VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROGRAM PRO AUTOMATICKÉ GENEROVÁNÍ KŘÍŽOVEK

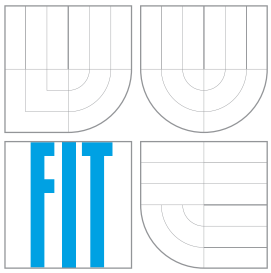
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. IVO SKALICKÝ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PROGRAM PRO AUTOMATICKÉ GENEROVÁNÍ KŘÍŽOVEK

PROGRAM FOR AUTOMATICAL GENERATION OF CROSSWORDS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. IVO SKALICKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. ROMAN LUKÁŠ, Ph.D.
SUPERVISOR

BRNO 2009

Abstrakt

Práce se zabývá návrhem algoritmu pro automatické generování švédských křížovek a záležitostmi s touto problematikou spojenými. Vedle základních pojmů a informací o křížovkách je v práci provedena analýza velikosti stavového prostoru a popis některých vyzkoušených metod generování. Navržená a implementovaná metoda pak odstraňuje nedostatky testovaných metod. Značná část práce je věnována otázce vykreslování křížovek s cílem přímého výstupu pro sazbu, což si žádá řešení problémů, jakými je například výstup v barevném modelu CMYK. Součástí práce je plně funkční implementace generátoru a nástroje pro návrh a sazbu křížovek.

Abstract

This thesis deals with automatic computer generation of crosswords with filled-in clue algorithm design and related problems. Beside basic definitions and information about crosswords, the state space analysis and description of some tested methods of generation is performed in this thesis. Designed and implemented method removes inadequacies of tested methods. Remarkable part of text is designated to solve crossword rendering problems. Results of rendering process should be suitable for DTP without any additional touches, which includes also CMYK color space support. Fully functional implementation of crossword generator and tool for computer aided crossword design is part of this thesis.

Klíčová slova

švédské křížovky, automatická tvorba křížovek, návrh křížovek počítačem, CMYK v jazyce Java, sazba křížovek

Keywords

crosswords with filled-in clue, automatic crosswords generation, computer aided crossword design, CMYK in Java, crosswords publishing

Citace

Ivo Skalický: Program pro automatické generování křížovek, diplomová práce, Brno, FIT VUT v Brně, 2009

Program pro automatické generování křížovek

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením pana Ing. Romana Lukáše Ph.D.

.....
Ivo Skalický
18. května 2009

Poděkování

Poděkování za odborné vedení, rady a konzultace patří vedoucímu práce Ing. Romanu Lukášovi Ph.D. Za testování, návrhy, přípomínky i odborné konzultace z praxe si poděkování zaslouží: Ilona Fischerová, Petr Šoba, Pavol Zdechovan, Jiří Tužil, Miroslav Šimáček, Petr Chochole a množství dalších. Za jazykovou korekturu děkuji Mgr. Ivaně Skalické.

Obsah

Kapitola 1

Úvod

Tato práce pojednává o problematice automatického generování křížovek s využitím počítače. Některé informace a postupy vycházejí ze zkušeností získaných při tvorbě programu Crosswords - ITPro CZ[1], který byl vyvíjen od roku 2003. Hlavním cílem práce je zdokonalit tyto postupy, poučit se z chyb a vytvořit generátor křížovek zcela znovu tak, aby křížovky programem vygenerované byly kvalitnější než doposud. Ve vytvořených algoritmech je taktéž třeba využít pokroku, který byl učiněn v oblasti hardware, a to zejména výskytu vícejádrových procesorů v běžných domácích, kancelářských a přenosných počítačích.

Zdokonalit je třeba však nejen techniku generování, ale i vykreslování křížovek tak, aby vytvořený výstup bylo možné bez dalších úprav přímo vložit do vytvářené tiskoviny.

Kapitola 2 seznamuje čtenáře s historií křížovek a s tím, co vlastně křížovka je. Nalezneme zde také definice a popis jednotlivých prvků křížovky a některá doporučení a normy stanovující kvalitu křížovek podle Svazu českých hádankářů a křížovkářů.

Kapitola 3 se zabývá automatickým generováním křížovek. Je zde provedena analýza velikosti stavového prostoru, popis některých metod generování, ale i návrh nové metody generování, která je předmětem této práce. Metoda je neformálně vysvětlena a zároveň je uveden i vývojový diagram popisující činnost algoritmu. Kapitola se také zabývá možností paralelizace generování křížovek a problémy s tím souvisejícími. Při této příležitosti jsou zde také provedena výkonostní poměření různých verzí běhového prostředí Java™. Více prostoru je zde věnováno metodám vyhledávání slov ve slovníku podle zadané vyhledávací masky. Řešeny jsou zde i problémy složitosti a rozmanitosti generovaných křížovek. Tato kapitola je stěžejní částí celé práce.

V kapitole 4 se čtenář může dočíst o různých problémech, kterým je třeba čelit při vykreslování křížovek. Zajímavou součástí této kapitoly jsou informace, které se týkají podpory barevného modelu CMYK v Javě™ a o přidání podpory tohoto barevného modelu do použité exportní knihovny. Tato část může být užitečná řadě programátorů, protože o tomto tématu není na internetu dostupno mnoho informací.

Kapitola 6 naznačuje směr, kterým se bude vývoj aplikace dále ubírat.

Diplomový projekt navazuje na semestrální projekt, ze kterého byly převzaty a rozšířeny kapitoly 2 a 3. Kapitola 3 však byla výrazně rozšířena především o podkapitoly týkající se kvality generovaných křížovek.

[1] `http://crosswords.itpro.cz`

Kapitola 2

Křížovky

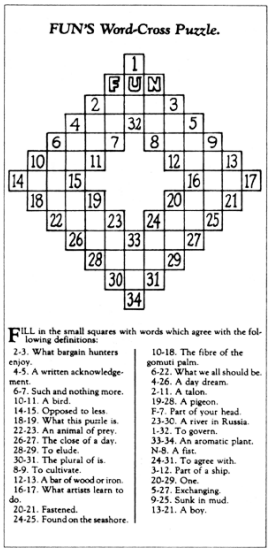
Křížovka je „druh hádanky složené zpravidla z obrazce a legendy. Obrazec je rozdělen na malá políčka, do nichž se vodorovně a svisle (křížem) vpisují hlásky nebo slabiky rozluštěných slov“ [3].

S křížovkami se v dnešní době můžeme setkat téměř v každém časopisu nebo novinách. Křížovky se tak objevují nejen jako příloha nebo zábavné doplnění tiskoviny, ale vycházejí dokonce noviny zaměřené pouze na křížovky. Poslední dobou je také velmi populární vydávat reklamní noviny distribuované zdarma, které využívají reklamní křížovky. To jsou takové křížovky, které obsahují ve svém rozložení grafickou reklamu a jejich tajenka je tematicky zaměřena na propagaci výrobku, firmy nebo služby.

2.1 Historie křížovek

Obecně je za „vynálezce“ křížovky považován Arthur Wynne, který svou hádanku „slovní kříž“, později nazývanou křížovka, uveřejnil v prosinci 1913 v *The New York Times* [12].

V roce 1924 byla vydána první kniha křížovek. V té době řada lidí považovala křížovky pouze za módní výstřelek a primitivní zábavu. V roce 1924 noviny *The New York Times* dokonce označily luštění křížovek za „hříšnou ztrátu času naprosto zbytečným hledáním slov, jejichž písmena se hodí do více či méně složité předpřipravené mřížky. Toto není vůbec hra a stěží může být nazýváno sportem... (luštitelé) z toho nic nemají kromě primitivní formy mozkového cvičení...“. *The New York Times* sice poté neotisly žádnou křížovku až do roku 1942, ale to rozhodně nezábránilo propuknutí křížovkářské mánie koncem 20. let 20. století. Křížovky se v té době začaly šířit a dobývat celý svět. V roce 1930 se pak objevil první křížovkářský slovník [12].


FUN'S Word-Cross Puzzle.

FILL in the small squares with words which agree with the following definitions:

2-3. What bargain hunters enjoy.
4-5. A written acknowledgement.
6-7. Such and nothing more.
10-11. A bird.
14-15. Opposed to less.
18-19. What this puzzle is.
22-23. An animal of prey.
26-27. The close of a day.
28-29. To elude.
30-31. The plural of is.
8-9. To cultivate.
12-13. A bar of wood or iron.
16-17. What artists learn to do.
20-21. Fastened.
24-25. Found on the seashore.
10-18. The fibre of the gomuti palm.
6-22. What we all should be.
4-26. A day dream.
2-11. A talon.
19-28. A pigeon.
F-7. Part of your head.
23-30. A river in Russia.
1-32. To govern.
33-34. An aromatic plant.
N-8. A fat.
24-31. To agree with.
3-12. Part of a ship.
20-29. One.
5-27. Exchanging.
9-25. Sunk in mud.
13-21. A boy.

Obrázek 2.1: Křížovka A. Wynnea z roku 1913

2.1.1 Křížovky v Čechách

V Čechách údajně sestavila první jednoduchou křížovku pražská učitelka Mlada Antošová již v roce 1899. Tuto hádanku tehdy nazvala „Čapka z písmen“. Samotný název „křížovka“ se však objevil v tehdejším časopise *Beseda lidu* až v roce 1923. Sestavování hádanek křížových se pak brzy stalo pro mnohé lidi doslova koníčkem. Křížovky

tehdy však mívaly daleko zajímavější vzhled. Bývaly zarámovány do více či méně zdařilých figurálních celků, jako byla například krinolína maskované dámy jdoucí na maškarní ples, podoba pštrosa či středověkého hradu. S jejich luštěním to však bývalo horší. Vzhledem ke značné autorské anarchii bylo dovoleno takřka všechno - slova ve všech pádech, v nejrůznějších tvarech i patvarech, slova obrácená, zkomolená i taková, která zná jen autor křížovky.

Z těchto důvodů žačaly všude po světě postupně na tvorbu křížovek dohlížet různé organizace. U nás je touto organizací Svaz českých hádankářů a křížovkářů (SČHAK), který byl založen v roce 1968. Podařilo se mu klasifikovat a kodifikovat stávající i nové druhy hádanek, křížovek a vícesměrek a vytvořit závazná pravidla pro jejich tvorbu a řešení.

Největší boom u nás zažívaly křížovky kolem roku 1989, kdy se dokonce po zahradničení zařadily na druhou příčku žebříčku nejoblíbenějších koníčků [11] [8].

2.2 Švédská křížovka - pojmy a definice

Švédská křížovka je nejpopulárnějším a zároveň nejrozšířenějším typem křížovky v České republice a na Slovensku. Tento typ křížovky se někdy také označuje jako *křížovka s vepsanou legendou*. Právě vepsání legendy přímo do mřížky křížovky je totiž hlavním rysem křížovek, které jsou populární v Evropě. Ve švédských křížovkách nalezneme legendu (kapitola 2.2.3) přímo v políčkách křížovky. Křížovky, které jsou obvyklé mimo Evropu mají legendové výrazy zapsány mimo mřížku křížovky a dohledávají se pomocí souřadnic políčka.

