisování často chybují. Nebudeme zde rozebírat, proč k chybám dochází, proč o textu, který vytváří, více nepřemýšlí, to je do jiné diskuse. Důležitý je ale fakt, že vyučující i studenti text píší v aplikaci TEXonWeb. Pokud by existovala možnost tento dokument studentům zpřístupnit, ti by si mohli vybrané části číst přímo na svém monitoru a případně si je i zkopírovat. Tím by došlo k výraznému snížení chybovosti zápisu zdrojového kódu dokumentu.

Je samozřejmě možné dohodnout se ve studijní skupině a prostřednictvím jedné z výše zmíněných služeb sdílet dokument. To je však zbytečně zdlouhavé, protože vyučující musí vlastní kód kopírovat do jiného dokumentu. Vhodnější řešení je tedy přímé nasdílení dokumentu v rámci aplikace TEXonWeb.

Návrh řešení

Jak bylo uvedeno výše, vlastnost našeptávání maker TEXu či LATEXu lze do editoru Ace přidat předefinováním vlastnosti `enableBasicAutocompletion` na hodnotu `true` a definováním funkce vracející seznam možných klíčových slov v závislosti na zapsaném podřetězci. Zdrojový kód je naznačen v ukázce uvedené níže.

```
ace.require("ace/ext/language_tools");
var editor = ace.edit("editor");
editor.setOptions({
    enableBasicAutocompletion: true
});
var autoCompleter = {
  getCompletions:
    function(editor, session, pos, prefix, callback) {
    if (prefix.length === 0) { callback(null, []); return }
       $.getJSON("getWords&word=" + prefix, wordList);
    }
}
```

Uživatel tak v podstatě jen začne psát úvodní znaky zamýšleného makra a následně stiskne klávesovou zkratku `Ctrl+mezerník` a v editoru se mu zobrazí kontextová nabídka se seznamem předdefinovaných značek. Je jasné, že našeptávač ve webové aplikaci nemá takové možnosti, jako v samostatném editoru. Umožňuje nabízet pouze předdefinovanou množinu značek. Značky, které si uživatel

definuje sám, nenabízí. Důvodem je, že by musel v reálném čase analyzovat kód dokumentu a připojených definičních souborů, což by způsobovalo velkou zátěž.

Sdílení dokumentů v aplikaci TEXonWeb má dvě úrovně. Jednak existuje možnost prostého zpřístupnění dokumentu mezi uživateli tak, aby se nabízený dokument objevil ve správci souborů jiného uživatele. Ten si pak může dokument otevřít či zkopírovat a dále s ním pracovat. Tento způsob sdílení funguje tak, že v okamžiku, kdy uživatel vybere určitý soubor ke sdílení, zkopíruje se soubor ostatním uživatelům do jejich pracovních adresářů. Avšak při kopírování souborů se do pracovních adresářů nekopírují celé soubory, ale pouze symbolické odkazy. To umožní mít sdílený soubor na jednom místě, a ostatní němu přistupují prostřednictvím symbolického odkazu. To značně usnadňuje implementaci i správu sdílených souborů. Tato možnost však neumožňuje sledovat změny v dokumentu provedené. Editor v aplikaci TEXonWeb v takové situaci pouze zobrazí aktuální stav, tak jak jej uložil poslední uživatel z množiny spolupracovníků. Aby studenti mohli sledovat, jakým způsobem vyučující modifikuje dokument, museli by neustále v prohlížeči obsah editoru obnovovat a to je velmi nekomfortní. Tento způsob lze ve výuce využít například v situaci, kdy student chyběl v minulém cvičení a nemá k dispozici rozpracovaný dokument.

Druhá úroveň sdílení dokumentu je možnost aktivní spolupráce na dokumentu, tak jak je známa i z jiných cloudových služeb. Podobně, jako v předchozím případě, je nutné vybraný soubor nasdílet skupině uživatelů, kteří chtějí změny sledovat. Následně si všichni zúčastnění otevřou v editoru soubor a v textu dokumentu mohou sledovat kurzory ostatních uživatelů opatřené jejich jmény a barevně odlišené pro snadnou identifikaci. V případě, že některý z nich začne dokument editovat, jsou změny distribuovány i všem ostatním a ti mohou v reálném čase změny sledovat. Navíc, v záhlaví dokumentu je k dispozici nabídka se seznamem všech uživatelů, kteří dokument v daný okamžik prohlíží. Pokud některý z uživatelů uloží dokument, uloží se změny všem, tak jak logicky vyplývá z principu sdílení s prostřednictvím symbolických linků.

Pro potřeby výuky je pak vhodný takový postup, kdy vyučující pracuje přímo na sdíleném dokumentu a studenti mají otevřená dvě okna editoru v aplikaci TEXonWeb. V jednom je otevřený jejich vlastní soubor s vytvářeným dokumentem a v druhém je sdílený soubor, kde sledují prováděné změny a případně si odtud kusy kódu kopírují. Tím je zajištěna poměrně velká šance správného zápisu kódu.

Pro zpřístupnění dokumentu všem spolupracujícím uživatelům je využita technologie WebSocket, která zajišťuje full-duplexní komunikaci mezi klienty prostřednictvím serveru [2]. Konkrétní implementaci celého řešení pro spolupráci uživatelů na tvorbě dokumentů pak podrobně popisuje [5].

Závěr a zhodnocení

Tento článek se věnuje návrhu způsobu řešení problémů při výuce předmětu Zpracování textů na počítači ve webové aplikaci TEXonWeb.

Možnosti našeptávače jsou v tuto chvíli omezené pouze na definovanou množinu značek, která je pro všechny uživatele stejná. Možné vylepšení by mohlo spočívat v rozšíření definice vlastním seznamem značek, tak jak existuje možnost přidávat si vlastní slova do slovníku korektoru pravopisu. Každý uživatel by tak měl možnost si množinu personalizovat o své vlastní často využívané konstrukce.

Aplikace TEXonWeb v současné chvíli neumožňuje při sdílení dokumentů definovat odlišné role uživatelů. Všichni uživatelé jsou rovnocenní a mohou dokument sledovat i upravovat. Pro potřeby výuky by rozhodně bylo vhodné implementovat možnost role *pouze pro čtení*, aby studenti nemohli zasahovat do sdíleného dokumentu. Bylo by též vhodné umožnit vyučujícímu definovat jednoduchým způsobem seznam uživatelů, se kterými chce dokument sdílet. V současné chvíli je nezbytné studenty přidávat po jednom pomocí jejich loginů. To je poměrně zdlouhavé a bylo by efektivnější, kdyby bylo možné přidat celou skupinu uživatelů naráz.

Nabízí se také srovnání navrženého řešení s konkurenčními projekty jako jsou shareLATEX či Overleaf. V současné době oba tyto projekty diskutovanou funkcionalitu nabízí. Projekt Overleaf umožňuje neomezenou spolupráci více uživatelů i našeptávání kódu. Naproti tomu projekt shareLATEX umožňuje bezplatnou práci pouze jednomu uživateli. Pokud bychom chtěli dokument sdílet s dalšími uživateli je nutné zaplatit 14 respektive 28 Euro za měsíc pro podporu deseti respektive neomezený počet spolupracovníků. Studentské tarify jsou za poloviční cenu. Funkce našeptávače je zde také k dispozici.

Z tohoto pohledu je jasné, že implementací diskutovaných vlastností aplikace TEXonWeb pouze dohání konkurenční projekty, které je již mají k dispozici. Jak uvádí [4], aplikace TEXonWeb však nabízí další možnosti, které v konkurenčních projektech nejsou dostupné, a tak si svou existenci stále dokáže obhájit.

Reference

[1] *How-To Guide*, Ace – The High Performance Editor for the Web, `https://ace.c9.io/#nav=howto`, [cit. 2016-06-09].

[2] LIU, W. T.: *Research on the Development of WebSocket Server*, Advanced Materials Research, 2014, 886, 694–697.

[3] PŘICHYSTAL, J.: *Možnosti tvorby dokumentů v TeXu pomocí webového prohlížeče*, In Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach, Zborník príspevkov medzinárodnej konferencie OSSConf 2014, Spoločnosť pre otvorené informačné technológie, 2014, s. 23–30, ISBN 978-80-970457-4-6.

[4] PŘICHYSTAL, J.: *Návrhář stylů v aplikaci TEXonWeb*, In Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach, Zborník príspevkov medzinárodnej konferencie OSSConf 2015, Spoločnosť pre otvorené informačné technológie, 2015, s. 71–76, ISBN 978-80-970457-7-7.

[5] PŘICHYSTAL, J. – TELENSKÝ, V.: *Collaboration of users creating documents in the application TeXonWeb*, 18th International Conference Enterprise And Competitive Environment 2014, Mendel University in Brno, 2015, pp. 747–755, ISBN 978-80-7509-342-4.

Kontaktní adresa

Ing. Jan Přichystal, Ph.D., Provozně ekonomická fakulta, Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika,

E-mailová adresa: `jan.prichystal@mendelu.cz`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 105–117

ELASTICSEARCH PRO PROSTOROVÁ DATA

JAN RŮŽIČKA (CZ) A VOJTĚCH LHOTSKÝ (CZ)

Abstrakt. Příspěvek se zabývá nástrojem Elasticsearch a jeho využitím pro práci s prostorovými daty. V první části jsou stručně popsány případy užití Elasticsearch s prostorovými daty. Druhá část popisuje základní informace o Elasticsearch a způsob ukládání dat a metodiku vyhledávání dat v něm uložených. Příklady prostorových dotazů a analýz, které je možno provádět v Elasticsearch, se zabývá třetí část. Poslední část obsahuje porovnání doby odezvy dotazování provedeného nad prostorovými daty v Elasticsearch a PostGIS.

Klíčová slova. Elasticsearch, prostorová data, prostorové dotazy, prostorové analýzy, doba odezvy, PostGIS.

ELASTICSEARCH FOR SPATIAL DATA

Abstract. The paper aims to provide a description of the ability of Elasticsearch to work with spatial data. The first part of the paper succinctly introduces the usecases of Elasticsearch with spatial data. The second part explains basic information about the way Elasticsearch works and how it stores data and how it carries out spatial or non spatial queries upon data that are stored in it. Examples of spatial data and analyses that are feasible in Elasticsearch are to be found in the third part of the paper. The last part of the paper contains a comparison of response times of spatial queries in Elasticsearch and PostGIS.

Keywords. Elasticsearch, spatial data, spatial queries, spatial analysis, time response, PostGIS.

Úvod

V posledních letech dochází k velkému nárůstu objemu dat, který vedl ke zvýšeným nárokům především na nástroje pro skladování, analýzu a vyhledávání dat. Program ElasticSearch spojuje tyto činnosti a umožňuje skladovat data a zároveň na nich provádět efektivní vyhledávání a analýzy. Elasticsearch je široce využíván pro práci s neprostorovými daty. Možnost pracovat s prostorovými daty však byla přidána do Elasticsearch relativně nedávno a je zde stále místo pro přidávání nových funkcí a rozšíření. Elasticsearch stále nabývá na popularitě a v budoucnu se bude jistě jeho role v práci s prostorovými daty nadále rozšiřovat. Cílem příspěvku je obecně popsat nástroj Elasticsearch a jakým způsobem uchovává prostorová i neprostorová data. Dalším z cílů příspěvku je popsat způsob vkládání prostorových dat do Elasticsearch a uvést příklady jednoduchých prostorových dotazů, které je možno v Elasticsearch provádět. Posledním z cílů příspěvku je celkově

zhodnotit Elasticsearch pro práci s prostorovými daty. Vyhodnocení posledního cíle je provedeno pomocí výčtu různých prostorových analýz a jejich grafických výstupů provedených v nástroji Kibana a porovnání délky doby odezvy dotazování vůči databázi PostgreSQL využívající PostGIS [1].

1. Použití

Typickým příkladem užití Elasticsearch s prostorovými daty je využití firmou tapSW [4] (Theory and practice of software) pro mapování poštovních směrovacích čísel ve Velké Británii pro taxi společnost bookingDialog, která nabízí možnost *on-line* rezervace taxi služby. Cílem je vytvořit geokódér a nástroj, pomocí kterého můžeme vyhledávat adresy. Užívání tohoto softwaru by mělo být především rychlé i navzdory uchovávání velkého množství dat. Prvním úkolem bylo připravení souboru obsahujícího informace o vztahu mezi poštovními směrovacími čísly a zeměpisnými souřadnicemi a souboru obsahující všechny názvy ulic. Po propojení těchto souborů v tabulce by stačilo, aby zákazník zadal korektní název ulice a operátor taxi služby by získal všechny potřebné informace k zajištění dopravy pro zákazníka. Bohužel člověk není neomylný a je tedy potřeba, aby i při zadání chybného názvu ulice byly alespoň podobné názvy ulic nalezeny a zákazníkovi nabídnuty. Elasticsearch tuto toleranci na chybné zadání vyhledávání obsahuje a zároveň umožňuje práci s prostorovou složkou dat. Elasticsearch je tedy dobrou volbou pro tento úkol.

Dalším příkladem užití Elasticsearch je prostorová analýza sociálního média od Michaela Kaissera [5]. Cílem je prozkoumat a zjistit co se děje na sociálních médiích v tomto případě především v okolí Berlína a následně provést vybrané prostorové analýzy. Zvolená sociální média byla Twitter, Instagram a Foursquare. Dat na sociálních sítích je obrovské množství a efektivní způsob přístupu a analýzy těchto dat je velmi žádaný. Kolem 5% všech příspěvků na sociální síti Twitter za jeden den je spojeno s prostorovými souřadnicemi, u Instagramu a Foursquare se jedná o 100% všech příspěvků. Elasticsearch umožňuje práci s takzvanými agregacemi, které slouží k seskupení všech bodů nacházejících se v určité oblasti. Nadefinováním těchto oblastí jako čtverců, které tvoří síť pokrývající celou zájmovou oblast a agregací všech bodů ležících v jednotlivých čtvercích je možné určit místa s vysokou koncentrací aktivit.

2. Komunikace a ukládání dat

Všichni klienti mohou komunikovat s nástrojem Elasticsearch za pomocí RESTful API [6]. Komunikace probíhá přes adresu počítače a port 9200. Můžeme využívat například nástroj cURL nebo plugin Sense, který je vyvíjený samotnou firmou stojící za Elasticsearch.

Elasticsearch využívá pro dokumenty datový formát JSON [7]. JSON je způsob zápisu dat. Je čitelný a zapisovatelný člověkem. Skládá se z kolekce párů klíčů a hodnot, které jsou umístěny v objektu. JSON obsahuje datové typy textový řetězec, číslo, logická hodnota (boolean), hodnota null, pole a objekt.

Elasticsearch umožňuje skladování a vyhledávání prostorových dat. Prostorová data jsou skladována za pomoci datového formátu GeoJSON. Elasticsearch je optimalizován a podporuje pouze práci se zeměpisnou šířkou a zeměpisnou délkou v rámci světového geodetického systému WGS-84.

GeoJSON [8] je rozšíření datového formátu JSON. Umožňuje zápis prostorových dat. Přidané datové typy jsou bod, multibod, linie, multilinie, polygon, multipolygon.

1. Bod – Bod je umístěn do prostoru párem souřadnic.
2. Multibod – Kolekce více bodů.
3. Linie – Linie spojuje dva body. Je dána dvěma páry souřadnic.
4. Multilinie – Kolekce více linií.
5. Polygon – Plocha tvořená uzavřenou linií, v polygonu se mohou nacházet díry.
6. Multipolygon – Kolekce více polygonů.

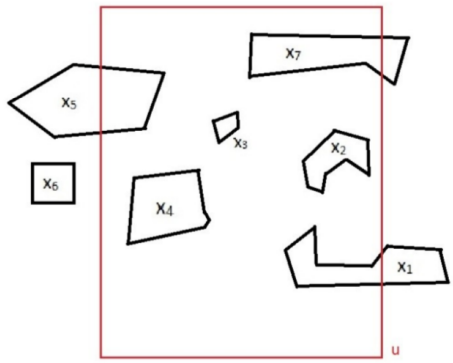
3. Prostorové dotazy

Query by geography (dotaz na polohu) slouží pro vyhledávání záznamů podle jejich polohy v prostoru a prostorových vztahů mezi nimi. Tento způsob dotazování je v rámci tohoto příspěvku velmi důležitý a především na něj bude během popisu dotazů v Elastisearch kladen důraz. Prostorové vztahy nazýváme topologické operace. Platí pro body, linie a polygony. Využitím topologických operací můžeme formulovat nejrůznější prostorové dotazy. Jako příklad můžeme vidět na obrázku 1 vyhledávání všech polygonů x, které se nacházejí uvnitř ohraničujícího obdélníka u. K vyhledávání využíváme prostorový vztah typu *within* (česky nachází se uvnitř). Z celkového počtu sedmi polygonů budou vybrány pouze tři polygony a to x_2, x_3 a x_4.

Následuje výčet nejběžnějších typů prostorových vztahů (topologické operace). Prvky cílové vrstvy představují prvky, které jsou vybrány a prvky výchozí vrstvy představují prvky, na které se dotazujeme.

Intersect: (česky protíná) vybere všechny prvky v cílové vrstvě, které se překrývají nebo zasahují do výchozí vrstvy. Umožňuje použití bodů, linií i polygonů jako výchozí nebo cílové vrstvy. V Elasticsearch je označeno jako vztah typu intersects.