☺	HESLO	ČÍŇAN	VÝBAVA LODI	☺	OTEKLÉ	TĚLNÍ VÝMĚŠEK	DOLOVAT UHLÍ	PARALÝZA	MUŽSKÝ HLAS	OBRAZ SVATÝCH
OVŠEM	Š	A	K	VE SROVNÁNÍ S NĚČÍM	O	P	R	O	T	I
STUPEŇ CITLIVOSTI FILMU	I	S	O	MLADÝ DUB / CIZOPASNÍK	D	O	U	B	E	K
TAJENKA	**F**	**I**	**T**	**V**	**U**	**T**	**B**	**R**	**N**	**O**
AUTOR BALETU DAFNIS A CHLOE	R	A	V	E	L	KAROSOL	A	N	O	N
DIPLOMATICKÁ FUNKCE	A	T	A	Š	É	RUSKÁ ŘEKA	T	A	R	A

Obrázek 2.2: Ukázka švédské křížovky

Základními prvky křížovky jsou: název křížovky, legenda, obrazec, vpisované výrazy a tajenka. Název křížovky, legenda a obrazec jsou zpravidla součástí zadání [9].

2.2.1 Obrazec

Obrazec každé křížovky je tvořen obrysem, síťkou a oddělujícími značkami. V případě švédské křížovky musí být síťka pravoúhlá (tvořit mřížku). Plochy mezi čarami síťky se nazývají *políčka*. Oddělovací značky, které vymezují *pozice* (kapitola 2.2.2) pro vpisování slov, jsou ve švédských křížovkách tvořeny celým políčkem, do kterého je vepsán *legendový výraz* (kapitola 2.2.3). V těchto políčkách musí být uveden legendový výraz pro jeden nebo oba směry doplňování.

Obrazec by měl být co nejvíce souměrný. Nesouměrnost a neuspořádanost políček je znakem nízké kvality křížovky. Políčka pro legendové výrazy by měla společně tvořit vodorovné, svislé nebo diagonální linie. Tato políčka by se neměla objevovat osamoceně s výjimkou políček, která jsou umístěna v těsném sousedství obvodu obrazce.

2.2.2 Pozice

Pozice je tvořena souvislou řadou políček ve vodorovném nebo svislém směru. Na jednu pozici lze zapsat jedno slovo. Protože se slova musí korektně křížit, procházejí přes každé políčko dvě pozice - jedna vodorovná a jedna svislá.

V křížovce se nesmí vyskytovat pozice, která je tvořena pouze jediným políčkem. Jednopísmenné výrazy se tedy ve správné křížovce nesmějí objevit.

Průměrná délka pozic u švédské křížovky má také vliv na kvalitu křížovky. Optimální průměrná délka pozic, a tím i vpisovaných výrazů, je 4,5 písmena [10]. Při průměrné délce pozic menší než 3,5 je již křížovka nevyhovující.

2.2.3 Legenda

Legenda je souhrnem legendových výrazů. Legendový výraz je výstižnou a aktuální charakteristikou, synonymem nebo definicí vpisovaného výrazu [9]. Legendový výraz značnou měrou určuje obtížnost křížovky. Například ke slovu „PES“ můžeme napsat legendový výraz „NEJLEPŠÍ PŘÍTEL ČLOVĚKA“ nebo „POLYESTEROVÉ VLÁKNO (ZKR.)“. Rozdíl v obtížnosti těchto dvou uvedených legendových výrazů je značný. Zatímco první výraz vede jasně ke slovu „PES“ a jeho doplnění zvládne i dítě prvního stupně základní školy, druhý výraz sice také jednoznačně ukazuje na slovo „PES“, ale doplnit ho již vyžaduje jisté znalosti. Legendové výrazy by měly svou obtížností odpovídat cílové skupině řešitelů.

Běžným doplňkem legendového výrazu je i stylistické hodnocení vpisovaného výrazu, a to z hlediska spisovnosti (obecně, hovorově, knižně, básnicky, nářečně, oblastně, slangově...), z hlediska frekvence a dobového výskytu (řídce, zastarale...) a z hlediska citového zabarvení (expresivně, familiárně, dětsky...) [9].

Legendovým výrazem nesmí být přímo vpisovaný výraz. Zvláštním případem je tzv. *přesmyčková legenda*, kdy jsou všechny legendové výrazy tvořeny přímo přesmyčkou vpisovaného výrazu.

2.2.4 Tajenka

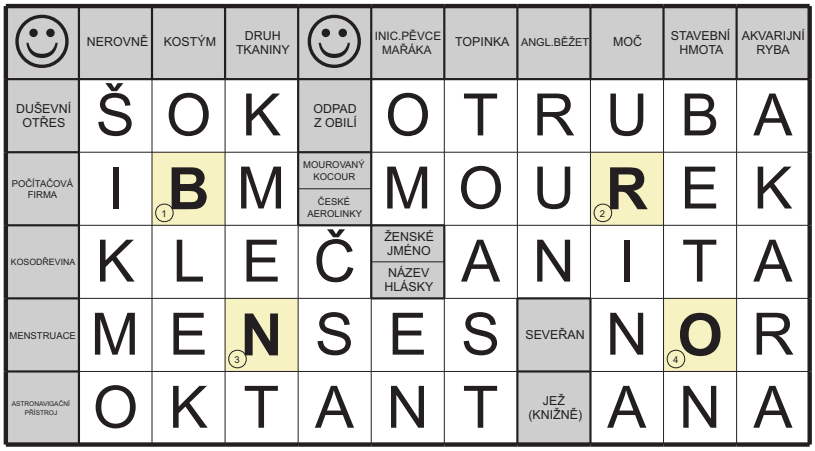
Tajenka je v obrazci vyznačená pozice nebo několik pozic, které po vyluštění křížovky odhalují luštiteli nějaké tajemství. Křížovku s takovouto souvislou tajenkou zobrazuje obrázek 2.2.

Tajenka však také může být nesouvislá. V takovém případě jsou označena jen některá políčka, která jsou zároveň očíslována posloupností přirozených čísel v pořadí, v jakém má být výsledná tajenka čtena. Křížovku s tímto typem tajenky ilustruje obrázek 2.3.

Počet políček, která jsou určena pro tajenku, by měl tvořit cca 12 % z celkového počtu políček, která luštitel vyplňuje [10].

2.2.5 Vpisované výrazy

Vpisované výrazy jsou slova nebo slovní spojení, která řešitel vyplňuje do *pozic* na základě informace z přiřazeného *legendového výrazu*.



Obrázek 2.3: Ukázka švédské křížovky s rozptýlenou tajenkou

Z hlediska kvality křížovky je třeba dát si pozor na tzv. limitované výrazy. Takovými výrazy jsou: zkratky, značky, kódy, cizojazyčné (překladové) výrazy, cizí zeměpisná jména, nepříliš známá vlastní jména útvarů v kosmu a jména etnických skupin, cizí rodná jména. Množství těchto limitovaných výrazů by nemělo přesáhnout 20 % obsahu křížovky.

2.2.6 Pomůcka

Pomůcka je zvláštním druhem políčka, které jako jediné může být tvořeno sloučením několika jiných políček. Do pomůcky může autor křížovky doplnit několik výrazů, které jsou přímo v řešení obsaženy a autor je považuje za velmi obtížné. Uvedením výrazu v pomůcce se snižuje obtížnost a zvyšuje luštitelnost křížovky.

2.2.7 Políčko symbol

Políčko symbol je do křížovky zaváděno z estetických důvodů. Jeho použitím se zachovává symetričnost obrazce. Na obrázku 2.3 můžeme vidět políčko tohoto typu v prvním řádku v prvním a pátém sloupci.

Do tohoto políčka se umísťuje dekorace tvořená grafickým symbolem nebo lze prostor v těchto políčkách použít pro logo či název časopisu, vydavatele apod.

Kapitola 3

Automatické generování

Protože po křížovkách je velká poptávka, zabývá se tvorbou křížovek množství autorů. Ti si za vytvoření křížovky obvykle účtují 1,50 až 4,00 Kč za každé políčko křížovky. Dodací doba se pak pohybuje mezi jedním až sedmi dny. Jednoduchým vynásobením lze zjistit, že vytvoření křížovky velikosti stránky A4 (20×28 políček) stojí 840,- až 2240,- Kč. Křížovky jsou tedy poměrně drahou komoditou. Proto je výhodné použít nástroj pro automatické generování křížovek, který zvládne křížovky generovat rychle a za výrazně nižší náklady. Programů, které automatické generování křížovek umožňují, existuje po světě několik, ale bohužel díky podpoře české abecedy a diakritických symbolů nebo dvojhlásky „CH“ lze v našich podmínkách použít jen některé. V potaz je také třeba brát kvalitu vygenerovaných křížovek (viz kapitola 3.4). Některé programy umožňují vytvářet kvalitnější křížovky než jiné, přesto se však křížovka vygenerovaná počítačem svou kvalitou málokdy dokáže vyrovnat křížovce, kterou vytvořil člověk.

Automatické generování křížovek počítačem využívá především „hrubé síly“, kdy je snaha postupně doplňovat písmena do políček nebo slova do pozic tak, aby se podařilo zaplnit všechna volná políčka v křížovce a zároveň, aby byly dodrženy požadavky na křížení slov a další pravidla, která musí křížovka splňovat.

3.1 Velikost stavového prostoru

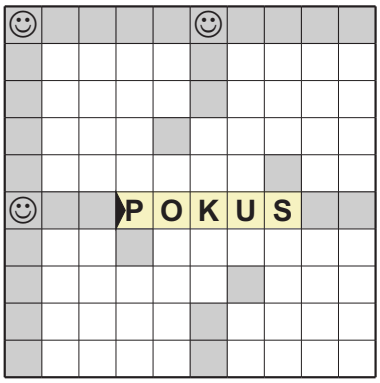
Automatické generování křížovek je výpočetně velmi náročné i přesto, že výkon současných počítačů neustále roste. Stavový prostor takové křížovky je totiž značně rozsáhlý. Generování křížovky lze klasifikovat jako NP-úplný problém [5].

Pokusme se nyní pro ilustraci provést odhad velikosti stavového prostoru na základě reálného příkladu. Nejprve se podívejme na jednoduchou křížovku o rozměru 10×10 políček (viz obrázek 3.1).

V textu této práce se s odkazem na tuto křížovku budeme setkávat i nadále, protože je dostatečně snadná pro ilustraci různých úkonů prováděných v křížovce.

Pokud vezmeme v potaz pouze českou abecedu, která má 42 písmen [14] a víme, že v křížovce na obrázku 3.1 je 64 volných políček, můžeme pomocí vzorce 3.1, kde za n dosadíme počet písmen abecedy a za k počet volných políček v křížovce, spočítat, že absolutní počet všech možných vyplnění políček této křížovky je přibližně $7,72 \cdot 10^{103}$.

$$n^k \tag{3.1}$$



Obrázek 3.1: Jednoduché rozložení křížovky 10×10 políček

Délka slova	Počet slov v křížovce v horizontálním směru	Počet slov ve slovníku
2	2	506
3	2	2050
4	8	4968
5	2	7338
6	2	8110

Tabulka 3.1: Počet slov v křížovce na obrázku 3.1

Značné množství těchto kombinací je však neplatných, protože netvoří žádná smysluplná slova. Zkusme se proto nyní zaměřit na počet možných stavů křížovky, jako na počet možností doplnění slov ze slovníku v jednom směru (vertikálním nebo horizontálním).

Hodnoty počtů slov jednotlivých délek ve slovníku v tomto příkladu (tabulka 3.1) jsou získány z reálného křížovkářského slovníku, který obsahuje necelých 40 000 slov.

Pokud označíme w množinu pozic pro slova v konkrétní křížovce a u množinu slov daných slovníkem a pokud budeme mít funkci $f_d(a, b)$, která z množiny a vybere všechna slova délky b, pak můžeme pomocí rovnice 3.2 spočítat teoretickou velikost stavového prostoru v případě, že bychom jednotlivé pozice vyplňovali na základě výrazů ze slovníku a umožnili bychom opakování stejného výrazu v jedné křížovce. To je však z hlediska kritérií správné křížovky nepřípustné (viz. [9] a [10]). Rovnice 3.3 se pokouší tento nedostatek odstranit.

$$\prod_{i=min(|w|)}^{max(|w|)} |f_d(u,i)|^{|f_d(w,i)|} \tag{3.2}$$

$$\prod_{i=min(|w|)}^{max(|w|)} \binom{|f_d(u,i)|}{|f_d(w,i)|} \tag{3.3}$$

Pokud bychom tedy umožnili opakování výrazů, byl by počet možných stavů pro křížovku na obrázku 3.1 přibližně $1,45 \cdot 10^{57}$. Bez možnosti opakování výrazů by tento počet stavů byl přibližně $2,24 \cdot 10^{52}$. I přesto, že toto číslo je přibližně $10^{51}\times$ menší než výsledek výpočtu, který nezohledňoval slova jako celky, je velikost stavového prostoru tak obrovská, že i kdybychom byli schopni procházet desetitisíce možností za sekundu, trvalo by kompletní prohledání stavového prostoru dobu, kterou můžeme vzhledem ke stáří vesmíru (13,7 ± 0,2 miliardy let [15]) přirovnat k nekonečně dlouhé.

3.2 Prohledávání stavového prostoru

Z kapitoly 3.1 víme, že počet možností vyplnění polí je extrémně veliký. Naštěstí nemusíme prohledávat všechny stavy, jelikož naprostá většina z příkladu uvedeného výše je neplatná, protože není bráno v potaz, že slova se musí korektně křížit.

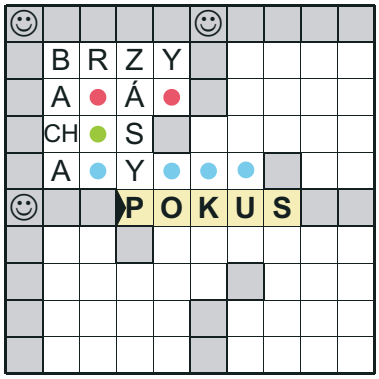
Idea generování křížovky je tedy taková, že se postupně doplňujeme slova do křížovky v určitém pořadí. V případě, že se dostaneme do stavu, ze kterého již nelze dále pokračovat, vrátíme se o vhodný počet kroků zpět. Hlavním problémem automatického generování křížovky je správně zvolit ono „určité pořadí“ doplňování a „vhodný počet“ kroků, o které se v případě neúspěchu vracíme nazpět. Jedná se tedy o využití backtrackingu (zpětného vyhledávání), které je založeno na procházení možných řešení do hloubky [7].

3.2.1 Naivní přístup

Mějme prohledávání do hloubky takové, že se doplňuje vždy křížící se slovo postupně od prvního písmena k poslednímu zleva doprava, resp. shora dolů a zároveň se bere ohled na nutnou podmínku korektního křížení se slov.

Tento způsob generování je asi první možností, která by každého, kdo by se nad tímto tématem zamýšlel, napadla. Jak si ale ukážeme dále, pro generování křížovek je zcela nevhodná. Pokus, který jsem v roce 2003 v počátcích vývoje provedl, ukázal, že vygenerování křížovky o rozměru 16×10 políček tímto způsobem trvalo více než 4 hodiny, což je ve srovnání s tím, jak dlouho trvá vymyslet takovouto křížovku člověku, naprosto nepřípustné.

Důvod, proč tento způsob generování není vhodný, je následující: Představme si, že generátor přesně dodržuje výše popsaný algoritmus. Pak se ale velmi často dostane do stavu uvedeném na obrázku 3.2.



Obrázek 3.2: Demonstrace problému při použití prohledávání do hloubky

V tomto stavu by se pokoušel nejprve korektně vyplnit políčka označená červeným kolečkem, poté zeleným, dále pak modrým. Problémem je, že v daný okamžik je zcela zbytečné zkoušet naprostou většinu kombinací pro červená a zelená pole, protože pro slovo s maskou `A-Y---` (modrá políčka) již existuje pouze jediný kandidát, a to slovo APYRIT. To však tento algoritmus nebere v potaz a zbytečně zkouší kombinace (pro zelené políčko dokonce opakovaně), které v zápětí vyloučí, kvůli nevhodnosti jediného kandidáta na slovo s modře označenými políčky.

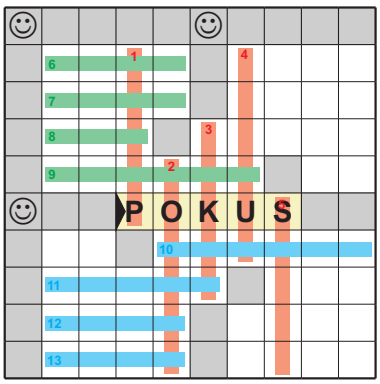
3.2.2 Generování metodou obalování slov

Metoda obalování slov byla použita v generátoru křížovek Crosswords - ITPro CZ řady 2.x. Tento způsob již je použitelný, další kapitola se však bude přesto zabývat zdokonalením této metody.

Idea obalování slov je taková, že zvolíme startovní slovo, od kterého začneme generovat. Toto slovo volíme obvykle jako nejdelší pozici v křížovce, kterým je zpravidla tajenka. Od startovní pozice poté postupně vyplňujeme všechna křížící se slova od začátku obalovaného slova do konce. Doplňování jednotlivých slov provádíme postupně po písmenech a při každém doplnění zkontrolujeme, jestli pro doplněnou pozici existuje vhodné křížící se slovo. V případě, že je v průběhu doplňování zjištěno, že vhodná kolmá maska neexistuje, je pro danou pozici poznamenáno, že na ní momentálně doplňované písmeno není vhodné. V praxi to tedy znamená, že pokud se do křížovky snažíme zapsat například slovo TÁTA a v průběhu zápisu zjistíme, že doplňování selhalo na čtvrté pozici, víme, že je zbytečné zkoušet doplňovat například slovo JANA, protože i přesto, že by doplnění prvních 3 písmen bylo úspěšné, na čtvrtém písmeně doplňování selže.

Vždy, když je do křížovky doplněno celé slovo, jsou všechny jeho křížící se pozice zařazeny *na konec* struktury typu fronta. Tím je zajištěno, že je vždy nejprve obalena celá pozice všemi křížícími se nedoplněnými slovy, a až poté je postoupeno na další úroveň zanoření.

Situaci ilustruje obrázek 3.3. Na něm vidíme, že jako počáteční pozice byla zvolena tajenka. Tuto pozici je nejprve třeba celou obalit, tzn. vyplnit všechny červeně označené pozice v pořadí na obrázku zapsaných čísel. V dalším zanoření se doplní zeleně označená slova, poté modrá. Všimněme si, že v modrém zanoření by měla být nejprve doplňována pozice, která je na obrázku označena zeleným číslem 9. Do fronty však není třeba vkládat pozice, které v ní již jsou. Při každém zápisu do fronty je tedy vhodné zkontrolovat, jestli již aktuálně přidávanou pozici neobsahuje a případně zápis duplicitní pozice ignorovat.



Obrázek 3.3: Generování metodou obalování slov

Použití této metody má relativně velké nároky na ukládané informace pro backtracking. Ukládat je třeba informace nejen o tom, jak vypadala křížovka v předchozí iteraci, ale i která slova byla již pro doplněné pozice vyzkoušena (aby se opakovaně netestovala slova, která již byla vyzkoušena a některé z jejich synovských zanoření selhalo). Pro jednotlivá políčka je třeba ukládat informaci o písmenech, která jsou pro případná další slova na určité pozice nevhodná (popsáno výše).

Metoda výběru další pozice	Doba generování	Počet zpětných navracení	Počet dopředných zanoření	Relativní počet navrácení
počet kandidátů	50 ms	9	47	19 %
nedopl. písmena	133 ms	127	284	44 %

Tabulka 3.2: Porovnání metod výběru následující pozice pro doplňování na křížovce na obrázku 3.1

3.2.3 Metoda nesouvislého doplňování s udržováním seznamu kandidátů

Metoda nesouvislého doplňování s udržováním seznamu kandidátů používá podobné principy jako metoda obalování slov. Hlavním rozdílem je však to, že slova nejsou doplňována postupně v pevně stanoveném pořadí.

Při využití této metody se ke křížovce nepřistupuje jako k dvourozměrné matici písmen, ale jako k seznamu pozic pro slova, který mimo jiné uchovává i informace o křížení slov. Díky tomu je možné pro každou pozici udržovat seznam kandidátů určených k doplnění. Generování pak probíhá takovým způsobem, že následující pozice pro vyplnění je vybrána na základě nějaké globální informace o všech pozicích v křížovce. Touto informací může být buď počet kandidátů nebo procentuální počet nedoplněných písmen pro danou pozici (poměr volná políčka v pozici / políček pozice). Pořadí vyplňování pozic se tak v průběhu generování mění podle toho, jaká jsou v křížovce momentálně vyplněna slova.

Z hlediska rychlosti generování je velmi výhodné, když pozice s menším množstvím kandidátů jsou vyzkoušeny dříve v hlubší úrovni zanoření. Porovnání ilustruje tabulka 3.2.