Disjoint: (česky disjunktní) vybere všechny prvky cílové vrstvy, které se nepřekrývají ani nezasahují do výchozí vrstvy. Umožňuje použití bodů, linií i polygonů jako výchozí nebo cílové vrstvy. V Elasticsearch je označeno jako vztah typu disjoint.



Obrázek 1. Operace WITHIN

Contains: (česky obsahuje) vybere všechny prvky, ve kterých se nachází výchozí vrstva. Umožňuje použití bodů, linií i polygonů jako výchozí vrstvy a pouze polygony jako cílové vrstvy. V Elasticsearch je označeno jako vztah typu contains.

Within: (česky nachází se uvnitř) vybere všechny prvky, které se celé nacházejí ve výchozí vrstvě. Umožňuje použití bodů, linií a polygonů jako cílové vrstvy a pouze polygony jako výchozí vrstvy. V Elasticsearch je označeno jako vztah typu within.

Are within a distance of: (česky je ve vzdálenosti od) vyžaduje definování vzdálenosti u výchozí vrstvy, kolem které se následně vytvoří v zadané vzdálenosti takzvaný buffer (česky obalová zóna), s prvkem výchozí vrstvy jako jeho středem. Všechny prvky cílové vrstvy, které protínají nebo se nacházejí v této obalové zóně, budou vybrány. Umožňuje použití bodů jako výchozí vrstvy a bodů jako cílové vrstvy. V Elasticsearch se provádí pomocí distance filter (vzdálenostní filtr).

Následující příklady ukazují, jakým zpusobem je možno dotazy konstruovat.

3.1. Prostorový dotaz číslo 1

Najdi všechny body do vzdálenosti x od výchozího bodu b. Dotaz tohoto typu je jeden z nejpraktičtějších prostorových dotazů, které můžeme provést v Elasticsearch. Elasticsearch vytvoří kruhový buffer (obalovou zónu) kolem bodu u ve vzdálenosti x a všechny dokumenty, které obsahují body, které se v této obalové zóně nacházejí, jsou vybrány. Příklad užití: Po zadání vlastní polohy do aplikace v mobilním telefonu nalezne daná aplikace všechny restaurace ve vzdálenosti 500 metrů od naší polohy. Dále můžeme seřadit restaurace podle nejvyššího

hodnocení kvality restaurace samotnými zákazníky. Příklad můžete vidět na obrázku 2. Dotazování provádíme nad indexem restaurace v okolí bodu, který se nachází v centru Ostravy a hledáme všechny restaurace ve vzdálenosti distance 1 kilometr a méně. V klíči *sort* definujeme, že si přejeme výsledky vyhledávání seřadit od nejvyššího po nejnižší hodnocení.

```
POST restaurace/_search
{   "query": {
    "filtered": {
      "filter": {
        "geo_distance": {
          "distance": "1km",
          "poloha": { "lat": 49.844080,
                      "lon": 18.264702  }}}},
    "sort": [{ "hodnoceni" : "desc" }]
}
```

Obrázek 2. Prostorový dotaz číslo 1

3.2. Prostorový dotaz číslo 2

Najdi všechny body od vzdálenosti x do vzdálenosti z od výchozího bodu b. Elasticsearch vytvoří dva kruhové buffery (obalové zóny) a jejich překrytím vznikne mezikruží ve vzdálenostech x a z. Všechny body, které se nachází v tomto mezikruží, jsou vybrány. Je možné vytvořit libovolný počet takovýchto mezikruží. Příklad užití: Vyhledání hotelu v blízkosti centra města na základě ceny za ubytování. Cena hotelu bude vysoká pakliže se nacházíme v centru města (první pásmo). Po opuštění centra, ale v jeho těsné blízkosti (druhé pásmo) bude cena nižší. Jako poslední bude oblast nacházející se daleko od centra města (třetí pásmo), kde budou ceny hotelů teoreticky nejnižší. Pakliže chceme být ubytování blízko centra, ale zároveň neplatit příliš drahé ubytování, zvolíme druhé pásmo. Příklad můžete vidět na obrázku 3. Dotazování provádíme nad indexem *ubytovani*. Pomocí funkce *must* stanovíme, že si přejeme zobrazit všechny body, které projdou filtrem. Následně v klíči *from* (česky odkud) a klíči *to* (česky kam) zadáme požadované vzdálenosti, ve kterých se vytvoří pásma a všechny body spadající do jednotlivých pásem budou vybrány. Poloha centra udává z. šířku a z. délku bodu, který se přibližně nachází ve středu centra města.

3.3. Prostorový dotaz číslo 3

Najdi všechny polygony, které protínají výchozí multinii x. Elasticsearch porovná pomocí prostorového vztahu typu intersects multinii x se všemi polygony a najde ty polygony, které protíná. Příklad užití: Vybrání měst, které protíná železniční trasa. Příklad můžete vidět na obrázku 4. Dotaz provádíme nad indexem mesta. V klíči *geo_shape* nastavíme multilinii, která představuje železniční

```
POST ubytovani/_search
{   "bool": {
        "must": {
            "match_all": {}},
        "filter" : {
            "geo_distance_range": {
                "from": "1km",
                "to":   "3km",
                "polohaCentra" : {  "lat": 49.820394,
                                    "lon": 18.262333  }}}}
}
```

Obrázek 3. Prostorový dotaz číslo 2

trasu a je uložená jako list bodů. Vztah typu intersects je výchozí a nemusíme ho nastavovat.

```
POST mesta/_search?pretty
{   "query": {
      "geo_shape" : {
        "location" : {
          "shape" : {
            "type" : "multilinestring",
            "coordinates" : [[  [18.015271,49.642734],
                                [18.212256,49.669903],
                                [18.371447,49.714038],
                                [18.159301,49.645366]  ]] }}}}
}
```

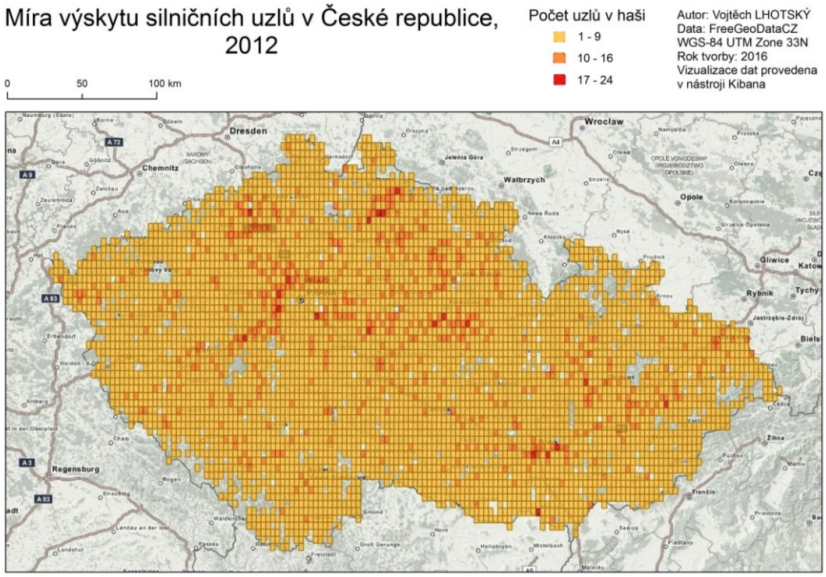
Obrázek 4. Prostorový dotaz číslo 3

4. Agregační dotazy a jejich vizualizace

Předmětem této části jsou analýzy dat uložených v Elasticsearch a důraz je kladen především na jejich prostorovou složku. Výhodou Elasticsearch je především provádění všech úkonů v reálném čase. To platí také pro grafické zobrazování rozsáhlých prostorových analýz na mapě v nástroji Kibana, které pokrývají například celou Českou republiku.

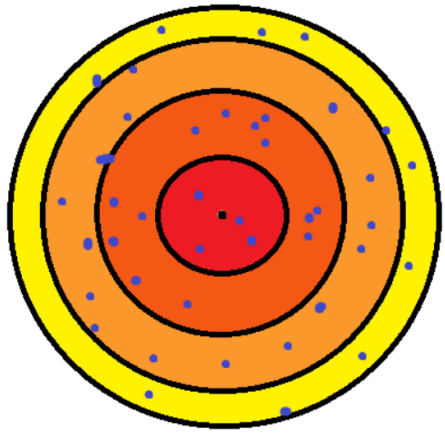
Pro ilustrační účely byla vybrána vrstva *silnice_uzly* z datasetu FreeGeoDataCZ, která obsahuje 22 126 dokumentů (bodů) rozmístěných po celé České republice a ty byly agregovány do sítě o rozměru jedné buňky přibližně 6 kilometrů na 5 kilometrů. Výsledek je na obrázku 5.

Jiným příkladem může být analýza pomocí agregace podle vzdálenosti. Během této analýzy vycházíme z jednoho bodu (střed) a kolem tohoto bodu vytvoří Elasticsearch v námi zvolených vzdálenostech pásma (obr. 6). Body (dokumenty) nacházející se v těchto pásmech jsou agregovány a zjistíme tak, kolik bodů se



Obrázek 5. Mapa míry výskytu silničních uzlů v České republice, 2012

v jednotlivých pásmech nachází. Využití při stanovení dopadů určitého jevu, např. havárie v chemickém závodě.



Obrázek 6. Ilustrace agregace bodů při agregaci podle vzdálenosti

Dotaz k provedení agregace můžete vidět na obrázku 7. Dotaz provádíme nad indexem body. V agregaci nastavíme cestu k prostorové složce bodů v klíči *field*,

uvedeme jednotky vzdáleností a zadáme výchozí bod v klíči origin (střed). V klíčích *from* (česky odkud) a *to* (česky kam) zadáme jednotlivá pásma pomocí rozsahů vzdáleností v kilometrech.

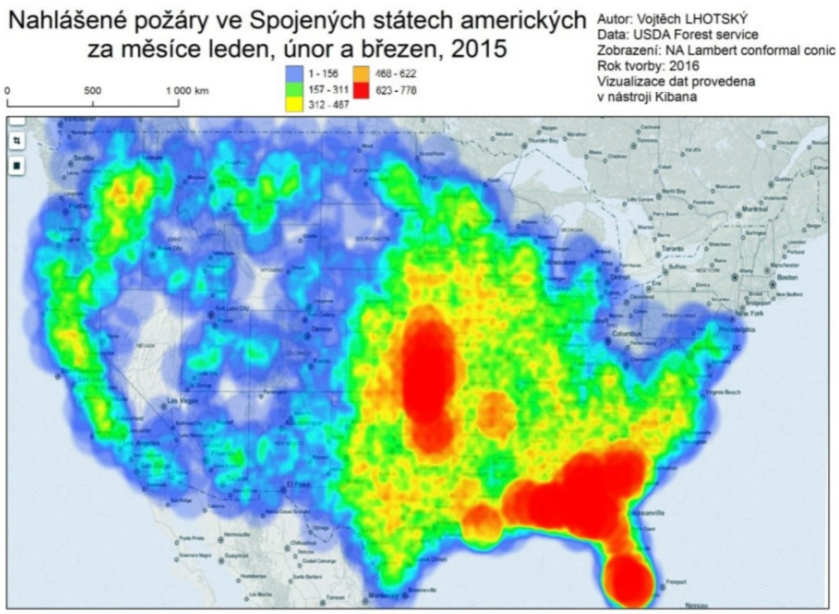
```
POST body/_search
     {   "aggs": {
         "mojeAggr" : {
           "geo_distance" : {
             "field" :    "point_location_bettermap",
             "unit" :     "km",
             "origin" : { "lat": 49.532623,
                          "lon": 17.469139  },
             "ranges" : [ { "from" : 0, "to" : 1 },
                          { "from" : 1, "to" : 2 },
                          { "from" : 2, "to" : 3 },
                          { "from" : 3, "to" : 4 } ] }}}
     }
```

Obrázek 7. Příklad dotazu pro agregaci bodů podle vzdálenosti

Jiným příkladem může být zobrazení dat za určité časové období. Elasticsearch podporuje datový typ datum a umožňuje tak vyhledávat data za určitá časová období. Možnost využití časové a prostorové složky dat tvoří velmi silnou kombinaci sloužící k analytickým účelům. Pro ilustrační účely byla použita vrstva *modis_fire_2015_365_conus* [9], která obsahuje 148 249 záznamů nahlášených požárů v Spojených státech amerických za rok 2015. Na obrázku 8 je demonstrováno využití údajů o poloze těchto požárů, pomocí kterých jsou agregovány v mapě a časových údajů, díky kterým je možné vybrat výskyt požárů v jednotlivých kvartálech roku 2015. Dotaz k provedení agregace je zobrazen na obrázku 9. Dotazování provádíme nad indexem *fires*. Klíč *size* s hodnotou 0 udává, že nechceme žádné dokumenty vybírat, ale pouze je agregovat. Klíč *query* říká, že hledáme všechny dokumenty, které se nachází v časovém období od 1. 1. 2015 do 31. 3. 2015. V klíči *aggs* definujeme typ agregace *geohash_grid* (agregace podle sítě haší) a v klíči *field* nastavíme klíč v dokumentech indexu *fires*, který obsahuje prostorovou složku dat. Přesnost je nastavené na 4. Velikost jedné buňky je přibližně 40 kilometrů na 20 kilometrů. Změnou časového období v každé agregaci můžeme vytvořit sérii map představující výskyt nahlášených požárů ve Spojených státech amerických během roku 2015.

5. Časová odezva při dotazování

Velkou výhodou Elasticsearch je jeho rychlost při vyhledávání dat, které probíhá v reálném čase. Porovnávat jeho funkční možnosti s prostorovými daty vůči např. PostGIS by nemělo velký smysl. V jedné oblasti by však Elasticsearch měl mít navrch a sice v rychlosti vyhledávání. Porovnány byly doby odezvy při vybraných prostorových dotazech v obou nástrojích. Pro test byly vybrány pouze dva



Obrázek 8. Mapa nahlášených požárů ve Spojených státech amerických za měsíce leden, únor a březen v roce 2015

```
POST fires/_search
{   "size": 0,
    "query" : {   "filtered" : {
       "query" : {
          "query_string" : {
             "analyze_wildcard" : true,
                 "query" :  "*"  }},
        "filter" : {
           "bool" : {
              "must" : [{
                 "range" : {
                    "properties.DATE" : { "gte" : "2015-01-01",
                                          "lte" : "2015-03-31" }}}] }}}},
    "aggs" : {
       "2" : {
         "geohash_grid" : {
            "field" : "point_location_bettermap",
            "precision" : 4     }}}
}
```

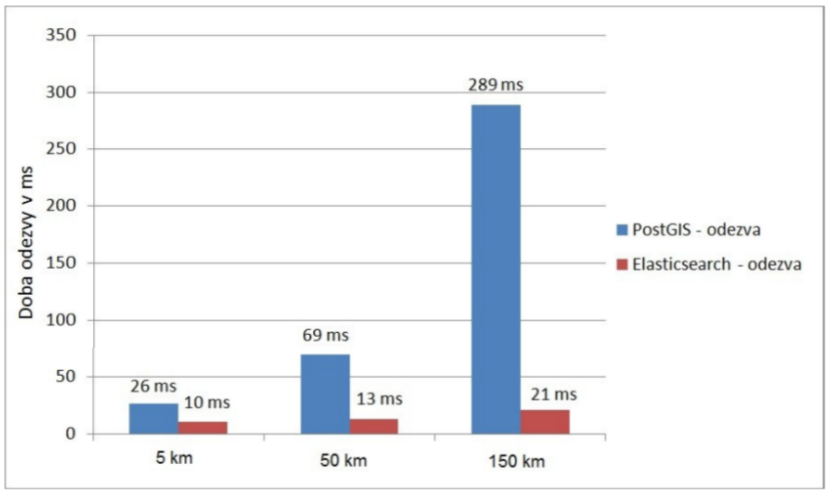
Obrázek 9. Dotaz pro agregaci bodů v mapě nahlášených požárů v USA za leden, únor a březen 2015

dotazy z testovaných dotazů. Zbylé dotazy nebyly testovány. Test byl prováděn na počítači s procesorem Intel Core i5-3350P a 8 GB pamětí a operačním systému

Windows 8.1 a pomocí programovacího jazyka Python. Použité verze softwaru byly následující: Elasticsearch 2.3.0, Postgresql 9.3, PostGIS 2.1.7 a Python 2.7.8.

5.1. Dotaz číslo 1

Najdi všechny body do vzdálenosti x od bodu y. Vyhledávání bylo prováděno nad vrstvou *silnice_uzly*. Je důležité zmínit, že data uložená v PostGIS byla indexována pomocí indexu GIST. Každé měření bylo opakováno 10 krát a výsledné časy jsou průměry naměřených hodnot. Na obrázku 10 můžeme vidět výsledky rychlosti zpracování dotazů. Ve vzdálenosti 5 kilometrů bylo vybráno 23 bodů, ve vzdálenosti 50 kilometrů bylo vybráno 2 333 bodů a ve vzdálenosti 150 kilometrů bylo vybráno 16 818 bodů.