3.3 Navržený algoritmus generování

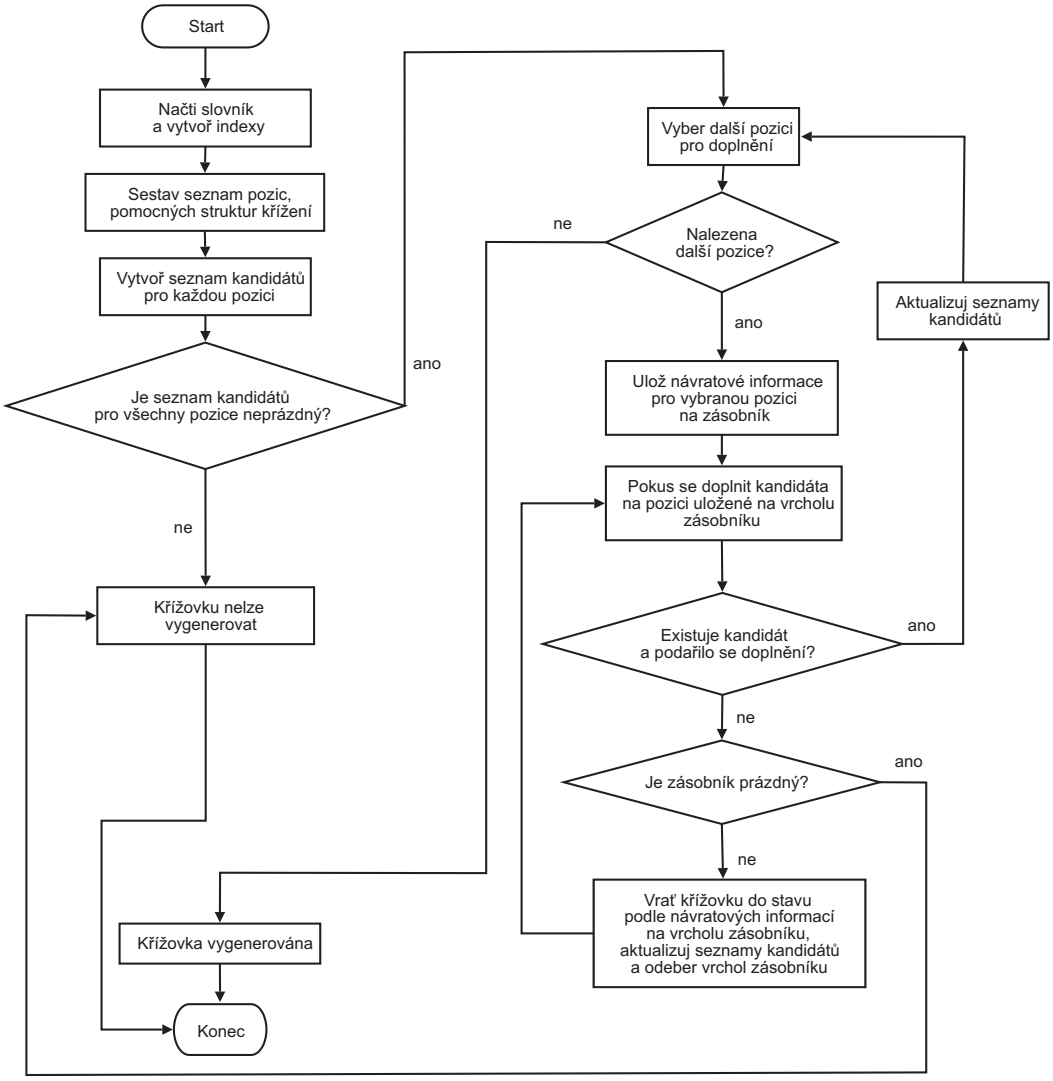
Generování křížovek metodou obalování slov, která byla popsána v kapitole 3.2.2 a implementována v generátorech křížovek Crosswords - ITPro CZ verze 2.x trpí řadou nedostatků a nevýhod. Například je třeba v každém cyklu kontrolovat počet volných políček, aby bylo detekováno, zda již není generování dokončeno. Ke křížovce se také přistupuje jako ke dvojrozměrné matici písmen, ve které je neustále potřeba detekovat hranice slov, číst masky pro vyhledávání atd. Je však třeba si uvědomit, že řada těchto nedostatků má i svoje opodstatnění. Tento algoritmus byl implementován v letech 2003 – 2004 a bylo vyžadováno, aby byl efektivně funkční na běžných domácích a kancelářských počítačích. Bylo tedy třeba úspornosti v ohledu paměťové náročnosti programu. V současné době se standardní charakteristika domácího počítače výrazně posunula, a proto je možné navrhnout algoritmus, který bude sice výrazně více paměťově náročný, ale zároveň bude efektivnější a bude umožňovat generovat daleko kvalitnější křížovky [10].

Dále je algoritmus třeba přizpůsobit faktu, že v současné době je zcela běžné, že počítač disponuje vícejádrovým procesorem. Je tedy třeba se zamyslet nad tím, jak úlohu paralelizovat a efektivně tím využít prostředky počítače.

Navržený algoritmus generování využívá metody nesouvislého doplňování s udržováním seznamu kandidátů, tak jak byl popsán v kapitole 3.2.3. Obecně by algoritmus šel naznačit vývojovým diagramem, který je uveden na obrázku 3.4.

Vzhledem k běžnému výskytu vícejádrových procesorů či víceprocesorových počítačů i mezi řadovými uživateli je vhodné tohoto potenciálu využít a celý algoritmus nebo alespoň jeho části paralelizovat.

I přes pokusy o paralelizaci celého algoritmu se mi tohoto nepodařilo dosáhnout. Parale-



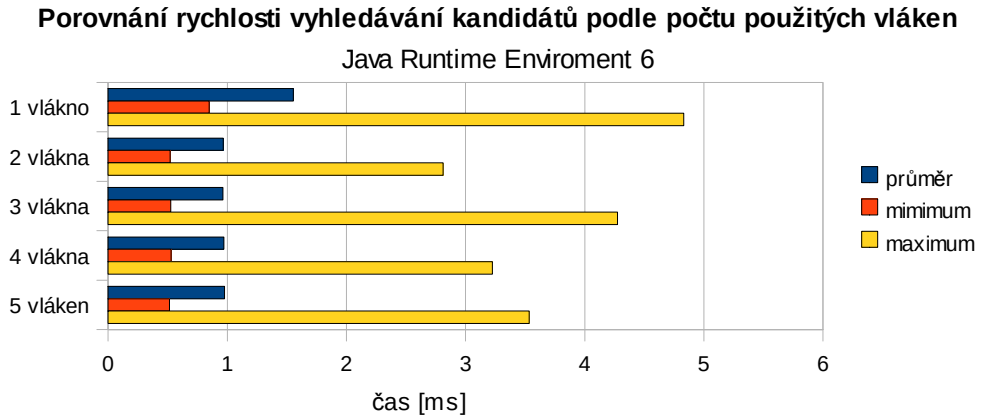
Obrázek 3.4: Vývojový diagram generátoru křížovek

lizace algoritmu je částečně v rozporu i s tradičním postupem ručního generování křížovek. Pokud si představíme dvě vlákna (vláken může být v praxi i více, ale v tomto příkladu si pro zjednodušení problému představujme pouze dvě vlákna), která se pokouší doplňovat vhodná slova do předem stanoveného tvaru, každé začínající z jiného místa, může se zdát, že jsme takto schopni celou operaci urychlit. Opak je však pravdou. V první řadě musíme v tomto přístupu k paralelizaci vyřešit problém kvalitní synchronizace. Problémem je především požadavek na kvalitní křížovku, který říká, že v křížovce se nesmí objevovat dvakrát to stejné slovo [10] [9]. Podobně tak i požadavky na obtížnost či výskyt limitovaných výrazů vyžadují množství sdílených informací, které jsou generovány oběma vlákny a pro sdílení vyžadují synchronní přístup, který zpomaluje činnost. Druhým, daleko závažnějším problémem, však je situace, ve které se obě generující vlákna dostanou na doplňování pozic, které se již vzájemně ovlivňují (tedy kříží, v horším případě dokonce překrývají). V této situaci, kdy jedno generující vlákno již přímo ovlivňuje možnosti stavového prostoru jiného vlákna, je třeba rozhodnout, které vlákno získá dominanci a které vlákno bude muset

ustoupit v případě, že je zjištěno, že ani jedno z generujících vláken již nemá další možnosti pokračování.

Při různých simulacích a pokusech jsem dospěl k závěru, že využití potenciálu vícevláknového generování z pohledu paralelního generování z více míst křížovky není výhodné. Vlákna totiž spolu buď soupeřila nebo jedno ustupovalo druhému, a tím přišel výsledek práce ustupujícího vlákna zcela vniveč. Výhody vyššího výkonu vícevláknové aplikace je tedy potřeba využít jinak. Popsaný algoritmus (obrázek 3.4) potřebuje ke své činnosti seznam kandidátů vhodných slov k doplnění pro každou pozici a elementární krok vyhledání vhodných kandidátů pro aktuálním krokem ovlivněné pozice je prováděn mnohokrát a zcela nezávisle. V generátoru tedy existuje pouze jedno řídící vlákno, které rozhoduje o doplnění slova i následující pozici, na které bude algoritmus v následující iteraci pokračovat. Další vlákna jsou využita pro vytváření seznamu kandidátů na dané pozice.

Vytvoření vlákna je časově náročnou operací. Praktické pokusy dokonce ukázaly, že vytvoření vlákna se na různých operačních systémech může lišit dokonce řádově. Vlákna je tedy pro tyto krátkodobé úkony (tj. operace probíhající v řádech do desítek milisekund) třeba recyklovat. K tomu slouží v programovacím jazyce Java™ třída `ExecutorService`, která umožňuje vytvořit vlákna, která budou opětovně využívána pro plnění nějaké činnosti (tříd implementujících rozhraní `Runnable`). Tato vlákna lze vytvořit pomocí volání `Executors.newFixedThreadPool(n)`, kde `n` je počet vláken, která mají být vytvořena. Otázkou je, kolik těchto vláken vytvořit. K dispozici jsem měl několik strojů, které disponují dvoujádrovým procesorem a na nich jsem chtěl experimentálně zjistit, jaké množství vláken je v tomto případě optimální vytvořit. Výsledky měření zachycuje graf na obrázku 3.5. Měření těchto časů bylo provedeno pomocí knihovny JETM[1].

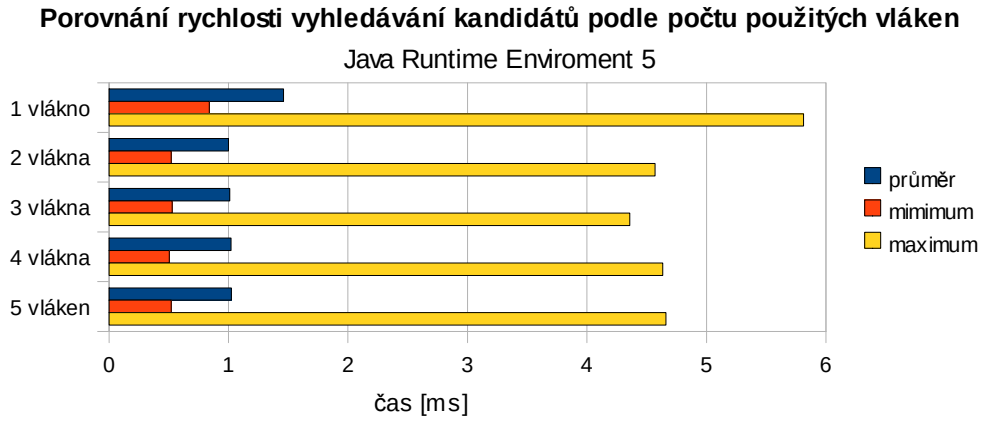


Obrázek 3.5: Porovnání rychlosti vyhledávání kandidátů podle počtu vláken - JRE 6

Při této přiležitosti jsem zároveň provedl porovnání rychlosti činnosti vícevláknových aplikací v Java™ Runtime Enviroment 5 a Java™ Runtime Enviroment 6, u kterého je proklamována optimalizace a vyšší výkon než v JRE5. Výsledky měření na obrázku 3.6 ukazují, že mírný rychlostní rozdíl mezi JRE5 a JRE6 existuje, není však příliš markantní.

Z obrázků 3.5 a 3.6 vyplývá, že nejvhodnější počet vláken pro tento úkon je roven počtu dostupných procesorů + 1. Důvodem je nedeterministické přepínání vláken operačním systémem. Vlákna, která mohou být aktivní, běží, ale v průběhu přepínání vláken dochází

[1] Java™ Execution Time Measurement Library - `http://jetm.void.fm/`



Obrázek 3.6: Porovnání rychlosti vyhledávání kandidátů podle počtu vláken - JRE 5

ještě k přípravě dalšího vlákna ke spuštění.

3.3.1 Vyhledávání ve slovníku

Z diagramu na obrázku 3.4 vidíme, že hledání a aktualizace seznamu kandidátů pro jednotlivé pozice je důležitou a často prováděnou činností algoritmu. Proto je třeba, aby tento podproblém byl řešen v co nejkratším čase. Následující podkapitoly se zabývají optimalizací tohoto problému.

Pro potřeby generování křížovek je třeba ve slovníku vyhledávat slovo nebo seznam slov, které odpovídají vyhledávací masce. Vyhledávací maskou rozumíme řetězec pevné délky, který na některých svých pozicích obsahuje speciální znak, který říká, že na dané pozici může být libovolné písmeno (např. vyhledávací masce `-ES` vyhovují slova PES, LES, VES...).

Protože délka hledaného slova je maskou pevně určena, je výhodné mít slovník rozdělen na jednotlivé sekvence slov podle délky. Díky tomu se pak prohledávají pouze slova požadované délky. Samotné dohledání slov odpovídajících konkrétní masce lze provést několika způsoby.

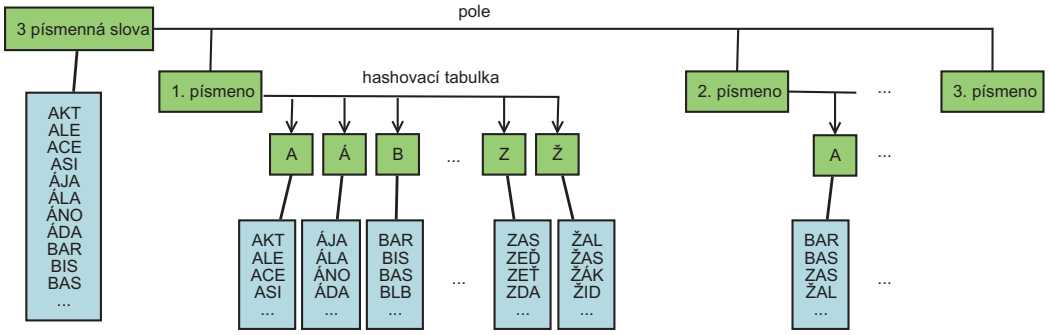
Sekvenční vyhledávání (SAM)

Každé slovo odpovídající délky je písmeno po písmenu porovnáno s maskou. V případě první neshody je porovnávání okamžitě ukončeno a slovo označeno jako nevhodné. Hlavní nevýhoda této metody vyhledávání je, že je vždy třeba projít všechna slova dané délky. Představu o počtu slov dané délky ve slovníku nám může poskytnout tabulka 3.1 uvedená na straně 9.

Index sekvenční vyhledávání (ISAM)

Při načtení slovníku je možné vytvořit kompletní písmenný index pro každou pozici a každé písmeno, tak jak zobrazuje ilustrace na obrázku 3.7

Index pro každou délku slov obsahuje pole, které má stejný počet položek jako je délka slov, kterých se index týká (pro trojpísmenná slova má tedy pole 3 položky). Každá položka se pak odkazuje na písmenný index, který je realizován pomocí asociativního pole



Obrázek 3.7: Ilustrace použitého slovníkového indexu

(resp. hashovací tabulky). Z toho důvodu nemusí existovat položka písmenného indexu pro všechny možné znaky, které se ve slovníku vyskytují. Pokud tedy neexistuje trojpísmenné slovo, které má na druhém místě písmeno „Q“, nebude položka Q písmenného indexu pro druhou pozici existovat a zabírat paměť.

S využitím tohoto indexu můžeme získat seznam slov s konkrétním písmenem na konkrétní pozici v konstantním čase (třída složitosti $O(1)$). Při vyhledávání slov odpovídajících masce je zjištěn počet slov v seznamu indexu pro každé v masce zadané písmeno. Z těchto nalezených seznamů je vybrán ten nejkratší a v něm je provedeno sekvenční vyhledání způsobem, jaký je popsán v předchozí kapitole.

Pokud budeme uvažovat českou abecedu, která má 42 písmen [14] a budeme předpokládat rovnoměrné rozložení výskytu jednotlivých písmen na všech pozicích, můžeme výpočtem podle vzorce 3.4, kde s_p je velikost jedné položky písmenného indexu a s_d je velikost jedné položky délkového indexu, spočítat, že tento index by měl zabrat při použití slovníku s 40 tisíci slovy přibližně 6 MB operační paměti. Taková spotřeba paměti za cenu tak výrazného zrychlení vyhledávání ve slovníku (viz dále) je při současných velikostech operačních pamětí osobních počítačů přijatelná.

$$\sum_{i=min(|w|)}^{max(|w|)} |f_d(u,i)| \cdot (i \cdot s_p + s_d) \tag{3.4}$$

Praktické měření však ukazuje, že index vytvořený nad konkrétním slovníkem, který je zmiňován výše, zabírá necelé 2 MB. To je způsobeno tím, že výskyt písmen rozhodně není rovnoměrný a řada položek indexu je nevyužita.

Indexové vyhledávání (IAM)

Pokud máme vytvořený index takový, jaký byl ilustrován na obrázku 3.7, můžeme vytvořit algoritmus vyhledávání, který bude založen pouze na množinových operacích nad položkami indexu. Myšlenka je taková, že projdeme v masce všechna pevně stanovená písmena a položky indexu odpovídající těmto písmenům a pozicím. Při prvním výskytu slova označíme v asociativním poli (hashovací tabulce) položku tohoto slova číslem jedna. Při dalších výskytech vždy provedeme inkrementaci této hodnoty. V závěru algoritmu pak sekvenčně projdeme přes nejkratší položku indexu a slova, která mají jako číselnou hodnotu počtu výskytů rovnou počtu pevně stanovených písmen v masce, vrátíme jako výsledek vyhledávání.

Metoda vyhledávání	Náhodné masky	Prázdné masky	Chybějící jedno písmeno	Jedno pevně stanovené písmeno
SAM	381 μs	464 μs	343 μs	372 μs
ISAM	12 μs	19 μs	13 μs	15 μs
IAM	26 μs	18 μs	171 μs	27 μs

Tabulka 3.3: Porovnání průměrné rychlosti vyhledávacích metod na různých typech masek

Metoda vyhledávání	Náhodné masky	Prázdné masky	Chybějící jedno písmeno	Jedno pevně stanovené písmeno
ISAM	12 μs	19 μs	13 μs	15 μs
ISAM mask	11 μs	19 μs	11 μs	13 μs

Tabulka 3.4: Porovnání průměrné rychlosti vyhledávací metody ISAM a ISAM se zkompilovanou maskou

Porovnání rychlostí vyhledávání

Praktické testy však ukazují, že indexový způsob vyhledávání vzhledem k režii při počítání výskytů slov není rychlejší než metoda ISAM. Tabulka 3.3 ukazuje porovnání rychlosti jednotlivých metod vyhledávání na různých typech masek.

Podle provedených testů se nejvýhodnější jeví použití index-sekvenčního vyhledávání (ISAM). I to však lze ještě částečně zrychlit. Pokud používáme textovou reprezentaci masky a vyhledáváme masky, které obsahují jen několik málo písmen, dochází k nadbytečnému a opakovanému kontrolování pozic, na kterých může být libovolný znak. Tuto redundantní činnost lze eliminovat předzpracováním masky. Předzpracování provedeme tak, že do pole uložíme pouze pevně stanovená písmena a jejich pozice. Při kontrole, zda slovo odpovídá masce, se pak porovnávají pouze ty pozice, na kterých se nalézá pevně stanovené písmeno. Výsledky urychlení zobrazuje tabulka 3.4. Urychlení sice není výrazné, přesto je implementace tak snadná, že se vyplatí předzpracovaných masek využívat.

V porovnání s klasickým sekvenčním vyhledáváním jsme dosáhli v průměru 35 násobného urychlení.

3.4 Nedostatky automaticky generovaných křížovek

Přestože automatické generování křížovek počítačem má výhody, mezi které patří hlavně nízké náklady a rychlost zhotovení, má tento způsob tvorby křížovek i řadu nevýhod. Následující dvě podkapitoly popisují hlavní nedostatky automaticky generovaných křížovek. Zároveň se také zabývají řešeními, která byla k eliminaci těchto problémů použita v implementaci tak, aby se kvalita vytvářených křížovek alespoň částečně přiblížila křížovkám vytvořeným člověkem.

3.4.1 Obtížnost

Hlavním cílem automatických generátorů křížovek je dokončit generování v konečném, přijatelně dlouhém čase. Protože ale počítač doplňovaným slovům nerozumí, často se stane, že se vytvořená křížovka skládá z velkého množství odlehlých geografických názvů, cizích slov a podobně. Tyto výrazy navíc patří do skupiny limitovaných výrazů (viz kapitola 2.2.5) a počet použití těchto výrazů by měl být v křížovce omezen.

Ani autor, který křížovky vytváří ručně, se nemůže zcela vyhnout použití složitých či limitovaných výrazů, pokud ale křížovka obsahuje takových výrazů příliš, jak tomu často bývá u křížovek generovaných počítačem, je luštění takové křížovky nepříjemné i pro zaryté luštitele. Proto je třeba, aby generovací algoritmus disponoval alespoň částečnou informací o obtížnosti daného slova. Toho lze docílit ohodnocením všech slov ve slovníku podle číselného klíče.

- 1 = velmi snadné – doplní i dítě 1. stupně ZŠ, obvykle dětem známá synonyma
- 2 = snadné – jasné výrazy odpovídající znalostem absolventa ZŠ
- 3 = mírně obtížné – výrazy odpovídající znalostem absolventa SŠ, často používané křížovkářské výrazy
- 4 = obtížné – křížovkářské záludnosti, obvykle známé notorickým luštitelům
- 5 = velmi obtížné – odpovídá znalostem odborníka v daném oboru

Každé slovo, které patří do skupiny limitovaných výrazů, by mělo být ve slovníku označeno, aby bylo možno dodržet maximálně 20 % obsahu limitovaných výrazů.

Nastavení obtížnosti křížovky musí být v rukou uživatele programu a musí být možné nastavit nejen maximální, ale i minimální obtížnost. Nastavení by mělo být možné pomocí koeficientů, které budou korespondovat s výsledným počtem použitých výrazů z jednotlivých obtížnostních kategorií. Je však nutné, aby koeficienty určovaly omezující podmínky s částečnou neurčitostí. Z experimetů vyplývá, že pokud vyžadujeme po algoritmu přesné počty výrazů z jednotlivých obtížnostních kategorií, neúnosně tím zvyšujeme dobu generování.

Právě volba mezi dodržením požadavku zadané obtížnosti a přiměřenou dobou dokončení generování činí řešení problému obtížnosti vygenerované křížovky náročným. Pokud algoritmus příliš dbá požadavků na obtížnost, může se stát, že tyto požadavky jsou natolik omezující, že se doba generování protáhne na dobu neúměrnou (za přijatelnou dobu generování lze považovat čas do 5 minut). V opačném případě, kdy je naším hlavním cílem křížovku dokončit co nejdříve, utrpí kvalita křížovky (časté výskyty limitovaných výrazů a přehnaná obtížnost).