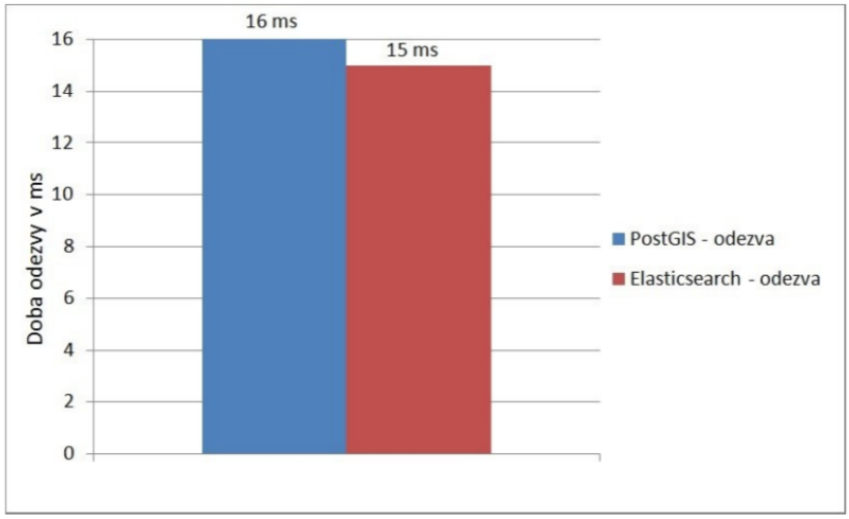
Obrázek 10. Doba odezvy výběru bodů u PostGIS a Elasticsearch ve vzdálenosti 5 km, 50 km a 150 km

Je vidět, že při nižší vzdálenosti vyhledávání jsou oba nástroje přibližně na stejné úrovni. Se vzdáleností 50 kilometrů se však doba odezvy u PostGIS významně navyšuje, zatímco u Elasticsearch zůstává nízká. U vzdálenosti 150 kilometrů je potom doba odezvy u PostGIS značně vyšší než u počátečních dvou vzdáleností, zatímco Elasticsearch se drží na přibližně stejné hodnotě jako na počátku.

5.2. Dotaz číslo 2

Najdi všechny polygony dotýkající se či nacházející se v polygonu x. Vyhledávání bylo prováděno nad vrstvou voda z datasetu FreeGeoDataCZ. Data v PostGIS byla opět indexována pomocí indexu GIST. Každé měření bylo provedeno

10 krát a výsledné časy jsou průměry naměřených hodnot. Na obrázku 11 můžeme vidět výsledky rychlosti zpracování dotazů.



Obrázek 11. Doby odezvy výběru polygonů v polygonu u PostGIS a Elasticsearch

V tomto případě byly naměřené doby odezvy dotazů téměř totožné. Celkem bylo vybráno 923 polygonů.

6. Diskuse

Můžeme konstatovat, že prvotní výsledky ukazují na zajímavé možnosti tohoto nástroje. Bude však nutné provést ještě mnoho dalších testů. Rádi bychom databázi naplnili větším množstvím dat a vyzkoušeli více různých typů dotazů. Také máme v plánu testovat možnosti nasazení ElasticSearch v síti počítačů tvořících cluster.

7. Závěr

Elasticsearch je široce využívaný nástroj pro ukládání a vyhledávání neprostorových dat. Je důležité brát ohled na skutečnost, že Elasticsearch je mladý nástroj a možnost práce s prostorovými daty byla do něj přidána relativně nedávno. Elasticsearch se bude jistě nadále v tomto odvětví vyvíjet a rozšiřovat své funkční možnosti, především by se mohlo jednat o možnost práce s rozdílnými souřadnicovými systémy. Potenciální využití Elasticsearch v oblasti práce s prostorovými je však již viditelné v této fázi vývoje. Prostorové dotazy, které je možné v Elasticsearch provádět, pokrývají široké spektrum běžných potřeb uživatelů moderních aplikací. Velkou předností Elasticsearch je vyhledávání v reálném

čase, díky kterému můžeme velmi rychle zobrazovat, pomocí vhodného grafického nástroje, výsledky analýz prováděných i nad velkým vzorkem dat. Elasticsearch je ideálním nástrojem pro provádění analýz dat ze sociálních médií obsahující prostorovou složku. Využití Elasticsearch v oblasti práce s prostorovými daty pramení především z nejrůznějších typů agregací neboli slučování. Možnost využití překryvných operací, společně s vysokou rychlostí provedení vyhledávání, je další velkou výhodou v této disciplíně. Nejzajímavější vlastností Elasticsearch je tedy jeho rychlost ukládání a vyhledávání prostorových i neprostorových dat.

Poděkování. Tento příspěvek vznikl s pomocí volně dostupných dat [2] a s pomocí nástrojů cURL, ElasticSearch, PostgreSQL, PostGIS, pgAdmin, a dalších open source nástrojů.

Reference

[1] Refractions Research: *PostGIS*, `http://postgis.org`, [cit. 2015-06-08].
[2] Landa, M.: *FreeGeodataCZ*, `http://freegis.fsv.cvut.cz/gwiki/FreeGeoDataCZ`, [cit. 2015-06-08].
[3] Open Street Map comunity: *Open Street Map*, `http://www.openstreetmap.org`, [cit. 2015-06-08].
[4] TapSW: *Mapping UK postcodes using Elasticsearch*, 2014, `http://www.tapsw.com/software-development-elasticsearch-uk-postcodes.php`, [cit. 1.2.2016].
[5] Kaisser, M.: *Geospatial analysis of social media posts with Elasticsearch*, 2014, `https://berlinbuzzwords.de/sites/berlinbuzzwords.de/files/media/documents/michael_kaisser_-_geospatial.pdf`, [cit. 1.2.2016].
[6] Elkstein, M.: *Learn REST: A Tutorial*, `http://rest.elkstein.org/`, [cit. 5.4.2016].
[7] ECMA: *Introducing JSON*, `http://www.json.org/`, [cit. 9.2.2016].
[8] BUTLER, H. et al.: *The GeoJSON format specification*, `http://geojson.org/geojson-spec.html`, [cit. 9.2.2016].
[9] USDA Forest Service, Remote sensing applications center: *MODIS Active fire detections for the CONUS*, 2016, `http://activefiremaps.fs.fed.us/data/fireptdata/modisfire_2015_conus.htm`.

Kontaktní adresy

Ing. Jan Růžička, Ph.D., Institut geoinformatiky, VŠB-TUO, 17. listopadu 15, 70833, Ostrava, Česká republika,
E-mailová adresa: `jan.ruzicka@vsb.cz`, `http://gis.vsb.cz/ruzicka/`

Bc. Vojtěch Lhotský, Institut geoinformatiky, VŠB-TUO, 17. listopadu 15, 70833, Ostrava, Česká republika,
E-mailová adresa: `vojtech.lhotsky.st@vsb.cz`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 117–124

JEDNODUCHÉ NENÍ JEDNODUCHÉ – ŘÁDKOVÝ PROKLAD POD LUPOU

JIŘÍ RYBIČKA (CZ)

Abstrakt. Úprava dokumentů vyžaduje určitou znalost základních zákonitostí sazby, možností nastavování parametrů, jejich funkce a technické řešení v textových procesorech. Zásadním parametrem je řádkový proklad, příspěvek shrnuje typografické aspekty pro jeho určování a zabývá se implementací příslušných voleb pro řádkový proklad v běžně užívaných textových procesorech. Přehled možností je doplněn komentářem o možných omylech při používání běžných textových procesorů a o nedostatcích vyskytujících se v různých doporučeních pro formátování dokumentů, jako jsou různé studentské průběžné a závěrečné práce.

Klíčová slova. Řádkový proklad, vlastnosti fontu, stupeň písma, design dokumentu.

SINGLE IS NOT SIMPLE – LINE SPACING UNDER THE MAGNIFYING GLASS

Abstract. Document design requires some knowledge of the basic typographic rules, basic typesetting parameters, their functions and of the technical solutions in word processors. One of the key parameters is the line spacing. This paper summarizes aspects for line spacing setting and deals with the implementation of appropriate options in the commonly used word processors. A list of options is enhanced by comments about possible mistakes when using common word processors and on drawbacks occurring in various recommendations for formatting documents, such as student term papers and final theses.

Keywords. Line spacing, font metrics, point size, document design.

1. Úvod

Úprava tiskovin se odvíjí od několika základních parametrů, jejichž vhodnou volbou můžeme dosáhnout maximálního efektu, naopak při nevhodném nastavení je takový dokument přinejmenším neestetický, v horších případech pak i obtížně čitelný a nesnadno zpracovatelný čtenářem.

Hlavním parametrem je bezesporu základní písmo (mnohdy je to jediný typ, který je použit pro všechny prvky dokumentu). Atributem základního písma je stupeň (velikost) a s tím úzce související šíře sazby.

Pro základní úpravu potřebujeme také určit geometrické charakteristiky odstavců základního textu, tj. odstavcové zarážky, způsob zarovnání, řádkový proklad a případné vertikální mezery.

Každý z uvedených parametrů má tedy zásadní vliv na celkovou úpravu dokumentu, bylo by tedy vhodné, aby úpravce měl o jejich hodnotách naprosto jasnou představu, znál přesně jejich funkci a uměl je technicky zvládnout v používaném programovém vybavení.

V následujícím textu se budeme zabývat jedním z těchto parametrů, a to řádkovým prokladem.

2. Co je řádkový proklad?

V prvé řadě je potřebné upřesnit samotný pojem řádkového prokladu. V odborné literatuře a v mnoha internetových zdrojích různé kvality se někdy obsah tohoto pojmu liší. Pro naše účely se budeme držet významu, jak jej popisuje Kočička a Blažek (2004, s. 25) – proklad je hodnota, kterou se prokládají jednotlivé řádky textu. Dříve se jednalo o velikost proložky, která se fyzicky vkládala mezi řádky. Podobně vnímá tento pojem i Beran (2003, s. 10) a zároveň dodává, že toto zvětšování a zmenšování mezer mezi řádky má výrazný vliv na čitelnost textu. Jde tedy o vertikální mezeru měřenou obvykle ve stejných jednotkách jako stupeň písma, nebo v relativním vyjádření závislém na stupni písma.

Velmi často se setkáváme s podobným pojmem – řádkování. Je to dáno především tím, že v běžně dostupném programovém vybavení je uveden jako jediná možnost nastavení řádkového prokladu a navazuje na technologii psacích strojů. Řádkování je zde však vždy míněno jako vzdálenost dvou po sobě jdoucích účaří v odstavci, tedy nikoliv jako přídavná mezera, která se vkládá mezi řádky. Beran (2003, s. 10) k tomu uvádí, že v typografické praxi je tento výraz (zvlášť v podání hodnot „jedna“, „jedna a půl“ či „dvě“) zásadně nesprávný a amatérský a doporučuje jej neužívat. Podobně se vyjadřuje i Kočička s Blažkem (2004, s. 27), když odkazuje pojem řádkování k psacímu stroji nebo ke „stařičkému“ editoru Text602. Protože však vývoj nejrozšířenějšího programového vybavení pro zpracování textů v zásadě nerespektuje žádné systematické názvosloví a pojmy v nabídkách jsou většinou tvořeny amatérsky, je také přirozeně použit amatérský pojem řádkování. Tomu se paradoxně přizpůsobuje pak i norma týkající se zpracování dokumentů v textových procesorech, která pojem řádkování uvádí jako jedinou alternativu nastavení řádkového prokladu (ČSN 01 6910, s. 36).

Hrubý (2007, s. 28) se rovněž jednoznačně přiklání k pojmu řádkování jako vzdálenosti dvou po sobě jdoucích účaří, pojem „proklad“ označuje za chybný a uvádí, že se jednalo o plátek vkládaný mezi řádky v kovové sazbě. Tento přístup není v přímém rozporu s přístupy ostatními, ale preference pojmu řádkování není přesně v souladu s typografickými pojmy a je vedena snahou přiblížit používané pojmy nabídkám konkrétních programových systémů.

2.1. Hodnoty řádkového prokladu

U poučených uživatelů, kteří vnímají rozdíl mezi strojopisem a knižním písmem je poměrně rozšířen názor, že řádkový proklad se nastavuje na 20 % stupně písma. Je pravda, že tato hodnota v řadě případů zcela vyhovuje, ale faktorů, které mohou mít na nastavení prokladu vliv, je více.

Kočička a Blažek (2004, s. 25–27) uvádějí, že se proklad v základu nastavuje na dva body, ale v sazbě menším písmem nebo ve verzálkové sazbě je potřeba proklad zvětšit a sazbu tím prosvětlit. Zvětšení prokladu si rovněž vynucuje sazba do dlouhých řádků. Naopak velký proklad působí rušivě u titulků knihy. Beran (2003, s. 10) upřesňuje, že průměrné stupně textového písma mezi 9 a 12 body se prokládají dvěma body, ale menší stupně (5–8 bodů) vyžadují proklad úměrně větší.

Jako další faktor ovlivňující řádkový proklad je zmiňována kresba písma. U písma s tmavším vzhledem nebo větší střední výškou je potřebné proklad zvětšit například o jeden bod (Beran, 2003, s. 11).

V počítačových programech se někdy nachází tzv. „automatický proklad“ – velikost prokladu se automaticky odvozuje procentem od nastaveného stupně písma. Jako předdefinovaná hodnota se nejčastěji vyskytuje 120 % (Kočička a Blažek, 2004, s. 27).

Podle nastaveného prokladu můžeme klasifikovat typy sazby na kompresní (proklad je záporný), kompaktní (proklad je nulový) a volný (proklad je kladný). Kompresní sazba se používá jen ve zcela výjimečných případech, neboť se mohou překrývat dolní dotahy jednoho řádku s horními dotahy a akcenty následujícího řádku. Sazba bez prokladu většinou působí stísněně a je vhodná pouze pro nenáročné tiskoviny (Pop, Flégr a Pop, 1989, s. 67). Tato informace je ovšem svázána s technologií kovové sazby, protože nepoužití proložek k řádkovému prokladu vedlo k úspoře materiálu a práce při sazbě. K tomu například Beran (2003, s. 10) v příkladu textu uvádí, že kompaktní sazba vypadá nepěkně. V době počítačové sazby samozřejmě není spotřeba materiálu již aktuální a kompaktní sazba tím přirozeně pozbývá smyslu.

Proklad se nastavuje v celé tiskovině jednotně. Je nepřípustné změnu řádkového prokladu použít jako nástroj pro rozvolnění nebo natěsnání části dokumentu do vymezeného prostoru (Beran, 2003, s. 12).

Zmíněná norma pro úpravu dokumentů textovými procesory doporučuje v odst. 12.3 volit řádkování „jednoduché nebo mírně zvětšené“. Dále uvádí, že řádkování musí být pro všechny odstavce stejné povahy jednotné a má působit vyrovnaně v celém dokumentu (ČSN 01 6910, 2014, s. 36). Je tedy vidět, že zde zcela chybějí informace o závislosti řádkování na uvedených faktorech, avšak zmíněné řádkování „jednoduché“ je navázáno na předpokládaný použitý textový procesor a jak dále ukážeme, není vůbec jednoduché.

V mnoha zdrojích však nalezneme informace, které jsou v přímém rozporu s uvedenými hodnotami. Řada z těchto zdrojů jsou různá doporučení (snad i závazné pokyny) pro úpravu kvalifikačních vysokoškolských nebo semestrálních prací, různých odborných dokumentů, příspěvků do časopisů a sborníků apod. Podrobnější analýza těchto zdrojů přesahuje možnosti daného příspěvku, můžeme však souhrnně poznamenat, že prakticky všude se lze dočíst, že řádkování má být nastaveno na „jedna a půl“, někdy je doporučeno například „1–1,5 (nejlépe 1,2)“ apod. Texty vycházejí obvykle z doporučení, která platila pro psací stroje a používala se tradičně v dané instituci již desítky let, postupem času byla jen dodána informace o tom, že písmo se má nastavit na Times New Roman, někdy je také zdůrazněno, že má být „asi 60 znaků na řádku a 30 řádků na stránce“. Je přirozené, že svým počtem tyto zdroje silně převažují (o jeden až dva desítkové řády) nad zdroji relevantními, nelze se tedy divit, když i produkované dokumenty obsahují v řádové převaze chybná nastavení řádkového prokladu.

3. Technické řešení řádkového prokladu

Z uvedeného stručného rozboru víme, jak má být řádkový proklad nastaven, proto se nyní budeme zabývat technickým řešením v různých systémech. Jako představitele tří různých skupin programového vybavení pro zpracování textů vybereme systém LaTeX, systém InDesign a systém LibreOffice Writer.

V každém z uvedených systémů je implementován zcela odlišný přístup k formátování dokumentů, s tím je pochopitelně spojena i zcela rozdílná možnost nastavování řádkového prokladu.

3.1. Systém LaTeX

Paradoxně není tento systém z hlediska řádkového prokladu vůbec ničím zajímavý, protože lze nastavit úplně cokoliv, tudíž nám systém neklade žádná omezení, nevnucuje své představy o prokladu a neomezuje pouze na určité přístupy. Řádkový proklad je ovlivněn nejčastěji dvěma nástroji:

- Systémem příkazů pro změnu stupně písma – `\normalsize`, `\large`, `\footnotesize` apod. Se změnou stupně písma se nastaví odpovídající řádkový proklad, který je implicitně zvolen na hodnotu 20 % stupně písma, toto nastavení lze libovolně změnit předchozí úpravou (redefinicí) konstanty `\baselinestretch`, kterou se násobí tento proklad a která má implicitně hodnotu 1.
- Příkazem `\fontsize{stupeň}{proklad}` – druhým parametrem příkazu se nastaví požadovaná vzdálenost účaří dvou po sobě jdoucích řádků textu.