Implementace umožňuje mít pro každé slovo několik legendových výrazů s různou obtížností. Přiřazený legendový výraz však v průběhu generování není ještě znám, proto je třeba obtížnost výrazu řešit obecněji. K dispozici tak máme nejnižší a nejvyšší obtížnost přiřazenou k jednotlivým slovům. Z těchto hodnot pak počítáme aritmetický průměr (vznikne interval $\langle \overline{diff_{min}}; \overline{diff_{max}} \rangle$) a na základě uživatelem zadaného intervalu $\langle u_{min}; u_{max} \rangle$ se snažíme vyloučit ty kandidáty, kteří by porušili platnost výrazu 3.5.

$$\langle \overline{diff_{min}}; \overline{diff_{max}} \rangle \cap \langle u_{min}; u_{max} \rangle \neq \emptyset \tag{3.5}$$

Volitelně lze zapnout i funkci, která se snaží alespoň částečně zahrnout do aktuálního intervalu $\langle \overline{diff_{min}}; \overline{diff_{max}} \rangle$ předpoklad budoucnosti. Vzhledem k tomu, že seznam kandidátů pro všechny pozice je již udržován, není problém přidat průběžný výpočet průměrné minimální a maximální obtížnosti pro seznam kandidátů na každou pozici. Označme si tyto hodnoty jako $\overline{cand_{min}}$ a $\overline{cand_{max}}$. Z těchto průměrných hodnot obtížností pro jednotlivé pozice opět spočítáme aritmetický průměr. Do tohoto průměru však zahrneme pouze ty hodnoty, kde rozdíl minimální a maximální průměrné hodnoty obtížnosti je menší nebo roven

hodnotě $future_{coef}$, kterou lze pro generátor nastavit. Pokud je rozdíl minimální a maximální průměrné hodnoty obtížnosti kandidátů pro jednotlivé pozice veliký, nemůžeme vůbec rozhodnout, zda na dané pozici bude v průběhu generování dosazeno slovo jednoduché nebo obtížné. Naopak čím menší je rozdíl těchto hodnot, tím jistější je, jaká obtížnost bude pro danou pozici použita. K hodnotám $\overline{diff_{min}}$ a $\overline{diff_{max}}$ pak připočítáme získané hodnoty popsané v tomto odstavci váženým průměrem. Upravené hodnoty použijeme ve vzorci 3.5 a kandidáty, kteří by svým vložením porušili uvedený vztah, vylučujeme.

Dalším implementovaným postupem pro zajištění požadované obtížnosti je prioritní seznam kandidátů. Uživatel může specifikovat interval obtížnostních koeficientů, které mají být v průběhu generování vybírány přednostně. Za tímto účelem je v programu definováno rozhraní `PriorityFilter`, jehož implementace má za úkol rozhodnout, zda dané slovo spadá do prioritní skupiny. Třída `PriorityCandidateList` je pak samotnou implementací prioritního seznamu. Tato třída podporuje i metody pro náhodný výběr (popsáno v kapitole 3.4.2) tak, aby byly přednostně vybírány položky, které náleží do prioritního výběru.

Posledním algoritmem pro ovlivnění obtížnosti vygenerované křížovky je naprosté vyloučení některých kandidátů se stanovenou obtížností. Toto omezení je však třeba používat velmi opatrně. Pokud totiž nastavíme rozsah povolených obtížností na značně omezený počet tříd obtížnosti, je velmi pravděpodobné, že křížovku nebude možné vygenerovat z důvodu neexistujících kombinací křížení. Je třeba si uvědomit, že i zkušený autor křížovek je občas i v jednoduchých křížovkách nucen použít obtížný výraz.

Všechny výše uvedené funkce a hodnoty lze v programu nastavit pomocí uživatelského rozhraní. Aby však uživatel nebyl z velkého množství nastavení zmaten, jsou v programu vytvořeny také profily, které umožňují uchovávat několik konfigurací, u kterých se počítá, že budou používány nejčastěji.

3.4.2 Náhodnost a rozmanitost

Náhodnost a rozmanitost křížovek je velmi důležitým požadavkem v případě, že křížovek v programu generujeme velké množství. Protože datové struktury jsou obvykle procházeny systematicky (z části za to mohou i požadavky na obtížnost), dochází při opakovaném generování křížovek k častému použití stejných výrazů nebo dokonce stejných segmentů doplnění, což je nežádoucí jev.

Náhodnost lze v navrhovaném řešení zajistit relativně snadno. Slova slovníku se procházejí v pořadí, v jakém jsou uvedena v indexačních strukturách. Tyto struktury lze na počátku každého generování pseudonáhodně zamíchat. Druhým stupněm náhodnosti je procházení samotného seznamu kandidátů v pseudonáhodném pořadí a to jen s minimálním navýšením časové náročnosti algoritmu. Seznam kandidátů stačí mít uložen v lineárně vázaném seznamu. Výběr prvku pak má sice lineární časovou složitost $O(n)$, samotné odstranění prvku ze seznamu pak lze provést s konstantní časovou složitostí $O(1)$.

Náhodný výběr obtížnostní kategorie však musí řešit i komponenta starající se o výběr kandidátů s požadavkem na obtížnost (viz kapitola 3.4.1). Obtížnostní kategorii pro danou pozici totiž nelze vždy vybírat podle stejného klíče, protože by tato činnost výrazně snížila rozmanitost výsledných křížovek.

Kapitola 4

Problematika vykreslování křížovek

Program CwdStudio si mimo jiné klade za cíl, aby vytvořené křížovky mohly být přímo z programu publikovány bez potřeby dalších úprav v grafických editorech. Tento úkol má ale řadu problémů, které se na první pohled nemusí jevit jako zřejmé. Tato kapitola tyto problémy popisuje a zároveň se věnuje i postupu jejich řešení v programu CwdStudio, který je implementován v rámci této diplomové práce.

4.1 Obrysy políček

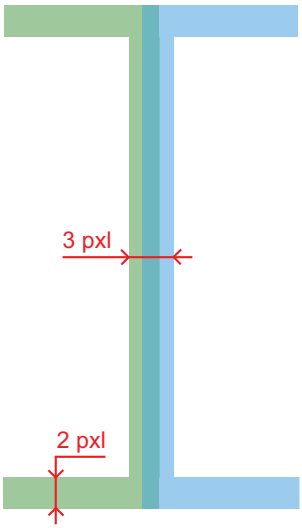
Vykreslování obrysů políček je skvělým příkladem problému, který není při počátečním zamyšlení nad problematikou zřejmý. Následující dvě podkapitoly popisují dva problémy, které při vykreslování obrysů a obsahů políček vznikají. Zároveň je uveden způsob řešení, jak se se vzniklými chybovými stavy vyrovnat.

4.1.1 Šířka čar obrysů

Prvním řešeným problémem je šířka čar na vykreslovacím plátně s celočíselnou aritmetikou. Vykreslujeme-li křížovku na grafické plátno, které je implementováno objektem `Graphics`, lze pro zobrazování používat pouze celočíslené souřadnice. Tento problém se týká i některých exportních formátů resp. jejich importu do některých méně pokročilých grafických editorů, které desetinné souřadnice ignorují (například EMF a program Corel Draw verze 9 a nižší). Šířka obrysů mezi dvěma buňkami pak může být větší než skutečná šířka orámování samotné buňky. Tuto situaci ilustruje obrázek 4.1. Nastavená šířka obrysu je v tomto případě 2 body, vzhledem k tomu, že 2 není liché číslo, může dojít z důvodů zaokrouhlování k vykreslení obrysů tak, že obrys každé buňky je vykreslen s posunutím o jeden bod. Přestože šířka obrysu buňky je nastavena na 2 body, mezi dvěma buňkami stejného typu vznikne obrys, který bude mít šířku 3 bodů.

Praktický důvod pro řešení tohoto problému však nalezeneme i u formátů a vykreslovacích pláten (instance `Graphics2D`), které zvládájí aritmetiku v plovoucí desetinné čárce. Pokud si uživatel přeje vyexportovanou křížovku dále upravovat v grafickém editoru, není příliš vhodné, aby dokument obsahoval zbytečně překrývající se duplicitní prvky. Takovéto prvky by mu zbytečně komplikovaly práci.

Řešením tohoto problému je vykreslovat čáry oddělující buňky pouze jedenkrát. Na základě porovnání typů dvou sousedních buněk je třeba rozhodnout, jaký obrys použít



Obrázek 4.1: Problém šířky obrysů na plátně s celočíselnou aritmetikou

	Prázdné	Tajenka	Legenda	Symbol	Pomůcka
Numerická hodnota pro porovnání	1	2	3	4	5

Tabulka 4.1: Prioritní koeficienty jednotlivých typů políček

a tento aplikovat pouze jednou. Rozhodnutí, který typ obrysu bude vykreslen, je provedeno tak, že pro každou třídu buněk je specifikována numerická hodnota určující prioritu a na základě těchto prioritních koeficientů je učiněno rozhodnutí. Tyto priority jsou uvedeny v tabulce 4.1.

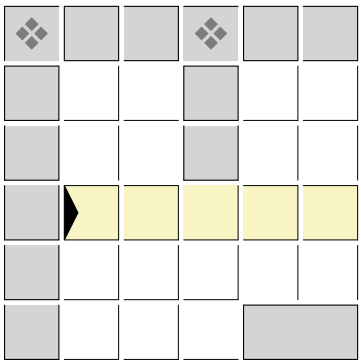
Jednotlivé čáry obrysu (tj. horní, pravý, dolní a levý) vykreslujeme na základě podmínky porovnávající prioritní koeficienty třídy buněk aktuálně vykreslovaného políčka a příslušné třídy buněk, do které náleží buňka sousední. Pokud příslušná sousední buňka není definována (tj. v křížovce je volný prostor), je obrys vykreslen vždy. Levá a horní čára obrysu je kreslena pouze v případě, že prioritní koeficient vykreslované buňky je vyšší než buňky sousední. Pravá a dolní linie obrysu je ale vykreslena nejen v případě nadřazenosti, ale i v případě rovnosti. Stejného (správného) efektu bychom dosáhli i jiným uspořádáním tak, že vždy musí být jedna linie z dvojice horní-dolní a levá-pravá vykreslována pouze v případě nadřazenosti a druhá v případě nadřazenosti i rovnosti.

Výsledné vykreslení obrysů ilustruje nejlépe obrázek 4.2.

4.1.2 Překrývání obrysů

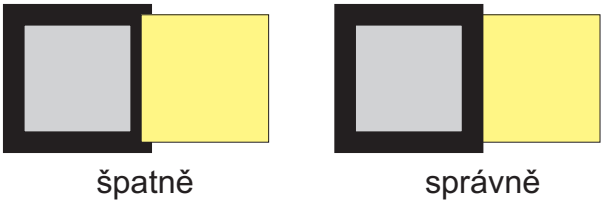
Dalším problémem, který je třeba při vykreslování křížovek řešit, je zajištění správného překrývání buněk, a tím i jejich obrysů. Samotný obsah buňky se nikdy nepřekrývá (nelze nastavit záporné mezery mezi buňkami), obrysy však přes sebe zasahovat mohou a v případě nulové mezery mezi buňkami zpravidla zasahují. Pokud bychom buňky vykreslovali v pořadí například podle pozice buňky (průchod dvourozměrným polem po řádcích), docházelo by k chybnému vykreslení tak, jak je vidět na obrázku 4.3 vlevo.

Pro předejití tohoto chybového stavu je vykreslení každého políčka rozděleno na 2 části: na vykreslení pozadí a obsahu a na vykreslení obrysu. Nejprve je vykresleno pozadí a obsahy



Obrázek 4.2: Řešení problému vykreslování obrysů buněk

všech políček. Protože se obsah políček nemůže překrývat, je možné provést vykreslení obsahů ve zcela libovolném pořadí. V druhé fázi jsou pak vykresleny obrysy. Zde však na pořadí záleží. Obrysy vykreslíme v pořadí podle typu buněk jak uvádí tabulka 4.1 od nejmenšího čísla po největší. Poté již dojde ke správnému vykreslení, které je naznačeno na obrázku 4.3 vpravo.