3.2. Systém InDesign

Základním nástrojem na úpravu řádkového prokladu je již zmíněný automatický proklad – lze nastavit, že vzdálenosti dvou po sobě jdoucích účaří budou

odvozeny od zvoleného stupně písma. Kromě toho lze lokálně nebo úpravou parametrů stylu nastavit tuto vzdálenost na hodnotu zadanou absolutním rozměrem. Vazba automatického prokladu na stupeň písma je významným ulehčením při nastavování parametrů sazby a v mnoha případech zcela vyhovuje.

3.3. Systém LibreOffice Writer

Pro nastavení je k dispozici nabídka s názvem „Vlastní hodnota“ s následujícími možnostmi:

- Jednoduché
- 1,5 řádku
- Dvojité
- Proporcionální
- Nejméně
- Proklad
- Přesně

Vzhled této nabídky groteskně dokumentuje bezkoncepčnost tvorby systémů tohoto typu, neboť se zde míchají v prapodivné směsici minimálně tři různé přístupy. Vybrat tu správnou variantu pro daný dokument a daný požadavek vyžaduje skutečně vysokou erudici uživatele, což v praxi vede k jedinému výsledku: uživatel neví, co vlastně volí. A aby toho nebylo dost, v nabídce se objevují kromě volby „Vlastní hodnota“ ještě položky, které umožňují přímou volbu (místo výběru z rozbalovacího seznamu), a tam je uvedeno nikoliv „Jednoduché“, ale „Řádkování 1“, a je tam navíc volba „Řádkování 1,15“. Formulace „Řádkování “ a „Řádkování jednoduché“ je zjevně zmatečná, i když obě volby vedou ke stejnému výsledku.

Je zřejmé, že taková směsice vznikala jako výslednice silných tlaků z více stran. Jedním z nejsilnějších tlaků bylo jistě napodobení nejrozšířenějšího systému patřícího do stejné kategorie – Wordu firmy Microsoft. Tam je ekvivalentní nabídka obsazena následovně:

- Jednoduché
- 1,5 řádku
- Dvojité
- Nejméně
- Přesně
- Násobky

Druhým silným tlakem byla jistě snaha vložit do nabídky možnosti jiných systémů, a obohatit tak repertoár Writeru. Dostala se sem tedy možnost „Proporcionální“ představující určitý odlesk automatického prokladu, ale její funkce je bohužel poněkud odlišná. Třetím tlakem, který v kontextu této nabídky působí dost komicky, je tlak na správnou typografickou terminologii, byl sem tedy vložen „Proklad“, který vlastně ani nijak možnosti nastavení neobohacuje, jen navozuje dojem profesionality uvedením správného typografického pojmu.

Měli bychom nyní objasnit, co se přesně pod jednotlivými pojmy skrývá.

Začneme nejjednodušší skupinou, a tou je bezesporu dvojice voleb „Nejméně“ a „Přesně“. V obou případech se nastavuje řádkový proklad jako vzdálenost účaří,

a to na hodnotu, kterou lze zadat v absolutních jednotkách (bodech). Drobný rozdíl mezi oběma možnostmi spočívá v tom, že u volby „Nejméně“ může dojít k rozšíření zadaného rozměru, pokud se v řádku objeví prvek (písmeno, obrázek apod.), jehož velikost by se do stanoveného rozměru nevešla. U volby „Přesně“ dojde v takovém případě k ořezání příliš velkého prvku shora.

K této skupině lze přidat volbu „Proklad“ – jde rovněž o absolutní nastavení, ale v tomto případě rozměru proložky mezi řádky, tedy nikoliv o vzdálenost účaří.

Zbývá poslední skupina – Jednoduché, 1,5 řádku, Dvojité a Proporcionální, jehož ekvivalentem s mírně odlišným způsobem zadání konstanty jsou wordové Násobky. Jak už bylo ukázáno, jde o skupinu zavedenou původně v microsoftích produktech. A jak také bývá u těchto produktů zvykem, terminologie je dost zavádějící, neboli jednoduché není zdaleka jednoduché a normální vůbec není normální.

Vše je postaveno na volbě „Jednoduché“. Co se touto volbou vlastně nastaví? Kdyby se jednalo o model psacího stroje, šlo by o normalizovanou řádkovou rozteč 4,23 mm, tedy 1/6 palce = 12 monotypových bodů (Slovníček pojmů, 2016). Ale tak tomu bohužel vůbec není, a to i v případě, kdy zvolíme strojopisné písmo.

Zásadní informací v tomto směru je odpověď na přímý dotaz na fóru `ask.libreoffice.org` – proklad je dán vlastnostmi použitého fontu a designér fontu tedy rozhoduje, s jakým prokladem bude implicitně tento font používán. Chceme-li tedy vědět, jak funguje „jednoduché řádkování“, musíme pro použitý font změřit vzdálenosti účaří po sobě jdoucích řádků.

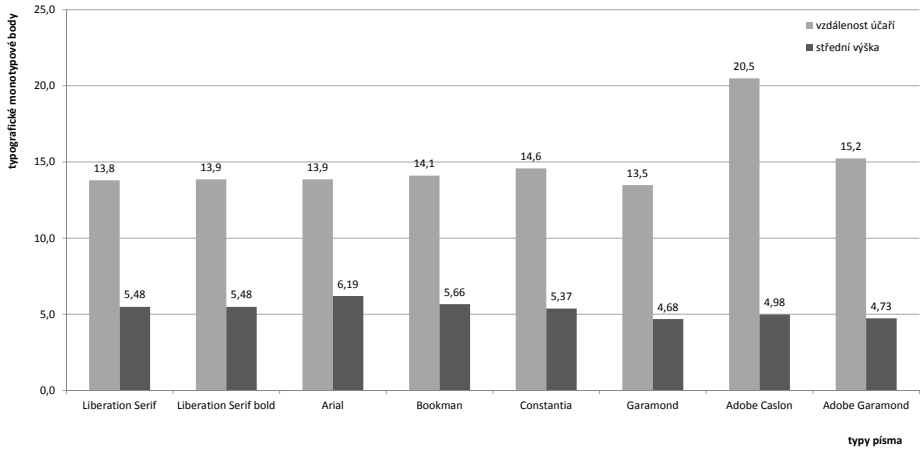
3.4. Stanovení rozměru řádkového prokladu při volbě „Jednoduché“

Pro stanovení přesného rozměru, který vznikne nastavením volbou „Jednoduché“ byl použit obyčejný hladký text, který byl exportován do formátu PDF. Následně byla ve výsledném dokumentu pomocí prohlížeče GSView provedena přesná měření vzdálenosti dvou účaří. Prohlížeč GSView umožňuje snímat pozici kurzoru na obrazovce s přesností 0,01 bodu, při maximálním možném zvětšení textu však nejsou pohyby kurzoru s touto přesností realizovatelné. Výsledky byly tedy zaokrouhlovány na desetinu bodu.

Z měření na několika vzorcích textu vyplývá, že řádkový proklad při nastavení „Jednoduché“ je dán typem písma, ale nesouvisí se střední výškou písma, která byla na každém vzorku rovněž změřena.

Výsledky uvedených několika orientačních měření jsou soustředěny do grafu na obr. 1.

V grafu je i z těch několika málo vzorků jednoznačně vidět, že závislost řádkového prokladu na střední výšce neexistuje, jde tedy o rozměr, který tvůrce fontu nastavuje podle svého uvážení a Writer pouze tento rozměr použije. Stejná situace je i v konkurenčních produktech. Na nastavení „Jednoduché“ navazuje nastavení „1,5 řádku“, „Dvojité“ a ve Wordu „Násobky“. Jde o pouhé násobení základního řádkového prokladu hodnotami 1,5, případně 2 a při nastavení „Proporcionální“



Obrázek 1. Závislost řádkového prokladu a střední výšky na typu písma

(resp. „Násobky“) pak zvolenou konstantou zadanou v procentech nebo jako koeficient.

4. Závěr

Jedním ze zásadních parametrů sazby dokumentu je bezesporu řádkový proklad. Podíváme-li se na možnosti, které nabízejí běžně dostupné procesory pro zpracování textů, bylo ukázáno, že ve značné části nastavení se pracuje s hodnotami, o nichž nemá uživatel žádné solidní informace, nastavuje tedy zcela neznámé hodnoty. S tím souvisí i řada zjevně nesprávných doporučení, která vyplývají z naprostého nepochopení základních principů zpracování textů v textových procesorech při použití proporcionálního písma.

Shrneme-li s krátkými komentáři možnosti nastavování řádkového prokladu, dostáváme následující přehled:

- Volba „Jednoduché“ – uživatel nastavuje neznámý rozměr, bere se informace z fontové metriky, která je uživateli nedostupná. Uživatel se v tomto případě spoléhá na návrháře fontu, který tímto způsobem určuje minimální použitelný řádkový proklad pro daný font. Nezanedbatelným dalším nebezpečím této volby je, že při změně fontu v dokumentu se může zásadně změnit tiskový rozsah, neboť změna řádkového prokladu má na rozsah značný vliv.
- Volba „1,5 řádku“ – opět jde o nastavení neznámého rozměru, neboť jde o 1,5 násobek hodnoty „Jednoduché“, avšak ve většině doporučení je tato hodnota mylně považována za řádkování strojopisné s hodnotou $1{,}5 \cdot 4{,}23$ mm. Podobně je na tom volba „Dvojité“, jejíž četnost v různých

doporučeních je však zanedbatelná a ze zkušeností je známo, že tuto volbu využívá nepatrná část uživatelů.

- Volba „Násobky“ (MS Word) – uživatel opět nastavuje neznámý rozměr, motivace k této volbě je však tvořena v podstatě dvěma proudy:
  a) uživatelé mylně předpokládají, že se jedná o kýžený násobek stupně písma, nastavují jej tedy na 1,2 (to platí o poučenějších uživatelích, kteří alespoň přibližně tuší, jak má vypadat řádkový proklad v proporcionálním písmu);
  b) hodnota „Násobky“ s konstantou 1,15 je přednastavena jako implicitní v posledních verzích Wordu – méně poučení uživatelé ji pokládají za optimální typograficky správnou hodnotu. Tento stav reflektuje volba „Řádkování 1,15“ ve Writeru.
- Volba „Nejméně“ – uživatel nastavuje známý přesný rozměr, vzdává se závislosti na kresbě a typu písma, případně nastavuje hodnotu v závislosti na celkovém designu dokumentu. V tomto případě lze nastavit jakoukoliv, tedy i zcela správnou hodnotu. Podobně se chová volba „Přesně“ s drobným rozdílem uvedeným již v úvodní části textu. Obdobou těchto voleb je „Proklad“ s možností zadání proložky mezi řádky.

Cílem příspěvku bylo ozřejmit vliv řádkového prokladu na celkovou úpravu dokumentu a vytvořit vazbu na volby dostupné v textových procesorech. Lze tedy konstatovat, že uživatel by měl aplikovat typografická pravidla volbami „Nejméně“, „Přesně“ nebo „Proklad“ a měl by také vědět, jak pracují další volby a jaká nebezpečí mohou ukrývat.

Reference

[1] BERAN, V. a kol.: *Aktualizovaný typografický manuál, Část 2: Sazba*, Praha, Kafka design, 2003.

[2] *ČSN 01 6910 – Úprava dokumentů psaných textovými procesory*, Praha, ÚNMZ, 2014.

[3] KOČIČKA, P. – BLAŽEK, F.: *Praktická typografie*, Brno, Computer Press, 2004, ISBN 80-722-6385-4.

[4] POP, P. – FLÉGER, J. – POP, V.: *Sazba I. Ruční sazba*, Praha, Státní pedagogické nakladatelství, 1989.

[5] Východočeské archivy: *Slovníček pojmů – písmeno O, Heslo „Okraje dokumentů a řádkové rozteče“*, `http://vychodoceskearchivy.cz/ustinadorlici/slovnicek-pojmu/o/` [cit. 2. 6. 2016].

Kontaktní adresa

doc. Ing. Jiří Rybička, Dr., Ústav informatiky, Provozně ekonomická fakulta, Mendelova univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika,
E-mailová adresa: `rybicka@mendelu.cz`, `http://akela.mendelu.cz/~rybicka`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 125–132

UŽITÍ TEXU A LUA PŘI SAZBĚ KNIHY PRŮVODCE SVĚTEM ARDUINA

PAVEL STŘÍŽ (CZ)

Abstrakt. Během sazby knihy *Průvodce světem Arduina* (viz `http://www.striz.cz/90arduino.php`) od Zbyška Vody a HWKitchen (Oldřich Horáček) jsem narazil na zajímavý problém. Během dokončení knihy jsem potřeboval extrahovat zdrojové kódy, zasáhnout do nich (vyházet diakritické znaky, nabízela se možnost přidání jednotného záhlaví dle typu souboru) a nadělat z nich individuální textové soubory, které v názvu souboru obsahují číslo strany knihy. Tyto soubory se pak šíří mimo tištěnou knihu. Navíc jsem potřeboval ošetřit situaci, kdy začíná na jedné straně knihy víc zdrojových kódů, tedy to chtělo nějaký čítač zdrojových kódů nebo číslo strany s dodatečným čítačem. Řešení jsem potřeboval univerzální, aby se při sazebním zásahu v knize všechny zdrojové kódy znovu extrahovaly a byly vždy aktuální. Představím své řešení a knihu nechám kolovat mezi účastníky konference s možností si ji na místě zakoupit.

Klíčová slova. Arduino, TEX Live, TEX, Lua, listings.

USING TEX AND LUA IN TYPESETTING BOOK THE GUIDE IN THE WORLD OF ARDUINO

Abstract. The article introduces problems which the typesetter encountered while typesetting book *The Guide in the World of Arduino* by Zbyšek Voda and HWKitchen (Oldřich Horáček) `http://www.striz.cz/90arduino.php`. The author presents his solutions, tips and tricks. The most interesting problem was an extraction of the source codes mentioned in the book. The filenames needed to include page number where source code starts. A unique counter was necessary because source codes can start on the same page. This was a fine place where the author used Lua. My brother and I have finished book under tremendous time pressure (just before Christmas 2015), but we accomplished that on time. Book was available for seeing and buying at the conference venue. There was 200 copies in the first printing and 1000 copies in the second printing.

Keywords. Arduino, TEX Live, TEX, Lua, listings.

TEXista kafrá do Open Hardware místo Úvodu

V červenci 2015 jsem se poprvé dozvěděl o Arduinu a neměl jsem s ním zkušenosti ani s Raspberry Pi. Měl jsem v hlavě divadelní projekt (viz samostatný článek v tomto sborníku) a velkou chuť si to „něco“ připravit. Proto jsem začal aktivně studovat, nakupovat součástky a experimentovat.

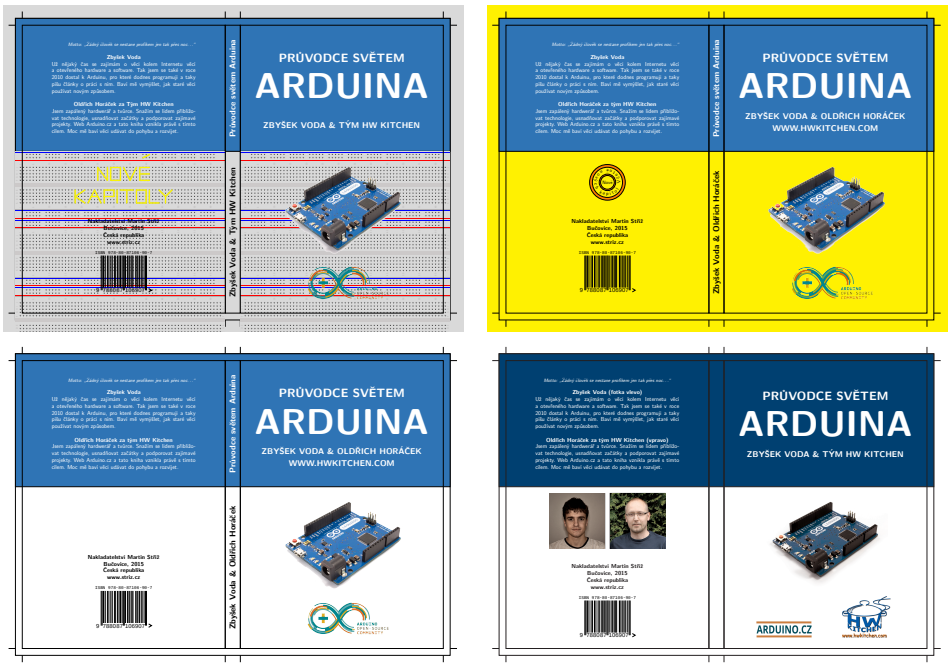
Hlavně první měsíce testů byly intenzivní a dostal jsem se do kontaktu s Oldřichem Horáčkem z HWKitchen. Kde mohl, poradil (jak začít, jaká relátka zvažovat ap.). Prozradil jsem na sebe, že jsem typograf a bratr má nakladatelství. On zas prozradil, že by rád dokončil papírovou knihu, kterou vydávali jako seriál na webových stránkách a jako PDF pro zájemce ke stažení zadarmo.