Obrázek 4.3: Špatné a správné překrývání obrysů

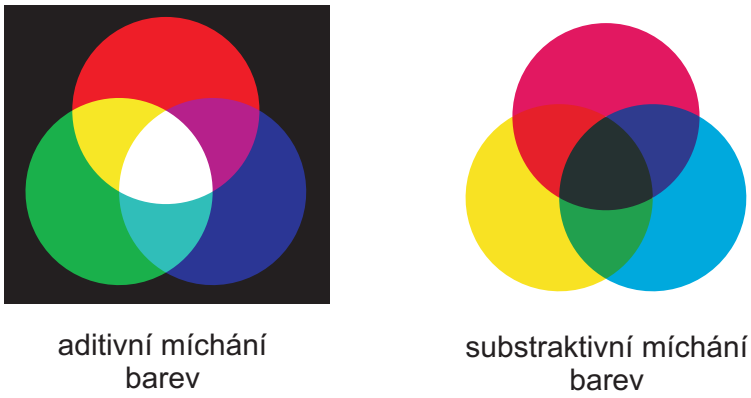
4.2 Podpora barevného modelu CMYK

Pokud provádíme předtiskovou přípravu nějaké tiskoviny, obvykle pracujeme s barevným modelem CMYK, který na rozdíl od obvykle používaného barevného modelu RGB, který skládá barvu aditivně ze tří složek červená, zelená a modrá, míchá barvy substraktivně ze složek čtyř. Rozdíl mezi aditivním a substraktivním mícháním barev ilustruje obrázek 4.4.

Při substraktivním míchání barev od sebe barvy v podstatě odečítáme, tedy omezujeme barevné spektrum, které se odráží od povrchu materiálu. Barevný model CMYK se používá především u reprodukčních zařízení, které barvy tvoří mícháním pigmentů (např. inkoustová tiskárna). Model obsahuje čtyři základní barvy:

- azurovou (**C**yan);
- purpurovou (**M**agenta);
- žlutou (**Y**ellow);
- černou (blac**K**), označovanou také jako klíčovou (**K**ey).

V ideálním případě by byly postačující pouze první tři barvy (model CMY), jejichž substraktivním složením dohromady by měla vzniknout černá barva. Ve skutečnosti však při



Obrázek 4.4: Aditivní vs. substraktivní míchání barev

použití běžných barviv vzniká barva tmavě šedivá, a zároveň je na rozdíl od ostatních barev černá výrazně levnější, proto většina tiskových technik používá ještě čtvrtou černou barvu [13].

Podpora tohoto barevného modelu je značná konkureční výhoda oproti podobným programům. Pro grafický export do vektorových formátů využívá program knihovnu FreeHEP VectorGraphics, kterou lze získat pod licencí LGPL na adrese `http://java.freehep.org/vectorgraphics/`. Tato knihovna však nemá zcela žádnou podporu pro barvy zadané jinými složkami než R, G a B[1]. Na základě studie zdrojových kódů této knihovny jsem zjistil, že barvy jsou fixně vypisovány vždy v barevném modelu RGB a to i v případě, že byly zadány jiným barevným modelem. Tzn. barevný model (instance objektu `ColorSpace`) je knihovnou zcela ignorován. Knihovnu tedy bylo třeba rozšířit a podporu barev se složkami C, M, Y a K[2] implementovat. Řešení tohoto problému však není až tak jednoduché, neboť licenční podmínky stanovují, že v případě úpravy zdrojových kódů musí být zachováno původní aplikační rozhraní. Následující podkapitoly popisují vyřešení zmíněného problému z pohledu programovacího jazyka Java™ i jednotlivých grafických formátů a jejich implementace v knihovně FreeHEP.

4.2.1 CMYK v Javě

Pro reprezentaci barvy existuje v Javě třída `java.awt.Color`. Tato třída zapouzdřuje barvy ve výchozím barevném modelu sRGB. Obsahuje však také podporu pro barvy vyjádřené jinými barevnými modely.

Barevný model pak reprezentuje implementace potomka abstraktní třídy `ColorSpace`. Pro podporu CMYK barevného modelu je možné vytvořit třídu odvozenou od `ColorSpace`, která bude v metodách `toRGB()` a `fromRGB()` převádět barvy podle vzorců 4.1 a 4.2 a případné hodnoty mimo rozsah 0.0 - 1.0 bude normalizovat [4].

$$\begin{aligned} k_1 &= 1 - CMYK_k \\ RGB &= (k_1 \cdot (1 - CMYK_c), k_1 \cdot (1 - CMYK_m), k_1 \cdot (1 - CMYK_y)) \end{aligned} \tag{4.1}$$

[1] červená, zelená a modrá
[2] azurová, purpurová, žlutá a černá

$$
\begin{aligned}
c &= 1 - RGB_r \\
m &= 1 - RGB_g \\
y &= 1 - RGB_b \\
k &= min(c, m, y) \\
CMYK &= \begin{cases} \left(\frac{c-k}{1-k}, \frac{m-k}{1-k}, \frac{y-k}{1-k}, k\right) & \text{pro } k \neq 1 \\ (0, 0, 0, k) & \text{pro } k = 1 \end{cases}
\end{aligned}
\tag{4.2}
$$

Jednodušší variantou než psát vlastní implementaci barevného prostoru je použít již hotovou třídu `SimpleCMYKColorSpace`, kterou nalezeneme v knihovně JAI[3]. Informace k této knihovně pro pokročilou práci s grafikou v Javě lze stáhnout a další informace získat na `http://java.sun.com/javase/technologies/desktop/media/jai/`. Uvedená třída kromě výše uvedených vzorců aplikuje také gamma korekce na barvu tak, aby převedené barvy odpovídaly barevnému modelu sRGB.

4.2.2 CMYK ve formátu PostScript v knihovně FreeHEP

PostScript je celosvětově používaný programovací jazyk, který je určen k popisu grafických dokumentů, který byl v roce 1985 vyvinut firmou Adobe Inc. Jazyk PostScript je zcela nezávislý na koncovém zařízení. V současné době je jazyk PostScript uznáván jako standardní jazyk pro tiskárny vyšší třídy.

Interpret PostScriptu je stavový stroj založený na zásobníkovém přístupu. Pro nastavení aktuálně používané barvy existují příkazy `setcmykcolor` a `setrgbcolor`. Tyto příkazy vyžadují 4 (resp. 3) parametry zadané číslem v plovoucí desetinné čárce v intervalu 0.0 - 1.0 [1].

I přesto, že knihovna FreeHEP VectorGraphics nepodporuje barvy v jiném barevném modelu než RGB, obsahuje úvodní prolog, který je zapisován do každého výstupního PostScript souboru, makra pro nastavení obrysové a výplňové CMYK barvy (makra `/k` a `/K`). Pro nastavení obrysové a výplňové RGB barvy jsou zde makra `/rg` a `/RG`.

Výstup do PostScriptu realizuje v knihovně FreeHEP třída `AbstractPSGraphics2D`, která má privátní metodu `setPSColor()`. Kdyby byla tato metoda pouze chráněná, bylo by přidání prodpory barevného modelu CMYK velmi snadné pomocí odvození vlastní třídy od této třídy a přetížením metody `setPSColor()`. Metoda je však privátní, a proto je třeba provést zásah do zdrojových kódů knihovny. Pro vyřešení tohoto problému jsem v rámci knihovny FreeHEP vytvořil rozhraní `PSColorWriter`, které je v základu implementováno přímo v knihovní třídě pro výstup PostScriptové grafiky. Zároveň však byla přidána metoda `setPSColorWriter()`, která umožňuje nastavit způsob, jakým bude barva zapsána do výsledného souboru. Uvedené rozhraní disponuje jedinou metodou `getColorString()` s parametry určujícími barvu, která má být vypsána a příznak o tom, zda se jedná o barvu obrysu nebo výplně. Pro výstup CMYK barev je rozhraní `PSColorWriter` implementováno uvnitř balíku programu CwdStudio. Podstatnou informací je také fakt, že ani PostScript ani PDF (které je od PostScriptu odvozeno) nepodporuje čísla zadaná v semilogaritmickém tvaru (např. 1e-3). Nelze proto použít výchozí metodu pro převedení čísla v plovoucí desetinné čárce na řetězec a je nutné nastavit explicitní převod, který zajistí, že číslo bude vypsáno vždy jako desetinné číslo.

[3] Java Advanced Imaging

4.2.3 CMYK ve formátu PDF v knihovně FreeHEP

PDF je dokumentový souborový formát, který si klade za cíl přenositelnost dokumentu bez ohledu na platformu a hardware. Tento formát byl vyvinut firmou Adobe a byl odvozen z jazyka PostScript. Protože řada prvků je s formátem PostScript shodná, je vhodné mít při čtení této podkapitoly přečtenou i podkapitolu 4.2.2. V červenci 2008 se PDF stalo oficiálním standardem ISO 32000-1:2008 [2].

PDF zavádí na rozdíl od PostScriptu volitelnou kompresi různých částí dokumentu, možnost šifrování a ochrany dokumentu, ale také možnost vkládání použitých fontů dovnitř dokumentu.

V knihovně FreeHEP je pro export do PDF vytvořena třída `PDFGraphics2D`. V této třídě existuje odkaz na otevřený datový proud, do kterého jsou průběžné zapisovány stavové informace. Datový proud je instancí třídy `PDFStream`. Tato třída již přímo CMYK podporuje a disponuje metodami pro zápis CMYKové barvy s využitím stejných maker, které byly popsány v předchozí kapitole. Bohužel ve třídě `PDFGraphics2D` se opět implicitně vypisuje barva pouze v barevném modelu RGB. Podobně jako bylo popsáno v kapitole 4.2.2 je i v tomto případě do knihovny přidáno rozhraní s v knihovně definovanou implementací, které slouží k zápisu barvy do výstupního datového proudu. Toto rozhraní je pojmenováno `PDFColorWriter` a disponuje jedinou metodou `writeColor()`, která má parametry určující barvu, příznak obrysu a odkaz na datový proud, do kterého má být barva zapsána. Podle zjištěného barevného modelu pak zavolá příslušnou metodu třídy `PDFStream` pro vypsání barvy do datového proudu.

4.2.4 CMYK ve formátu SVG v knihovně FreeHEP

SVG je vektorový grafický formát, který má v současné době potenciál velkého rozšiřování. Podporován je v internetových prohlížečích, ale i v řadě grafických editorů, které umožňují editaci vektorové grafiky. Je založen na dokumentovém objektovém modelu (DOM) v jazyce XML. Jazyk SVG je navržen W3C konsorciem, které se stará o jeho další vývoj a vylepšování. Aktuální verze SVG je verze 1.2.

Bohužel SVG 1.2 nepodporuje barvy zadané ve CMYK. Protože SVG přejímá některé zvyklosti od CSS a XHTML, barva se ve formátu SVG zadává buď pomocí hexadecimálního zápisu `#rrggbb` (případně doplněného o složku průhlednosti) nebo pomocí decimálního zápisu `rgb(r,g,b)`. Jediná možnost zadání barvy jiným způsobem je využití atributu `icc-color`. Podpora barev v barevném modelu CMYK pomocí atributu `icc-color` je však více méně teoretická. Vyzkoušel jsem řadu programů pro editaci vektorové grafiky[4], které umožňují načítání nebo ukládání grafiky ve formátu SVG, ale žádný z nich nedokázal do SVG barvy v barevném modelu CMYK uložit a ani načíst. Webové prohlížeče podporující zobrazení SVG grafiky, zobrazí barvy zadané pomocí ICC profilu jako černou. Praktické uplatnění využití atributu `icc-color` je tedy mizivé.