To však vyžadovalo sazební zásahy, čištění a sjednocení zdrojových kódů (víc autorů, články psané v různých obdobích). A najednou byl projekt na mém ponku.

1. Sazba mimo blok knihy: obálka

Ačkoliv jsem byl pod zajímavým časovým tlakem, zkusil jsem si několik verzí obálky. Představuji dvě ukázky nevyužitých nápadů a barevnou variantu přesázenou a dokončenou dle návrhu autorů (představuji rozkresy, se kterými se čtenář knihy normálně nesetká). Na tento úkol téměř výhradně používám Ti*k*Z.

Bylo potřeba převést barevné obrázky do stupňů šedi a zajistit, že jsou v barevném prostoru CMYK. Užil jsem GraphicsMagick. Na generování několika variant jsem použil proměnné (příkaz `echo`) a několik běhů TEXu (načítání generovaného souboru programem `bash` přes `\input`).



Obrázek 1. Dva nápady: užít Fritzing a jinou barevnost s razítkem, ve spodním řádku je původní rozkres dle návrhu autorů (vlevo) a dokončená obálka po dvou či třech iteracích

2. Zásahy před sazbou knihy: čištění

Letos se na konferenci zadařilo pokračovat v představování LuaTEXu (Pavel Tišnovský, včetně dvou workshopů), resp. ConTEXtu (Tomáš Hála, zvaná přednáška). Máme z toho velkou radost. Velice efektivní je používat Lua přímo za běhu TEXu, nemusíte věci řešit před či po běhu TEXu, je to jiný svět. V dřívějším zaměstnání jsem si dokonce ověřoval věci z webu za běhu TEXu (`\write18` a `wget`). Poněvadž jsem zdrojové kódy „vracel“ zpátky autorům, kteří umí s pdfLATEXem, tak jsem Lua používal hlavně mimo TEX.

Masivně používám Lua na čištění a zanášení korektur u datových, textových, obecně TEXových souborů. O tom snad víc v budoucnu. Používám na to regulární výrazy (příkaz Lua `unicode.utf8.gsub`) s tahem dostudovat knihovnu `LPeg`.

Představím dva momenty z čištění před TEXováním. Ve zdrojových kódech (prostředí `lstlisting`) jsme se domluvili, že nebude diakritika, někde byla, někde ne. Nepodařilo se mi znovu dohledat univerzální řešení pro všechny diakritické znaky (Unicode, asi to bylo v Perlu), udělal jsem to pro české diakritické znaky.

Druhým momentem bylo aplikování nedělitelných mezer za jednoznakové předložky (já s oblibou nastavuji i spojky). Na to se hojně používá program `vlna` od Petra Olšáka, ale musel jsem zajistit, aby se tentokrát do zdrojových kódů nezasáhlo. To je TEXové prostředí, nikoliv však `verbatim`, které by program `vlna` ještě uměl vynechat (parametr `-n`).

2.1. Miniukázky

Začněme jednoduchou ukázkou, kdy chceme desetinné tečky převést na desetinné čárky. Princip je jednoduchý, jak však Lua není nejužívanějším jazykem, je dobré na to poukázat. Procento a tečka tvoří operátor (tečka doslova), jinak tečka znamená libovolný znak.

U druhé ukázky využijeme anonymní funkci a znaky otočíme, co byla tečka bude čárka a obráceně. V parametru `s` (používám to jako zkratku slova String) máme uložený nalezený znak a na jeho místo vracíme něco nového.

U třetí ukázky pracujeme s vícebajtovou sekvencí (diakritické znaky), to už s příkazem `string.gsub` nevystačíme. Můžeme pracovat s vestavěným příkazem `unicode.utf8.gsub`. Ukažme si to na počtu znaků.

```
textA=[[123.45 24.78 81.36]]
textA=string.gsub(textA, "%.", ",")
print(textA) -- 123,45 24,78 81,36

textB=[[47,54 61.7 4,5 9.8 10,4]]
textB=string.gsub(textB, "[%.,]", function(s)
  if s=="." then return "," else return "." end
  end)
print(textB)-- 47.54 61,7 4.5 9,8 10.4
```

```
textC=[[Pavel Stříž šel na oběd.]]
print(string.len(textC))       -- 29, chybné (je to počet bajtů)
print(unicode.utf8.len(textC)) -- 24, správné (je to počet znaků)
```

Pokud máme následující řádky uloženy pod `zamen.lua`, spustíme si (autor z TEX Live 2016): `texlua zamen.lua`.

Podívejme se na dvě náročnější ukázky z čištění bloku knihy.

2.2. Náhrada diakritických znaků ve zdrojových kódech

Náhradu diakritických znaků si dáme do cyklu. Už musíme jen vyhledat bloky zdrojových kódů, v knize se používal balíček listings a prostředí `lstlisting`. Ponevadž není pravděpodobné, že by se prostředí vnořovala, lze použít sekvenci „`.-`“: najdi nejbližší.

Pro zájemce kolem úpravy textů hledejte více ukázek např. v balíčku chickenize.

```
-- Zadej texty.
obsah=[[Text předem.
\begin{lstlisting}
Text uprostřed kódu.
\end{lstlisting}
Text za zdrojovým kódem.]]
celkem=0 -- Počty měněných znaků.
diafrom, diato=
  "áéíóúýčďěňřšťžůÁÉÍÓÚÝČĎĚŇŘŠŤŽŮ",
  "aeiouycdenrstzuAEIOUYCDENRSTZU"

-- Zpracuj texty.
obsah=unicode.utf8.gsub(obsah,
"\\begin{lstlisting}.-\\end{lstlisting}",
   function(s)
   for poradi=1,unicode.utf8.len(diafrom) do
     znakfrom=unicode.utf8.sub(diafrom,poradi,poradi)
     znakto=unicode.utf8.sub(diato,poradi,poradi)
     s,kolik=unicode.utf8.gsub(s,znakfrom,znakto)
     if kolik>0 then celkem=celkem+kolik end
     end -- for
   return s
   end -- function)

-- Vypiš výsledek.
print("Celkem měněných diakritických znaků: "..celkem)
-- Dostáváme: Celkem měněných diakritických znaků: 2
```

2.3. Rychlá vlna přes Lua

Zajímavější situace je, když chceme inverzi, tedy mimo zdrojové kódy. Nevýhoda operátoru `%b` je, že pracuje jen s jedním znakem. Místo, abych přešel na `LPeg`, tak jsem si začátek prostředí nahradil za znak `\002` a konec prostředí za znak `\003`. Po úpravách si nastavím zpět. Dává mi to možnost párového vyhledávání i cyklu přes různá prostředí. U inverze už stačí si přidat trojku na začátek a dvojku na konec řetězce. Vyhledávat `\002` až `\003` je prostředí, `\003` až `\002` jsou bloky mimo prostředí. Podobný způsob se dá využít na komplikovanější vyhledávání: uvnitř a mimo matematiku ap.

Za pozornost u tohoto problému stojí nejen program `vlna` (Petr Olšák), ale i `encxvlna` (Zdeněk Wagner, Petr Olšák) a `luavlna` (Michal Hoftich).

```
-- Zadej upravovaný text
obsah=[[Text v přední části s~i bez nedělitelné mezery.
\begin{lstlisting}
Zde jsem v zóně.
Nic neměnit!
\end{lstlisting}
Text pokračuje v této části.]]

-- Tam
obsah=unicode.utf8.gsub(obsah, "\\begin{lstlisting}", "\002")
obsah=unicode.utf8.gsub(obsah, "\\end{lstlisting}", "\003")
obsah="\003"..obsah.."\002"

-- Zásahy
obsah=unicode.utf8.gsub(obsah, "%b\003\002",
  function(s)
    s=unicode.utf8.gsub(s,"([%s~][KkSsVvZzOoUuAaIi])%s","%1~")
    -- Zde může být řada dalších zásahů a čištění.
  return s
  end)

-- Zpět
obsah=unicode.utf8.sub(obsah, 2, -2) -- Umaž pomocný první a
poslední znak.
obsah=unicode.utf8.gsub(obsah, "\002", "\\begin{lstlisting}")
obsah=unicode.utf8.gsub(obsah, "\003", "\\end{lstlisting}")
-- Vypiš výsledek
print(obsah)
--[[Text v~přední části s~i~bez nedělitelné mezery.
\\begin{lstlisting}
```

```
Zde jsem v zóně.
Nic neměnit!
\end{lstlisting}
Text pokračuje v~této části.--]]
```

3. Zásahy po sazbě knihy: extrakce zdrojových kódů

Existuje několik TEXových balíčků a pomůcek za běhu Lua. TEXové balíčky jsem nevyužil, protože jsem potřeboval do souborů zasahovat a do poslední chvíle nebylo jisté, jestli se bude přidávat komentář, tzv. hlavička souboru (v Arduinu `/* ... */`), případně kontrolní znaky či velikost souboru na konci zdrojového kódu. Lua za běhu TEXu nešlo použít, neb jsem chtěl zůstat v pdfLATEXu. Kdyby nastal problém či si autoři potřebovali zasáhnout, měli by možnost.

Postup byl takový, že TEX si u každého zdrojového kódu zapisoval do `.aux` souboru informace jako u každého jiného křížového odkazu. Využil jsem vlastní značku (`lst-pajovo`), ale šel by užít extra soubor.

Po dokončení TEXování si Lua vytáhl informace z tohoto `.aux` souboru, vytáhl si zdrojové kódy z TEXového souboru (s bratrem obvykle pracujeme s jedním TEXovým souborem, usnadňuje to vyhledávání a globální náhrady v celé knize), extrahoval je do souboru, který se jistým klíčem pojmenoval.

V preambuli dokumentu za `\usepackage{listings}` jsem dopsal:

```
\let\oldbeginlstlisting=\lstlisting
\newcount\pajovo
\pajovo=0
\def\lstlisting{%
   \global\advance\pajovo by 1%
   \label{lst-pajovo-\the\pajovo}%
   \oldbeginlstlisting}
```

V `.aux` souboru jsem snadno dohledával informace typu:

```
\newlabel{lst-pajovo-1}{{6.4}{24}{Programovací jazyk}{section.6.4}{}}
\newlabel{lst-pajovo-2}{{6.4}{24}{Programovací
jazyk}{lstnumber.-1.9}{}}
\newlabel{lst-pajovo-3}{{6.4}{25}{Programovací
jazyk}{lstnumber.-2.3}{}}
[... zkráceno ...]
```

Řídicí soubor `vytahni.lua` si vytáhl potřebné a připravil extrakci zdrojových kódů s příslušnými názvy obsahující číslo strany knihy, kde zdrojový kód začíná.

Za pozornost stojí volání si příkazu `unicode.utf8.gsub` v anonymní funkci uvnitř `unicode.utf8.gsub` (hledám část ve vyhledané části). Také si výsledek neukládám do proměnné `tex`, nepotřeboval jsem jej.

```
-- Vstupní parametry
    zdrojaux="preambule.aux"
    zdrojtex="arduino-blok.tex"
    adresar="!zdrojaky/"

    -- Otevření souborů a načtení obsahů
    soubor=io.open(zdrojaux,"r")
    aux=soubor:read("*all")
    soubor:close()

    soubor=io.open(zdrojtex,"r")
    tex=soubor:read("*all")
    soubor:close()

    -- Načtení údajů a zpracování
    jadro={}
    unicode.utf8.gsub(aux, "\\newlabel{lst%-pajovo%-(%d+)}(%b{})",
       function(citac,strana)
       strana=unicode.utf8.sub(strana,2,-2)
       strana=unicode.utf8.gsub(strana,"%b{}(%b{}).+","%1")
       strana=unicode.utf8.sub(strana,2,-2)
       print(citac,strana)
       jadro[tonumber(citac)]=strana
       end)

    -- Úpravy a zápis zdrojových kódů do souborů
    citac=0
    unicode.utf8.gsub(tex, "\\begin{lstlisting}(.-)\\end{lstlisting}",
       function(s)
       s=unicode.utf8.gsub(s, "^%s-%b[]", "") -- vol. parametr pryč
       s=unicode.utf8.gsub(s, "^%s+", "") -- mezery začátek řádků
       s=unicode.utf8.gsub(s, "%s+$", "") -- mezery konec řádků
       citac=citac+1
       fstrana=unicode.utf8.format("%03d",jadro[citac])
       fcitac=unicode.utf8.format("%03d",citac)
       print("Zpracovávám kód číslo "..citac.."...")

    kam=io.open(adresar.."strana-"..fstrana.."-poradi-"..fcitac..".txt",
    "w")
       kam:write(s)
       kam:close()
       end)
```

U této konkrétní knihy pak v předpřipraveném adresáři vznikly TXT soubory s přibližně těmito názvy:

```
strana-024-poradi-001.txt
strana-024-poradi-002.txt
strana-025-poradi-003.txt
[... zkráceno ...]
```

Promazat si adresář jsem dělal na úrovni operačního systému v dávkovém souboru či se nabízí `Makefile`.

Název obsahuje stranu v knize (můžete si sami ověřit, až budete knihu držet v ruce) a celkové pořadí kódu v knize. Poznámky v úvodu či na závěr kódu se nedávaly. Poněvadž se kódy dají vykopírovat z PDF, tak se autoři rozhodli tyto soubory zatím nevyužít. Připravené to však je, dodávám s humorem.

4. Kdy mi práce končí: předávka knihy

Má práce skončila, když jsem předal vše (upravené a ořezané obrázky, zdrojové kódy obálky a bloku, vyextrahované zdrojové kódy) a soubor přijala tiskárna. Použil jsem program `rar`. Vyžádali si jen jedno vylepšení u obálky, a to rozšíření okrajů, aby neměli problém s tiskem na spád (do okrajů). To už byla typografická operativa do hodiny.

Byla to fuška, na konferenci jsem však mohl knihu představit s úsměvem na tváři. První tisk byl u velké tiskárny nastaven na 200 kusů, druhý na 1000 kusů.

Několik myšlenek místo Závěru

Musím uznat, že tento knižní projekt znamenal větší část mých příjmů za rok 2015. Proč ne? Ovšem utratil jsem je dokázal téměř obratem za elektrosoučástky a příslušenství k Arduinu a Raspberry Pi.

V domácích úkolech mám už roky dostudovat Lua knihovnu `LPeg`, která je ještě obecnější a mnohem silnější na regulární výrazy, ale u produkce jsem s příkazem `unicode.utf8.gsub` (plus jejich případným vnořováním) zatím vystačil.

Tak si s úsměvem říkám, jestli by to už nechtělo zase něco nového, stále se vyvíjející se LuaTEX je nuda, no ne? Že by LaTeX3 či ConTeXt?

Kontaktní adresa

Ing. Pavel Stříž, Ph.D., Nakladatelství Martin Stříž, U Škol 940, Bučovice, okres Vyškov, CZ-685 01, Česká republika,

E-mailová adresa: `pavel@striz.cz`

8. ročník Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 133–142

ARDUINO MI POMÁHÁ NA DIVADLE: VSTUPUJEME DO SVĚTA SVĚTELNÉHO DESIGNU

PAVEL STŘÍŽ (CZ)

Abstrakt. V článku navazuji na zkušenosti z dřívějších let kolem technické pomoci u divadelních ochotníků v Bučovicích (Česká republika). Zaměřuji se na začlenění Arduina do světelného designu, ale nahlédnu na dřívější problém spuštění dávkových souborů z PDF v Linuxu i se pokusím popsat současné plány.

Klíčová slova. Arduino, divadlo, ochotníci, světelný design, ovládání elektrospotřebičů, relé, SRD, SSR, `bash`, `stty`, TEX, `hyperref`, `Foxit Reader`, `xpdf`.

ARDUINO HELPS ME IN THE THEATRE PRODUCTION: ENTERING THE WORLD OF LIGHT DESIGN

Abstract. The article summarizes my new experience as a technician in the local theatre in Bučovice (Czech Republic). It focuses on implementing Arduino to the light design, but I also look back to the previous problem of running bash scripts from the PDF file under Linux as well as I am describing my current plans.

Keywords. Arduino, theater production, amateurs, light design, handling electrical devices, relay board, SRD, SSR, `bash`, `stty`, TEX, `hyperref`, `Foxit Reader`, `xpdf`.

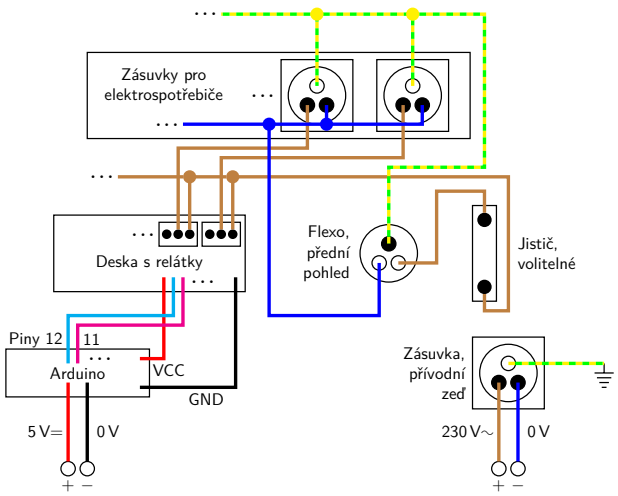
↯ Upozornění autora ↯

Během bastlení v tomto článku se pracuje s malým, ale hlavně s otevřeným nízkým napětím, které může být smrtelné (230 V∼). Pokud nemáte příslušné vzdělání či zkušeného odborníka k ruce, bastlení se raději vyhněte! Pokud vám tento článek nedává spát, nezapojujte potenciálně smrtelné napětí (přívodní kábl ke spotřebičům do sítě), i tak můžete sledovat LEDky u relátek, kde je maximálně 5 V=. Pokud si musíte zapojit vše, nechte si zapojení alespoň zkontrolovat, protože pustit si omylem 230 V∼ do notebooku jej může naprosto zlikvidovat. Nezbytné varování vyřčeno, pojďme na to.

Představení problému

S Arduino světem jsem se setkal na konferenci OSSConf v roce 2015 při neformální diskuzi. Poradil mi to Remigiusz Olejnik, díky mu. Měl jsem v hlavě „něco“ co by mi ovládalo světla přes notebook, ale měl jsem jen mlhavou představu o Raspberry Pi. Jinak totiž musím vše ovládat přes vypínače, to je únavné a při velkém množství spotřebičů nerealizovatelné. Nakoupit DMX mixážní pult by asi šlo, ale nejlevnější stojí cca 18 tisíc korun (cca 650 eur) a není zajištěna

ovladatelnost přes notebook, ale ručně přes tzv. šavle (posuvníky). Ty dražší mají možnost paměti a připojení na notebook přes převodník (obousměrný USB a DMX). Má tehdejší myšlenka byla, že bych si přál ovládat světla (obecně elektrické spotřebiče) z PDF tak, jako to mám u hudby. Postupně jsem se seznamoval s možnostmi Arduina, nakupoval součástky a začal bastlit. Měl jsem jasný cíl, a to u takových (školských) experimentů pomáhá.



Obrázek 1. Naznačení schématu zapojení

1. Strana hardware

Jakmile jsem ovládal LEDky, tak už bylo jen otázkou času, kdy přejdu na vyšší napětí. Ze svých technologických studií jsem si matně vzpomněl na relátka a bádal jsem dál za pomoci elektrikářů. Laicky řečeno, relé je přepínač. Dostane-li na jedné straně dost voltů, zapne či vypne druhou stranu.

Došel jsem k HWKitchen a tam mi Oldřich Horáček poradil desky s relátky (zde je osmikanálové). Dají se pořídit jedno-, dvou-, čtyř-, osmi-, ale i šestnáctikanálové. To jsou ty elektromagnetické (SRD, SRD-05VDC-SL-C). Udrží to spotřebiče do 240 W (typ C), resp. 300 W (typ A). Oblíbená zkratka NC (Normally Closed), znamená, že původní stav relátka na vyšší straně napětí je obvod uzavřený, NO (Normally Opened) pak otevřený.

Později jsem narazil na polovodičové (jsou tiché, méně se zahřívají) a ty jsem si pořídil přes AliExpress (SSR, Solid State Relay, Omron G3MB-202F). Udrží 2 A, ale nemají NC a NO. Základní stav je vypnutý a spíná se.

Nevýhodou obou desek je, že neobsahují optočleny, které by zajistily ještě vyšší bezpečnost vašemu Arduinu (fyzické oddělení stran napětí). Taky je potřeba desky dávat na nevodivou plochu.

Existují relátka, která udrží mnoho desítek ampér, ty by se možná hodily na velké reflektory, ale ty zatím nezvažuji připojovat. Ty, které na divadle máme z 60. let (1000 i víc wattů), půjdou do šrotu, neb nedávají adekvátní svit.

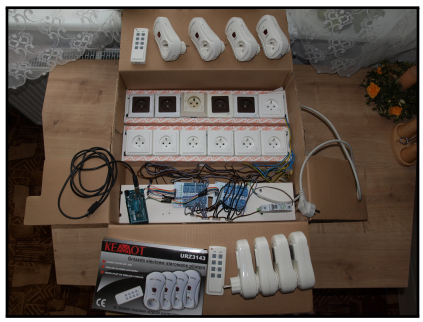
Pracuji s Arduino Uno či Arduino Mega, dle počtu spotřebičů (to závisí na studované inscenaci). Přivádím 230 V∼ přes přívodní kábl (ze starých káblů k počítači jsem si udělal flexa odstřihnutím koncovky), které rozvádím do zásuvek a fázi ještě do relátek. To má výhodu, že nemusím do přívodních káblů u spotřebičů zasahovat či se nějak vrtat ve vidlicích. Spotřebič zůstává beze změny.

Z WAGO spojek/svorkovnic to mám vedené do relátek a dál do zásuvek. U některých YouTube videí si napojují relátka přímo mezi sebou, ale do toho jsem nešel. Naopak nulák a zem mezi zásuvkami napojené mám.

Považoval jsem za povinnost u svého prvního projektu připojit jistič, použil jsem Noark Ex9BN 1P B25. Funguje dobře, ale když se mi podařil zkrat, nezachytil vše. Tohle musím domyslet, aby mi výboj nešel mimo pokoj. Do (vrchem) a z jističe (spodem) mi jde fáze.

Zvláštní část mých experimentů mi tvořily dálkově ovládané zásuvky (RF 433,92 MHz), o tom snad jindy. Vyhovoval mi polský výrobce Kemot s produktem URZ3143 se statickým kódováním na ovladači. To má výhodu, že nedochází k fyzickému styku s Arduinem, udrží 2300 či až 3600 W u jiných výrobců, ale ne se všemi ovladači se dá rozumně domluvit a musí být zajištěno čisté RF vysílání i přijímání. Zůstal jsem u káblíků, protože něco vést do výšky dvou metrů znamená, že mi tam některý z herců projde, vést něco nad hlavami herců znamená, že to musím dobře uchytit, aby se nic nepohnulo, nedejbůh nespadlo.

Na obrázku 1 je schéma pro spotřebiče, na fotce (obrázek 2) můžete vidět zapojení pro 12 spotřebičů (tzv. instantní krabicové řešení). Pokud nepoužijeme digitální pin 13 (problikává na začátku, díky za tip Slavko Fedorikovi), tak nám stačí Arduino Uno a digitální piny. Analogové piny jsem nezkoušel. Jiný typ desky by neměl být problém. Raspberry Pi 2, model B, s 3,3 V na digitálních pinech by mělo být dostatečné, viz toto YouTube video. Napočítal jsem jich 17 dostupných. Projekt v akci, resp. otevřenou krabici, bylo možné vidět na OSSConf 2016.



Obrázek 2. Řešení se 12 zásuvkami a 8 dálkově ovládanými zásuvkami (RF, firma Kemot)

2. Strana Arduina

Z Arduina se stává prostředník mezi notebookem a spotřebiči. Přijímá komunikaci přes sériovou linku, každý bajt/příkaz má svůj účel (má konvence: velká písmena zapínají spotřebiče, A je první spotřebič, B druhý atd.; malá vypínají), v mém případě jen zapnutí či vypnutí spotřebiče. U prvních testů jsem používal i cifry, ale těch je jen 10, spotřebičů mohu mít víc. Vícebajtové sekvence lze uvážit, ale snažil jsem se tomu vyhnout. Nebyl velký problém si zásuvky tužkou popsat. Testy funkčnosti kódu mohu dělat v Arduino IDE/Studiu pomocí monitoru (`Tools` → `Serial Monitor`), znaky mohu zasílat z tohoto místa. V produkci sekvenci píši v TEXovém souboru. Malou prodlevu mezi spotřebiči obvykle nastavuji na úrovni kódu Arduina, aby se spotřebiče nevypínaly či nezapínaly v jednom okamžiku.

```
// Abych nemusel kroutit káblíky, mám to otočené zde.
// Digitální pin 13 spíš nepoužívám, podobně se snažím šetřit 0 a 1.
int jadro[] = {0, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2};
int citac; // cyklus přes digitální piny
int velikost = sizeof(jadro) / sizeof(int); // počet pinů
int h; // načtený znak přes sériovou linku

void setup() {
  Serial.begin(9600);
  for (citac = 1; citac < velikost; citac = citac + 1) {
    pinMode(jadro[citac], OUTPUT);
  } // for()
} // setup()

void loop() {
  if (Serial.available() > 0) { // Dorazil příkaz/bajt?
    h = int(char(Serial.read())); // Ano, ukládám si.
    if (h < 'a') { // Je to velké písmeno?
      digitalWrite(jadro[h - 'A' + 1], HIGH); // Ano, zapni spotřebič!
    } else { // Ne, není, je to malé písmeno.
      digitalWrite(jadro[h - 'a' + 1], LOW); // Není, vypínám
 spotřebič!
    } // if(), zapni/vypni spotřebič
    delay(50);// Chvíli přibrzdi, nezapínej/nevypínej vše co
 nejrychleji.
  } // if(), příkaz
} // loop()
```

Zde představuji další funkční možnost, kdy jsem přes Arduino ovládal 4 dálkově ovládané zásuvky od polské firmy Kemot, produkt URZ3143. Jedním z mých pracovních cílů bylo spojit kódy, tedy ovládat 12 elektrospotřebičů přes káblíky a 4 či 8 elektrospotřebičů přes tyto zásuvky. Používám 433MHz RF link kit. Jak vidíte, spojení bude operativa na úrovni bastlení a programování.

```
#include <RCSwitch.h> // Užívaná knihovna.
RCSwitch mySwitch = RCSwitch();
int pocet = 3; // Vlastní počet pokusů.
String h; // Načítaný bajt ze sériové linky.
void mal(char* malsend) { // Příkaz odesílající znaky.
  for (int malcount = 0; malcount < pocet; malcount++) { // Počet
 pokusů.
    mySwitch.sendTriState(malsend); // Zašli sekvenci na zásuvky.
    delay(200); // Přestávka mezi pokusy. }
  delay(100); // Přestávka mezi příkazy. }

void setup() {
  Serial.begin(9600); // Zahájení komunikace.
  mySwitch.enableTransmit(2); // Aktuální nastavení na zásuvky.
  mySwitch.setPulseLength(164); // Délka vln přenosu, experimentálně
 nastaveno.
  mySwitch.setProtocol(1); // Sada zásuvek, nastavení dle příjmu v
 RCSwitch.
  //mySwitch.setRepeatTransmit(1); // Počet pokusů si nastavuji sám. }

void loop() {
  if (Serial.available() > 0) {
    h = String(char(Serial.read()));
    Serial.println(h); // Kontrolní výpis na terminál.
    // Zapnutí zásuvek, mám jich max. 8, zvolil jsem cifry
    if (h == "1") {mal("FF00F0FFFF01");}
    if (h == "2") {mal("FF00FF0FFF01");}
    if (h == "3") {mal("FF00FFF0FF01");}
    if (h == "4") {mal("FF00FFFF0F01");}
    // Vypnutí zásuvek, čísla s klávesou Shift na anglické klávesnici.
    if (h == "!") {mal("FF00F0FFFF10");}
    if (h == "@") {mal("FF00FF0FFF10");}
    if (h == "#") {mal("FF00FFF0FF10");}
    if (h == "$") {mal("FF00FFFF0F10");}
  } // Serial.available()
} // loop()
```

3. Odeslání dat

V produkci to mám tak, že před otevřením PDF se nahraje HEX soubor do Arduina (ten jsem si vytáhl z dočasných souborů, cesty jsou vidět při zapnutí logu v Arduino IDE). V Arduino IDE: `File` → `Preferences`, `Show verbose output during:`, zaškrtávám si obojí `compilation` a `upload`. Pracuji z PDF a po uzavření PDF se načte minimální HEX soubor do Arduina (vyčištění paměti mikrokontroleru na další práci), tedy:

```
void setup() {}
void loop() {}
```

V pozadí se po nakliknutí hypertextového odkazu spouští dávkové soubory, v naší ukázce jsou generované TEXem, ale mohou být předpřipravené a vedle světel mohou zasahovat i do hudby a zvuků.

Název se dá zjistit rychle při srovnání výpisů `/dev` před a po zapojení Arduina. Například přes program `diff` nebo `colordiff`:

```
$ ls /dev >~/mount-before.txt
$ # zapojíme Arduino
$ ls /dev >~/mount-after.txt
$ diff ~/mount-before.txt ~/mount-after.txt
138a140 > ttyS5
```

Po aktivaci (Linux či CygWin pod Microsoft Windows) komunikace jsem mohl přejít na testy, na mém stroji mám Arduino na `ttyS5`, resp. `COM6`:

```
$ stty -F /dev/ttyS5 9600 cs8 cread clocal
```

Pod Microsoft Windows zašlu sérii bajtů přes dva kroky:

```
echo ABCDE >xfile.txt
print /D:COM6 xfile.txt
```

Pod CygWinem/Linuxem jedním krokem:

```
$ echo ABCDE >/dev/ttyS5
```

Pro zájemce o sériovou komunikaci mohu doporučit tuto stránku s diskuzí.

V produkci před a po tomto bloku nahrávám do Arduina předpřipravený HEX soubor a čistím jej dalším HEX souborem (Microsoft Windows), např.:

```
@rem Nahrání souboru do Arduina
"C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe"
 -C"C:\Program Files
 (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -v -patmega2560
 -cwiring -PCOM4 -b115200 -D
 -Uflash:w:svetla\ovladani-svetel.cpp.hex:i
```

```
@rem Spuštění PDF
"C:\Program Files (x86)\Foxit Software\Foxit Reader\FoxitReader.exe"
 carodejky-ze-salemu.pdf
@rem Vyčištění Arduina pro další experimentální práci
"C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe"
 -C"C:\Program Files
 (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -v -patmega2560
 -cwiring -PCOM4 -b115200 -D -Uflash:w:svetla\prazdny-soubor.cpp.hex:i
```

4. Strana PDF: generování

Z pohledu automatizace celého procesu je však výhodné si várky znaků připravit dopředu či nechť mi je generuje TEXový soubor při přípravě PDF. Ukáži cestu, jak TEX generuje dávkové soubory, které zařizují odeslání dat Arduinu.

Pracuji přímo s BAT/SH soubory a poněvadž je bezpečnost i na straně TEXu, tak generuji dočasný soubor, který se na požadovaný název přejmenuje.

TEX se spouští se zapnutým `--shell-escape` (TEX Live), na to mám extra profil v editoru TEXworks. To TEXu dovoluje volat si příkazy z operačního systému. Šlo by to udělat bez toho, vygenerované soubory by se přejmenovaly po dokončení TEXování (např. `Bash`, `Makefile`).

```
% !TeX program = LuaLaTeX (+shell) TeX encoding = UTF-8
\documentclass[12pt,a4paper]{article}
\usepackage{hyperref}
\newwrite\zapis\newcount\citac
\def\malcom{COM6}\def\maltty{ttyS5}
\def\malukladej{soubory}
% \immediate\write18{mkdir \malukladej} % TeX přímo nepustí.
\def\klik#1 {\advance\citac by 1%
  \def\malformat{\ifnum\citac<100 0\fi\ifnum\citac<10 0\fi\the\citac}%
  \immediate\openout\zapis=\malukladej/\malformat.tex%
  \immediate\write\zapis{echo #1 >xfile.txt}%
  \immediate\write\zapis{print /D:\malcom\space xfile.txt}%
  \immediate\closeout\zapis
  \href{run:\malukladej/\malformat.bat}{#1}%
  \immediate\write18{cd \malukladej\space && mv \malformat.tex
 \malformat.bat}
  \immediate\openout\zapis=\malukladej/\malformat.sh%
  \immediate\write\zapis{echo #1 >/dev/\maltty}%
  \immediate\closeout\zapis  }
\begin{document}
  vzorek \klik ABCDE nebo \klik FGHIJKL \par
  vzorek \klik abcde nebo \klik fghijkl \par
```

```
  vzorek \klik ABCDEabcdeABCDEabcde nebo \klik fghijklFGHIJKLfghijkl
 \par
  vsechno zapnout: \klik ABCDEFGHIJKL \par
  vsechno vypnout: \klik abcdefghijkl
\end{document}
```

Po ručním vytvoření si adresáře `soubory` do něj TEX ukládá dávkové soubory, postupně očíslované. Příkaz `\klik` si načte všechny znaky až po mezeru a zapíše je. Hypertextové odkazy zajišťuje balíček hyperref. Zde je ukázána cesta přes `run:`, o vylepšeném způsobu v Linuxu přímo pomocí odkazů za okamžik.

5. Strana PDF: spouštění odkazů pod Microsoft Windows

Před několika lety (OSSConf 2013, Jak se TEXista mezi..., str. 131–138) jsem řešil problém spuštění dávkových souborů přes PDF, abych mohl spouštět hudbu a zvuky u místních ochotníků. Vyřešil jsem to pod Microsoft Windows, v programu Foxit Reader a pomocí programu MPlayer. Používám to na zkoušky i divadelní produkci, jen jsem přešel na MPV, odnož s možností konfigurovatelnosti přes jazyk Lua, který v TEXovém světě frčí.

6. Strana PDF: spouštění odkazů v Linuxu

Co mě však trápilo, bylo, že jsem neměl řešení pro svět Linuxu. V `xpdf` sice dávkový soubor spustíte (testy roku 2013 i 2016), ale vyskočí vám dialogové okno, což je nepřijatelné zdržení na zkoušky či divadelní produkci.

Během psaní článku mě něco osvítilo (procházení manuálu `xpdf`), že bych se měl na dávkové soubory dívat jako na odkazy (po kliknutí se otevírá webový prohlížeč) a jen to trochu předefinovat. Po nainstalování `xpdf` (testováno v Lubuntu)

```
$ sudo apt install xpdf
```

si v domečku stačí vytvořit konfigurační soubor `.xpdfrc` a přidat jeden řádek:

```
urlCommand "sh '%s'"
```

V TEXovém světě (balíček hyperref) je pak každý odkaz (nepíšeme ani `run:`) zvažován jako dávkový soubor. Je to jádro, se kterým by se dalo dál pracovat, pro divadelní produkci je to řešení nejkratší. Jednoduché uvozovky zajišťují práci se soubory, kde je v názvu mezera, což by nebyl můj případ, takže by se to dalo ještě o dva znaky zjednodušit. Co jedno kliknutí v PDF, tak jedno spuštění.

Praktické by bylo užít `urlCommand "sh rozcestnik.sh '%s'"`, kde by rozcestník otestoval, zda-li je parametr spustitelný soubor (postoupil by `sh`) nebo jiný soubor (postoupil by například v `xpdf` předdefinovanému chování `netscape -remote 'openURL(%s)'`). Nechť je to pro zájemce jako domácí úkol.

Tím se uspokojivě řeší oba světy (Mac OS X už trochu považuji za linuxový svět a dál to netestuji). Shodou okolností jsem nalezl ještě jiný přístup. Obejít bezpečnost přes vlastní příponu, které se nastaví příslušná `mime` pravidla. Pro zájemce viz `https://tug.org/pipermail/texhax/2010-September/015705.html`. Bylo by poutavé zkoumat nastavitelnost dalších programů (Evince, Adobe Reader, KPDF ap.) v tomto duchu, ale to je nad mé možnosti a rozsah článku. Problém je to zajímavý i u webových stránek (řeší např. PHP, JavaScript ad.).

7. Možná vylepšení a posuny kupředu

- Rád bych nalezl stabilní a bezdrátovou komunikaci mezi notebookem a Arduinem, ale teoreticky i mezi několika Arduiny, případně notebookem a více Arduiny (zvažuji RF i WiFi: XBee, ESP8266 či Slavko Fedorik radí HC-11, možná zkusit přímo Arduino Uno WiFi či Tiān, WeMos D1, MKR1000, nRF24L01+ či RS485). To by mi pomohlo s praktickým zasíťováním jeviště. Na druhý konec bych si dal Arduino a světla bych nemusel síťovat každé zvlášť ze svého technického koutku, kde mám zatím jediné Arduino. WiFi zní jako přípustné řešení, ale bude to chtít asi vlastní WiFi router, protože místní WiFi se nezdá stabilní. Zvlášť je ve spodním patře pod jevištěm mikrovlnná trouba jako součást kuchyně a zvuková aparatura jako součást restaurace.
- Testy se spuštěním animací (AV) za sebou nemám. To zatím řeším dalším notebookem, který má na starosti jen to. Spustit VLC či MPV někde bokem by mělo jít. AV výstup používám na LCD monitor jako nápovědka s texty.
- Podobně jsem nepostoupil s testy u Vixen Lights (`www.vixenlights.com`), programu, který by měl usnadnit přiřazení hudby a světel. Na YouTube je to oblíbené téma u vánočních světýlek (viz Scott Shaver).
- Jistou možnost dává program PD (`www.puredata.info`, Pure Data), ale to chce hlubší studium, viz například kniha Jan Kavan: Pure Data: platforma pro tvorbu interaktivního díla, ISBN 978-80-7460-033-3.
- Největší plány mám u světel. Mám za sebou první úspěšné pokusy u stmívačů (IGBT-N i BT136/BT139; 60 i 200W žárovky), u toho si chci vyzkoušet přípravu plošných desek s následným osazením. Cesta přes IGBT bude o něco lepší skrz programování (nedohledávání přechodu přes nulu) a možná s menším stupněm rušení. V hledáčku mám HGTG30N60A4D, ale musí se to koupit přes AliExpress či eBay, u nás je to drahé. To bude chtít inovovat jednobajtové sekvence jako příkazy, protože potřebujeme zaslat nejen kanál, ale i intenzitu (0 až 255, tedy dva bajty). Je dost možné, že bude víc Arduin, tak se bude muset i upřesnit, které Arduino příkaz vykonává.
- Rád bych zahrnul spotřebiče ovládané přes IR dálkové ovladače. Například RGB LED reflektor, který je takto řízen. Signály z ovladače dokáží číst, ale nedaří se mi je zpátky vyslat.

- Rád bych vyzkoušel RGB LEDkové pásky. Každý barevný kanál se řídí zvlášť, takže se na to jde trochu jinak než u obyčejných žárovek. Dává to však naději, že divadlo bude konečně trochu řádně osvětlené (tlumené světlo, barva, podsvěcování užších ploch).
- Komunikace a řízení přes protokol DMX512 by mělo usnadnit práci s RGBW kanály. Převodníky mám vyhlídlé dva (OpenDmx a místní firmu SOH.cz).
- Můj největší plán je vyzkoušet si práci s RGB LED čipy, například s WS2812B či dražšími APA102. Doplňkem by měly být silné XML T6, SMD či COB LEDky, ze kterých bych rád udělal mobilní reflektory ve stylu baterek či čelovek. To by snad pomohlo i se zajištěním barevných filtrů. Velikost by dala možnost napojit krokové motorky či servomotorky a rotaci takového reflektoru, to je aktuálně hudba možné budoucnosti.

Závěrem

Mohu říci, že jsem vlastně spokojen, během roku jsem projekt dostal do fáze, že jej mohu používat a celý proces není pro mne černá skříňka. Respekt z nízkého (ale potenciálně smrtelného napětí) je velký, ale během bastlení mám vše vypnuté a proud zapínám až jsem hotov. Testy komunikace jsem prováděl a sledoval LEDky u relátek, to stačilo. To se obvykle dělá tak, že vedle produkčního Arduina máte další bez zapojených spotřebičů. Já to dělal tak, že jsem si zapojil úplně vše, ale přívodní kábl zásobující spotřebiče jsem neměl v zásuvce, tedy celou soustavou probíhalo maximálně 5 V=.

Velice zajímavá situace by nastala, kdyby některé z relátek odešlo, to by se dalo řešit paralelním napojováním více relé, nebo na TEXové úrovni předefinováním výstupu. Tak hluboko jsem se v projektu nedostal a prakticky bych asi přehodil žílu na desce s relátky z nefunkčního na volné relátko.

Zklamání je z těch malých 20cm káblů u Arduina. Na druhou desku s relátky od Arduina to mám vedené nějak přes prodlužky, ale není to ono. Jakmile se pustím do stmívačů a dalších projektů, logistika káblíků bude taky důležitá.

Jiří Rybička mě potěšil, že by měl rád nasvícená kolejiště. To by mohl být zajímavý projekt pro nejbližší období. Budiž světlo!

Kontaktní adresa

Ing. Pavel Stříž, Ph.D., Nakladatelství Martin Stříž, U Škol 940, 685 01 Bučovice, Česká republika,

E-mailová adresa: pavel@striz.cz

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk
OSSConf

OSSConf 2016: 143–148

INTERACTIVE OUTDOOR GAME BASED ON NFC "FIND THE TREE"

PETER ŠARAFÍN (SK), RÓBERT ŽALMAN (SK), MICHAL CHOVANEC (SK) AND VERONIKA OLEŠNANÍKOVÁ (SK)

Abstract. Proposed game is based on using the RFID and NFC technology. The main idea of the game is to collect the Tree Points deployed in given area. The game has the motivational and education feature. This article briefly describes the functional principle of the used technology, explains work with online tool MIT App Inventor and offer Open-SW solution for similar tasks.

Key words and phrases. RFID, NFC, interactive game.

INTERAKTÍVNE OUTDOOROVÁ HRA S VYUŽITÍM NFC „FIND THE TREE"

Abstrakt. Vytvorená hra je založená na využití RFID a NFC technológie. Princíp hry spočíva v zbieraní Tree Pointov rozložených v danej oblasti. Hra má motivačný a edukačný charakter. Tento článok stručne popisuje princíp fungovania použitej technológie, vysvetľuje využitie online nástroja MIT App Inventor a ponúka Open-HW riešenie pre podobné typy úloh.

Kľúčové slová. RFID, NFC, interaktívna hra.

Introduction

Nowadays almost every person owns a smartphone, especially young generation including children in preschool years. These people can use various technologies, they can work intuitively and they like to learn new things in a playful way. Our game "Find The Tree" has an education nature and it is supposed to motivate young people to know the nature around better and meanwhile gain the best score. The rules of the game consist of downloading the application to their smartphone equipped with the NFC readers. At be beginning of the game, the player has the first clue how to find the "Tree Point" in chosen area. The set of clues is different for different levels:

- **Basic Level:**
  - *Tree description:* The picture of the tree leaf, the height of the tree, the height of the branches above the ground, the diameter of the tree, bark color of the tree.

- *Position description:* E. g. go straight on the path for 30 meters/60 steps, cross the pond, then turn to the left and look for the tree based on given description.
- *Help:* There is a help as GPS coordinates, picture with the position of the tree on the detailed map but if the palyer use it, points from his/her score will be subtracted.

- **Pro Level:**
  - *Tree description:* The description of the tree leaf, the height of the tree/bush, height of the branches above the ground, the diameter of the tree/bush, flower/bark color of the tree/bush.
  - *Position description.*
  - *Help.*

In the basic level, there is a group of the most familiar tree exemplars for given area. The pro level contains not so familiar trees and also bushes of the given area. When the player finds the right tree after reading the "Tree Tag" by the application there will be displayed more detailed information.

About the tree:

- picture of the individual tree (with an option for reporting the Tree Tag, if it is not correctly placed or in the case of other issues);
- an interesting fact about given kind;
- the clue to other "Tree Point";
- information about gained points and the player's score.

The player can browse the gained Tree Points and see the score anytime. The suitable room for deploying this game are the parks, wood parks, educational tourist's paths and so on. The game could be used even in the elementary schools at natural history classes or in outdoor school activities. The children can get familiar with nature around in a playful way and verify gained knowledge from the books. The game can fill the motivation role also and encourage more people to active approach to the leisure activities and make the stay in the nature interactive.

1. Near Field Communication and RFID tags

NFC is a way of wireless and contactless communication similar as Bluetooth or WiFi but instead of radio transmission uses the electromagnetic radio field. There are three types of NFC standards: A, B and FeliCa (used in Japan). Two types of devices are used – passive and active. Passive devices hold the information but cannot communicate by itself. Unlike the other devices, usage of RFID tags does not need any external power source because the NFC reader is acting as the power source for the passive RFID device in the moment of communication. They might be used as RFID tags that other devices can read. An active NFC device

can read and also transmit the information, e. g. smartphone can read RFID tags, but also exchange the data with other active device, alter data if authorized or even can emulate the credit card. NFC standard has against RFID standards extra security features. There are methods preventing eavesdropping in two ways – first NFC technology itself is supposed to be used at short distances, in case of using credit cards payments the range is usually up to 6 centimetres. Also the secure channel is established, the information are encrypted and only authorized device is able to decode it. Some readers have also implemented features for listen to data corruption attacks by the third side. It can happen when the data are altered on the way between transmitter and receiver. Other way how to secure NFC communication is active-passive pairing. There is always chance that the phone is stolen, the customer can always protect the phone by setting a password or other type of lock. Even if it seems that NFC opened sensitive information to new risks, the truth is it may be even safer than a stolen credit card from the wallet [1].

Our application uses the theory of passive RFID tag which is read by active device in our case a smartphone. Nowadays most of the smartphones are equipped with NFC readers/writers. The users need the NFC reader for reading and collecting the Tree Points. In the field, the RFID tag is attached to the chosen tree. There were two possible solutions what data should the tag stored. The first way is to record all the required information about the tree into the tag. The application then would read the whole information. There is also another approach – write to the tag only the unique identifier, e. g. the name of the tree, number of the tree. We chose the second methodology for more reasons. The used RFID tags have the memory capacity of 1KB, which could be sufficient for our application. However the fewer information is kept in the tag the faster the communication with NFC reader is. Also if the tag is lost, we can easily replaced it, just with need of record the number or the name of the tree with any NFC device without need of database information. All the information about the tree are stored in the application and the identification of the Tree Tag is based on the main identifier.

2. MIT App Inventor

Since the most of the smartphones use Android operating system, we decided to create aplication for this OS. There was a motivation to find a free online tool where the source code can be easily understandable for wide society. We consider the MIT App Inventor as a very suitable for its graphical environment and it is also the known tool with strong background.

The current version of the program MIT App Inventor has origin as the result of cooperation between Google Company and Massachusetts Institute of Technology. It is an innovative program for beginners intended for programming and

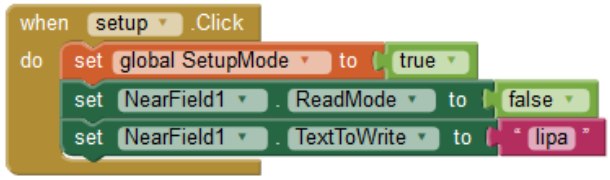


Figure 1. Sample code for setup the RFID tag

creating applications. It transforms complex text oriented language into visual language and the environment is designed of drag-and-drop blocks. The goal is to provide a tool for creating the applications for beginners and as the same to define certain standard of the quality of applications. Users are not forced to waste their time by writing the source code, because the application offers basic chain of pre-programmed functions. As an open-source tool, the App Inventor tries to put programming and creating of the applications accessible for wide spectrum of users. It is mainly aimed for formal and informal lecturer of computer science, government employees and volunteers, designer and product managers who see the potential of open-source software, as well as scientific and research workers for creating own application used for collection, processing, analysing and evaluation of data in many scientific fields. The community of people supporting MIT App Inventor is composed almost of 3 millions users from 195 countries and the main partners as Scheller Teacher Education Program, MIT Media Lab and MIT Computer Science and Artificial Intelligence Lab (CSAIL) [2].

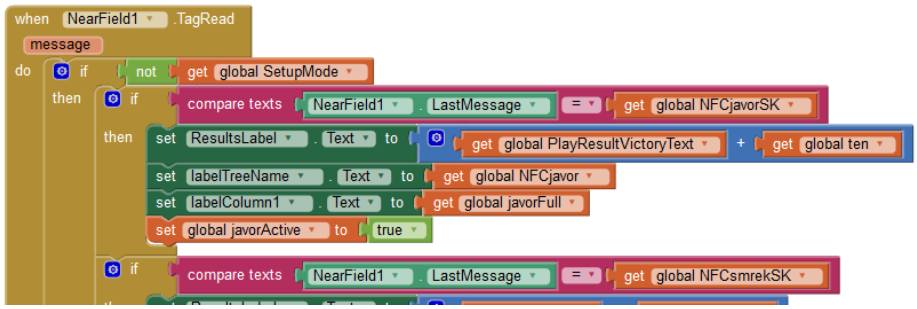


Figure 2. Sample code for reading the RFID tags

The source code of the application can be found at [3]. To be able to build the application based on reading the RFID tags, there is a need to setup the basic function writing required data to the tag. As can be seen in the Fig. 1, the simple pre-build block structure exists. This simple method will be disable in final release in the application, it is used only for initialization process, where we need to set the tags.

The next of the main functions is show in the Fig. 2. This provides the reading of the deployed tags. At first there is the check if the application is not in the setup mode, which stands for writing to RFID tags. Afterwards, the read data from the tag is compared to the given conditions. Based on the string provided by the tag, other actions are executed. Fig. 3 informs the application in which state the game is and which tree is the player looking for.

In the Fig. 4 the final application is presented. There are two states, the left screen shows the hints about the wanted tree and the right screen displays after getting data from the RFID tag belonging to "maple" string.

Figure 3. Sample code for button action

3. Conclusions

Find The Tree application was deployed for testing purposes into the woodpark called "Lesopark Chrasť" in the city of Žilina, Slovakia. Some bugs were fixed while test run. There is also few ideas how to improve the application. For example add the option for choosing the "playground" since there can be deployed more games in wide area.

References

[1] *Near Field Communication official site*, `http://www.nearfieldcommunication.org/`.
[2] *MIT App Inventor official site*, `http://appinventor.mit.edu/`.
[3] *The source code of Find The Tree application*, `https://drive.google.com/file/d/0B_7ia4U-9wbAS0s4TWdqZk0tYmM/view?usp=sharing`.

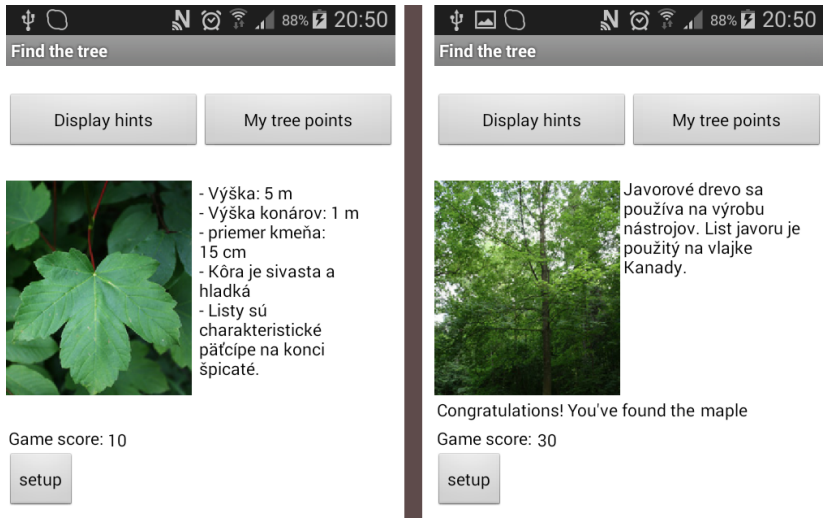


Figure 4. Screenshots of the application states

Contact addresses

Ing. Peter Šarafín, Department of Technical Cybernetics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: `peter.sarafin@fri.uniza.sk`

Ing. Róbert Žalman, Department of Technical Cybernetics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: `robert.zalman@fri.uniza.sk`

Ing. Michal Chovanec, Department of Technical Cybernetics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: `michal.chovanec@fri.uniza.sk`

Ing. Veronika Olešnaníková, Department of Technical Cybernetics, Faculty of Management Science and Informatics, University of Žilina, Univerzitná 8215/1, 010 26 Žilina, Slovak Republic,
E-mail address: `veronika.olesnanikova@fri.uniza.sk`

8. ročník
Žilina 29. 6. - 1. 7. 2016
Otvorený softvér vo vzdelávaní, výskume a v IT riešeniach
ossconf.soit.sk

WEBMAP JE OTVORENÝ PROJEKT PRE KOMUNITU ŠTUDENTOV A UČITEĽOV GIS

Martin Kalivoda, Miloslav Ofúkaný

V roku 2011 vznikol WebMap ako myšlienka na poskytovanie online GIS služieb pre študentov, ktorí prostredníctvom hostingu na virtuálnom privátnom serveri (VPS) umiestňujú svoje GIS webové aplikácie na stránku www.webmap.sk. Medzi jej užívateľmi z Prírodovedeckej fakulty UK Bratislava sa pred dvoma rokmi vytvorilo silnejšie odborné puto a vytvorili tím WebMap, ktorí sa prezentoval už na konferencii OSSConf 2015 v sekcii OpenGIS. Od januára 2016 ich aktivity inštitucionálne zastrešuje klub č. 962 pod Asociáciou pre mládež vedu a techniku (AMAVET), ktorej grantová komisia o dva mesiace neskôr rozhodla o finančnej podpore pre projekt WebMap – GIS hosting pre školské projekty. Po aprílovom podujatí OSS Víkend Bratislava sa tím rozrástol o nových členov.

Cieľom projektu je poskytnúť študentom a učiteľom mapový server, priestorovú databázu a geoprocesné nástroje, všetko postavené na open-source technológiách, aby mohli efektívne publikovať vlastné geografické dáta a GIS aplikácie. Budúcim užívateľom chce WebMap priniesť komparatívne zvýhodnenie na trhu práce v oblasti geoinformatiky. Ponúka im takto príležitosť publikovať v internetovom prostredí svoje geografické práce na obhajobách, konferenciách alebo počas pracovných pohovorov.

WebMap server funguje nad VPS s Linuxom, na ktorom sa používa PostGIS, Geoserver, OpenLayers, GRASS a i. Na vývoj administratívneho rozhrania sa využíva jazyk JavaScript – okrem frontendu aj backend pomocou Node.js. Aktuálne prebieha vývoj administratívne rozhranie GIS hostingu, aby sa proces publikovania študentských aplikácií mohol automatizovať. Na hostovanie sa prijímajú GIS aplikácie vytvorené nad akoukoľvek open-source technológiou.

Otvorenosť projektu je deklarovaná aj prijímaním nových členov do AMAVET klubu č. 962 a používateľov WebMapu. Nadšenci geoinformatiky môžu participovať nielen pri vývoji GIS hostingu, ale vďaka klubovej činnosti získajú aj iné technické školenia, redakčné skúsenosti pri tvorbe obsahu informačných portálov, zlepšia si svoje komunikačné a líderské zručnosti.

Projekt WebMap – GIS hosting pre školské projekty bol podporený z dotácie Ministerstva školstva , vedy, výskumu a športu SR „Programy pre mládež 2014–2020“, ktorú administruje IUVENTA – Slovenský inštitút mládeže.

JUPYTER NOTEBOOK

Michal KAUKIČ

Jupyter je webové rozhranie pre prácu s mnohými (interpretovanými aj tradične kompilovanými) jazykmi. Vznokol zovšeobecnením projektu IPython, keď sa funkcionalita webového rozhrania oddelila od jadier (kernels) pre jednotlivé jazyky. Prvým bol, samozrejme, kernel pre Python, ale skoro súčasne sa objavil kernel pre jazyk Julia a pre otvorený systém pre štatistické výpočty – jazyk R. Jupyterhub NB server, využitie.

V rámci pracovného stretnutia ukážeme, ako si Jupyter notebook nainštalovať, povieme si o jeho využití pre prácu so skupinami používateľov (študentov) a o autorizácii napr. cez LDAP (riešenie Jupyterhub). Predvedieme niekoľko ukážok, hlavne grafických, s použitím knižnice Matplotlib, ale aj interaktívne „widgets“ pomocou knižnice Bokeh, ktorá transformuje naše príkazy v pythone do jazyka javaScript, ktorý je zrozumiteľný pre webove prehliadače. Ukážky budú hlavne z matematickej analýzy (plochy – funkcie dvoch premenných, ich dotyčnice, dotyková rovina), algebry (zaujímavá hra so zhášaním svetiel), či z oblasti približných výpočtov (približné určovanie plôch, prekladanie kriviek cez body, atď.). Zaujímavou je tiež ukážka využitia OpenStreet máp v prehliadači, kde si pomocou modulu Folium môžeme znázorňovať napr. polohy členov určitej komunity a údaje o nich.

Ukážeme tiež prvky GUI (grafického užívateľského prostredia), napr. posuvníky, ktorými môžeme interaktívne meniť parametre modelu vidieť tie zmeny na príslušných grafoch. Používatelia si tiež niektoré veci môžu samostatne vyskúšať, máme pre nich pripravené Jupyter notebooky na serveri invimath.fri.uniza.sk (v rámci KEGA projektu o tom, ako sa v matematike dajú veci názorne zviditeľňovať).

SLEDOVANIE VÝŠKY SNEHU NA PRINCÍPE CROWDSOURCINGU

Pavel KRAJČÍ

Príspevok sa zaoberá alternatívnym spôsobom získavania informácií o výške snehovej pokrývky v oblasti Západných Tatier na severnom Slovensku. Tieto informácie sú dôležité hlavne pre návštevníkov hôr v zimnom období. Pre tento účel využíva princíp crowdsourcingu. Tomuto princípu sa venuje najprv teoreticky a následne prakticky pri tvorbe internetovej mapovej aplikácie. Táto aplikácia informácie nielen vizualizuje ale umožňuje užívateľovi ich aj aktualizovať a vylepšovať.

PARALELIZÁCIA INTERPOLAČNÉHO MODULU V.SURF.RST GEOGRAFICKÉHO INFORMAČNÉHO SYSTÉMU GRASS POMOCOU KNIŽNICE OPENMP

Michal LACKO, Stanislav ZUBAL, Jaroslav HOFIERKA

Cieľom príspevku je paralelizácia interpolačného modulu v.surf.rst geografického informačného systému s otvoreným kódom GRASS s cieľom eliminovať čas potrebný na výpočet a vytvaorenie digitálneho modelu terénu. Dosiahnuté výsledky indikujú redukciu času potrebného na výpočet viac ako 40 percent. V porovnaní so sekvenčnými operáciami CPU dosiahnutými jednotlivými jadrami tento výsledok dosahuje značné zlepšenie. Výsledkom je modul v.surf.rst.mp implementovaný v GIS-e GRASS, ktorý je určený pre počítače s viacjad-rovými dátovými procesormi. Redukcia výpočtového času pomáha profesionálnym pracovníkom zlepšovať efektivitu ich práce.

INTEGRATION WITH RED HAT JBOSS FUSE

J. LUDVÍČEK

The purpose of this talk is an introduction of Red Hat JBoss Fuse with focus on Apache Camel. Camel is an integration framework with a rich (and still growing) community of users. The talk is designed as a first encounter with Fuse and related tooling.

SMARTHOUSE ON SILVERTHINGS

P. MACÍK, M. JAROŠ

Come witness the Internet of Things solution covering all the aspects from connected sensors through service integration and mobile interaction up to the infrastructure level powered by Red Hat's Middleware products such as BRMS, OpenShift and AM-Q and built on open-source IoT projects from SilverThings family.

IOT ON SILVERTHINGS PLATFORM

P. MACÍK, M. JAROŠ

See how to build simple IoT solution, that is written once and deployed (almost) everywhere. Describe hardware world with Bulldog library, integrate your apps with Silverspoon and command them with SilverWare.

NASAZENÍ OTEVŘENÉ PLATFORMY CRISMAPP PRO VIZUALIZACI ZEMĚTŘESENÍ V NORSKU

Rostislav NÉTEK

Příspěvek popisuje reálné nasazení open-source platformy Crismapp. Jedná se o webové GIS řešení, určené primárně pro potřeby krizového managementu. V rámci výzkumné stáže na Univerzitě v Bergenu byla Crismapp implementována pro potřeby monitoringu zemětřesení v oblasti Norska v rámci celoevropské pozorovací sítě EPOS. Aplikace postavená na technologii HTML5, reflektuje princip WebGIS 2.0 a využívá v co nejvyšší míře otevřené nástroje jako např. mapovou knihovnu Leaflet či datový formát GeoJSON. Vedle vizualizačního klienta disponuje kompletním administračním rozhraním. Příspěvek seznámí posluchače s přednostmi platformy Crismapp včetně demonstrace její flexibility na konkrétním nasazení pro vizualizaci zemětřesení v Norsku.

Prezentovaný výzkum byl podpořen v rámci programu Norských fondů CZ07, číslo projektu NF-CZ07-INP-5-287-2015. Projekt „GIS for crisis management in Norway – Sharing know-how“ si klade za cíl vytvořit vhodné zázemí pro vytvoření úspěšného partnerství na bilatelární a mezinárodní úrovni mezi institucemi a jednotlivci zapojenými do vzdělávací sféry v České republice a v Norsku. Za obsah přednášek a publikací ručí výhradně autor a vydavatel. DZS a Kancelář finančních mechanismů neodpovídají za žádné případné užití dotčených informací.

SATELITNÉ SNÍMKY A Z NICH ODVODENÉ PRIESTOROVÉ ÚDAJE PRÍSTUPNÉ AKO OPENDATA

Jozef NOVÁČEK, Ján TÓBIK, Peter PASTOREK, Martin TUCHYŇA, Tomáš KLIMENT

Priestorové údaje postavené nad diaľkovým prieskumom Zeme sa aktivitami v oblasti vesmírneho programu Európskej vesmírnej agentúry (ESA) výrazným spôsobom priblížili spracovateľom. Zjednodušený proces prístupu k satelitným snímkam zefektívnil spracovanie a umožnil vznik novým technológiám, postavených na automatizovanou procese. Tento fakt do výraznej miery prispel k zmene prístupu k takto vytvoreným údajom a zároveň vytvoril spôsob prístupu k samotným podkladovým údajom. Prezentácia sa v svojom základe prikláňa k aktivitám Európskeho spoločenstva – COPERNICUS a zároveň poukazuje na reálne výsledky kooperácie členských štátov v oblasti tvorby a sprístupnenia priestorových údajov pod licenciou otvorených dát. Príspevok taktiež poskytne informáciu o možnostiach podpory pri overovaní využitia satelitných snímok v oblasti tvorby SK Open Land Use Map.

PRIESKUM VYUŽÍVANIA OSS PRE GIS NA ŠKOLÁCH A V PRAXI

Miloslav OFÚKANÝ, Jakub FUSKA

Cieľom prieskumu, organizovaného združením AMAVET klub č. 962, bolo zmapovať užívateľské skúsenosti s open-source softvérom (OSS) pre geografické informačné systémy (GIS) na školách (ZŠ, SŠ, VŠ) a v praxi (verejné inštitúcie a súkromné spoločnosti).

Respondentov sme sa pýtali, ktorý je ich najdôležitejší zdroj informácií o softvéri pre GIS a či poznajú filozofiu OSS. V prípade, že o princípoch OSS už mali informácie alebo vlastné skúsenosti, tak ďalej vymenovali používané proprietárne a OSS programy pre prácu s GIS, CAD, databázami a grafikou. Potom sme zisťovali 3 najdôležitejšie výhody a nevýhody OSS pre GIS, spoločenské oblasti ich využitia, používané metódy a nástroje.

Osoby odpovedajúce v prieskume taktiež hodnotili (stupnicou od 1 do 5) spokojnosť s používaním, vyučovaním a propagáciou OSS pre GIS na školách a v praxi. Dostali priestor aj na vyjadrenie svojho osobného postoja vo vzťahu k vzdelávaniu a šíreniu osvety otvorených softvérových riešení. Pedagógovia ďalej porovnávali používanie OSS a proprietárneho softvéru pre GIS vo výučbe z pohľadu skúseností svojich žiakov alebo študentov.

Záverečnými otázkami boli získané demografické údaje respondentov – pohlavie, vek, miesto pobytu počas ZŠ, SŠ, VŠ. Stimulačným nástrojom pre uvedenie e-mailovej adresy bola v dotazníku možnosť výhry vecnej ceny.

Priebežné výsledky prieskumu boli prezentované nielen vo forme tabuliek a grafov, ale aj pomocou mapových výstupov spracovaných v programe QGIS.

MYGEODATA CLOUD

Antonín ORLÍK

MyGeodata Cloud je webový portál, který umožňuje ukládání, sdílení a publikování prostorových dat, jejich konverzi, transformaci a vizualizaci. Je tvořen třemi dílčími aplikacemi: MyGeodata Drive – webová aplikace, určená pro svobodné sdílení prostorových dat v prostředí internetu. Uživatelé zde mají přidělený diskový prostor, který mohou využít k nahrání jak vektorových, tak i rastrových prostorových dat. Po nahrání dat se provede automatická detekce a evidence metadat – uživatel pak může některá metadata upravit, zadat klíčová slova a přidělit datové sadě licenci a bližší popis. Data pak může sdílet s konkrétními uživateli nebo je publikovat veřejně – v tomto případě pak taková datová sada bude zahrnuta do výsledků vyhledávání pro ostatní uživatele. Vyhledávání je umožněno pomocí zadání klíčových slov, která data obsahují v názvu nebo v metadatech. Dále je možné aplikovat filtr dle geografické oblasti zájmu a dle typu dat (bodové,

liniové, polygonové, rastrové). Umožněno je data i seřadit podle různých kritérií. U vektorových dat je umožněno zobrazit atributovou tabulku hodnot nebo data zobrazit v mapě. Uživatelem vybrané datové sady pak je možné stáhnout v původním formátu nebo je konvertovat do jiných datových formátů či souřadnicových systémů. MyGeodata Converter – webová aplikace, určená pro provádění formátových konverzí a transformaci souřadnicových systémů prostorových dat – vektorových i rastrových. Je podporována většina nejpoužívanějších formátů a souřadnicových systémů z oblasti GIS a CAD systémů. Uživateli je umožněno nahrát a konvertovat či transformovat svá vlastní geodata nebo může konvertovat data, která jsou sdílená jinými uživateli v aplikaci MyGeodata Drive. Po nahrání nebo výběru dat může uživatel zvolit výstupní datový formát, výstupní souřadnicový systém a jiné parametry pro danou konverzi. Výsledek konverze je možné stáhnout komprimovaně – ve formátu ZIP. MyGeodata Map – jednoduchá mapová prohlížečka prostorových dat, uložených na MyGeodata Drive.

Webový portál používá systém pro správu obsahu Django CMS s využitím návrhových šablon Bootstrap 3.3 a je dostupný prostřednictvím běžných webových prohlížečů na adrese https://mygeodata.cloud. Serverová část je realizována v programovacím jazyce Python 2.7. Klíčovou programovou knihovnou pro manipulaci s prostorovými daty je knihovna GDAL/OGR, verze 2.0.2. Pro vizualizaci geodat je na straně serveru použit MapServer 7.0, na straně klienta pak OpenLayers 3.

OPENSTREETMAP, FREEMAP A OMA V ROKOCH 2015 A 2016

Michal PÁLENÍK

V prednáške si predstavíme aktivity komunity okolo projektu OpenStreetMap (nielen) na Slovensku. Ukážeme si služby portálu www.freemap.sk (vrátane jednotlivých API), poukážeme na možnosti a služby portálu www.oma.sk (vyhľadávanie, listovanie, API), ako i na horúce novinky (najbližšia MHD, interaktívny výškový profil trás, predpoveď počasia...).

Vydanie tohto zborníka bolo podporené grantom slovenskej kultúrno-edukačnej agentúry KEGA č. 011ŽU-4/2014 „Experimentálna matematika – zviditeľnenie neviditeľného“.

OTVORENÝ SOFTVÉR VO VZDELÁVANÍ, VÝSKUME A V IT RIEŠENIACH
Zborník príspevkov medzinárodnej konferencie OSSConf 2016,
konanej 29. júna–1. júla 2016 v Žiline

Prvé vydanie 2016
Elektronická sadzba programom pdfLaTeX Rudolf Blaško
Vydala Žilinská univerzita v Žiline
Tlač EDIS – vydavateľstvo Žilinskej univerzity v Žiline
Náklad 100 ks

ISBN 978-80-554-1292-4

ISBN 978-80-554-1292-4