Použití atributu výše uvedeného však není příliš uspokojivé. Naštěstí existuje tabulka 147 názvů barev, které mají odpovídající hodnoty v barvách zadané v barevném modelu CMYK. Tuto tabulku nalezneme v příloze A. Podobným způsobem, jako bylo popsáno ve dvou předcházejících kapitolách, je do knihovny FreeHEP do třídy, která vytváří export do formátu SVG (třída `SVGGraphics2D`), přidána metoda pro nastavení implementace rozhraní `SVGColorWriter`. Vlastní implementace nejprve zkontroluje hodnoty složek CMYK (jedná-li se o barvu z barevného modelu CMYK), a pokud je vektor těchto hodnot nalezen

[4] Adobe Illustrator, Scribus, Corel Draw, InkScape

v převodní tabulce uvedené v příloze A, je zapsáno na místo kódu barvy její odpovídající jméno. V opačném případě je zapsána barva pomocí ICC profilu. Tímto řešením je kompatibilita s grafickými editory alespoň částečně zvýšena.

4.3 Vykreslování víceřádkového textu

Při vykreslování legendových výrazů do políček pro legendu je třeba vyřešit zalamování textu. Vyskytnout se však mohou i slova, která jsou tak dlouhá, že se nevejdou ani samotná na řádek. Tuto situaci lze vyřešit dvěma způsoby.

První možností je zmenšování velikosti písma. Zmenšování písma však snižuje čitelnost. Je třeba si uvědomit, že běžnou velikostí buňky je čtverec o hraně 1 cm. Písmo tedy lze zmenšit pouze nepatrně a i poté se může stát, že se text do buňky stále nevejde.

Druhou možností je dělení slov. Počítačové dělení slov se obvykle řeší pomocí slovníku vzorků. Inspiraci jsem nalezl v kancelářské sadě OpenOffice, která využívá slovníky vzorků, které byly vytvořeny pro sazební program LaTeX. Tyto slovníky fungují tak, že jednotlivé vzorky říkají, po jaké sekvenci písmen je vhodné slovo zalomit a po jaké to naopak vhodné není. Ve slovníku je na každém řádku uveden jeden vzorek.

Vzorky se skládají z:

- *nenumerických znaků* - Vyznačují sekvence písmen, které se v rámci slova skutečně vyhledávají.
- *znaku tečka „.“* - Značí hranice slova, tj. začátek nebo konec slova.
- *číslice* - Označuje index vhodnosti rozdělení slova v místě, na kterém se číslice nachází. Lichá čísla vyznačují místa, která jsou vhodná pro rozdělení. Sudá čísla vyznačují místa, kde je dělení nepřijatelné. Číslice vyšších numerických hodnot mají vyšší prioritu než čísla nižší.

Ukažme si praktický příklad strojového dělení slov. Uvažujme slovo „zúžit“. K tomuto slovu nalezneme v českém slovníku pro dělení slov následující vzorky: `i1t`, `2t.`, `2z2ú1`, `zú3ž`, `1ú`, `ú2ž`, `1ž`. První vzorek říká, že mezi písmeny I a T je vhodné slovo dělit, druhý vzorek s vyšším důrazem však říká, že pokud slovo končí písmenem T, nemělo by být toto odděleno. Aplikujeme-li všechny vzorky na zvolené slovo, získáme následující tvar `z2ú3ž0i2t`. Po aplikaci všech vzorků a získání slova v tomto tvaru, můžeme prohlásit, že slovo lze rozdělit pouze na těch pozicích, kde se vyskytuje liché číslo. Rozdělení slova pak vypadá takto: „zú-žit“, což odpovídá správnému rozdělení tohoto slova.

Protože dělení slov (ale i zmenšování velikosti písma) jsou operace, které jsou časově náročné, ale zároveň je není třeba provádět při každém vykreslení, je velmi výhodné zjištěné hodnoty předpočítat a upravovat pouze při změně textové informace nebo parametrů buňky. K ukládání předpočítaného stavu pro vykreslování víceřádkového textu nabízí třída `AbstractWordCell` (viz obrázek 5.2 na straně 29) podpůrné funkce.

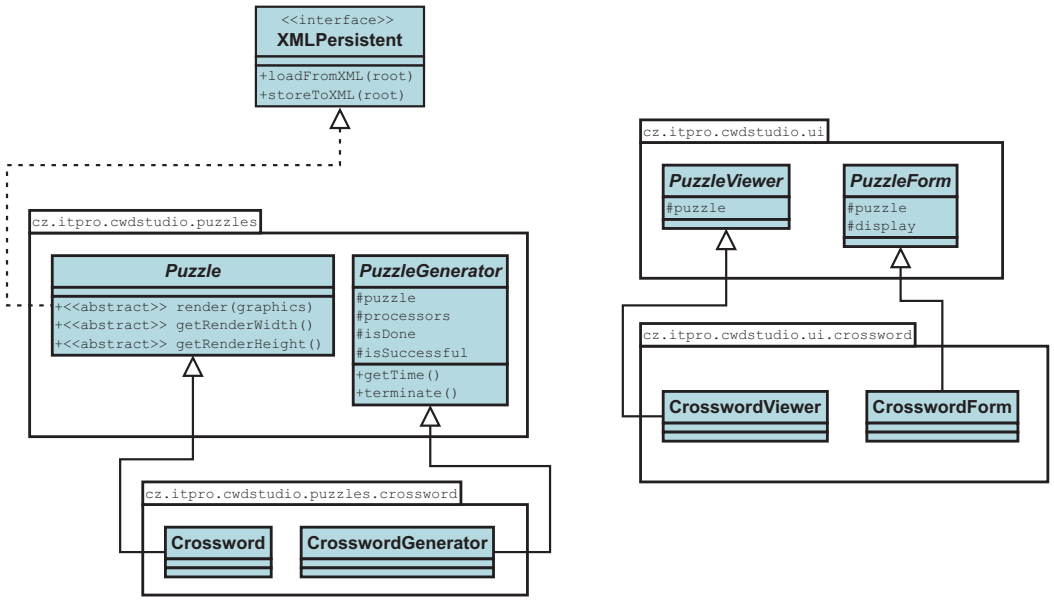
Kapitola 5

Architektura aplikace

Popisovat detailně celou strukturu aplikace by bylo zcela zbytečné, jelikož v současné době je tvořená více než dvěma sty tříd. Zajímavé však bude popsat jádro modelu, ve kterém se již od počátku vývoje počítá s budoucí podporou dalších typů hádanek (např. osmisměrky, klasické křížovky...). Zajímavou částí je také popis objektové reprezentace švédských křížovek a vnitřní struktury, která slouží k rychlému a bezproblémovému přístupu k jednotlivým pozicím v křížovce. V závěru této kapitoly je také naznačena struktura slovníku, která umožňuje implemetovat vyhledávací algoritmy, které jsou popsány v kapitole 3.3.1.

5.1 Jádro aplikace

Struktura jádra aplikace je zobrazována na obrázku 5.1. Na tomto obrázku je možné si všimnout logického rozčlenění na balíky. Balík `cz.itpro.cwdstudio.puzzles` zastřešuje datovou reprezentaci různých typů hádanek, které program podporuje, metody pro jejich vykreslení a třídy starající se o automatické vygenerování hádanky.



Obrázek 5.1: Jádro modelu aplikace CwdStudio

Pro jednotlivé typy hádanek je bázovou třídou abstraktní třída `Puzzle`. Ta poskytuje abstraktní metodu `render()`, která slouží k vykreslení hádanky. Tato metoda má pouze jeden argument, který identifikuje plátno (instanci třídy `Graphics`), na které má být hádanka vykreslena. Velikost vykreslení je pevně dána. Rozměr vykreslení je určen hodnotami, které vrací metody `getRenderWidth()` a `getRenderHeight()`. Chceme-li vykreslit hádanku v jiném rozměru, je třeba provést změnu vykreslovacího měřítka plátna pomocí metody `scale()`. Pro zjednodušení implementuje třída `Puzzle` i metodu pro vykrelení hádanky v zadané velikosti. Ta spočítá měřítko, které je třeba na plátno aplikovat, aby výsledná velikost vykreslení odpovídala požadavku. Implementace následníků třídy `Puzzle`, musí povinně implementovat rozhraní `XMLPersistent`, jehož metody slouží k uložení a načtení hádanky do XML souboru.

Pro manipulaci s XML soubory se v celém projektu používá knihovny JDOM[1], se kterou je sestavování a procházení objektového modelu dokumentu[2] velmi snadné. Koncept rozhraní `XMLPersistent` je takový, že metodám předá vždy jen instance třídy `Node`, která reprezentuje uzel v obecném n-árním XML stromu a metody buď připíší potřebné atributy reprezentující hodnoty, které mají být perzistentní, k uzlu nebo z atributů, kterými uzel disponuje, načtou hodnoty do zadané instance třídy, kterou načítáme. Díky tomuto přístupu je velmi snadné kombinovat perzistentní reprezentace jednotlivých tříd do různých kolekcí. Ku příkladu grafický styl křížovky tak může být bez problémů součástí souboru s křížovkou, ale může být uložen i zcela samostatně v souboru uchovávajícím předvolené styly.

Do XML souboru je někdy třeba uložit také nějaká binární data. Toho se využívá například při uložení náhledů hádanek do souboru s dokumentem. Díky tomu je při otevírání souboru možno velmi rychle zobrazit malý náhled křížovky, protože tento je již předpočítán a vykreslen v souboru. Binární data jsou převedena do kódování *Base64*, které ke kódování dat používá pouze 64 znaků (písmena malé a velké anglické abecedy, číslice, znaky „/“ a „+“ a znak „=“ pro zarovnávání) [6]. Kódování *Base64* funguje tak, že trojice 8 bitových binárních znaků jsou převedeny do reprezentace ve dvojkové soustavě. Tento řetězec je rozdělen na čtyři 6 bitové podřetězce a ty jsou vyjádřeny jednou z 64 hodnot. Velikost zakódovaných dat odpovídá $\frac{4}{3}$ velikosti původních dat.

Pro generátory jednotlivých typů hádanek je vytvořena bázová třída `PuzzleGenerator`. Instance tříd následníků třídy `PuzzleGenerator` implemetují rozhraní `Runnable`, které slouží ke spuštění úkolu v separátním vlákně. Aby bylo možné průběžně sledovat a ovládat stav generování obecné hádanky, jsou zavedeny metody pro předčasné ukončení generování (`terminate()`), pro zjištění uběhlého času generování (`getTime()`) a další metody pro získání parametrů, které určují, zda již bylo generování dokončeno a s jakým výsledkem. Průběh generování je potřeba také vizualizovat, údaje pro vizualizaci jsou získávány z vizualizačního vlákna metodou dotazování (*polling*). Ukončení generování (ať už úspěšné či nikoliv) je oznámeno vytvořením asynchronní události. Vizualizaci totiž není třeba provádět příliš často a zbytečně tím spotřebovávat procesorový čas (stačí i obnovování s frekvencí 1 Hz). Pokud by i v tomto případě bylo použito událostního přístupu, docházelo by k plýtvání výkonem počítače. Na případné zásadní změny stavu je však třeba reagovat okamžitě, proto je pro tyto změny využit událostní přístup, zatímco vizualizační data jsou mezi vlákny předávána dotazováním.

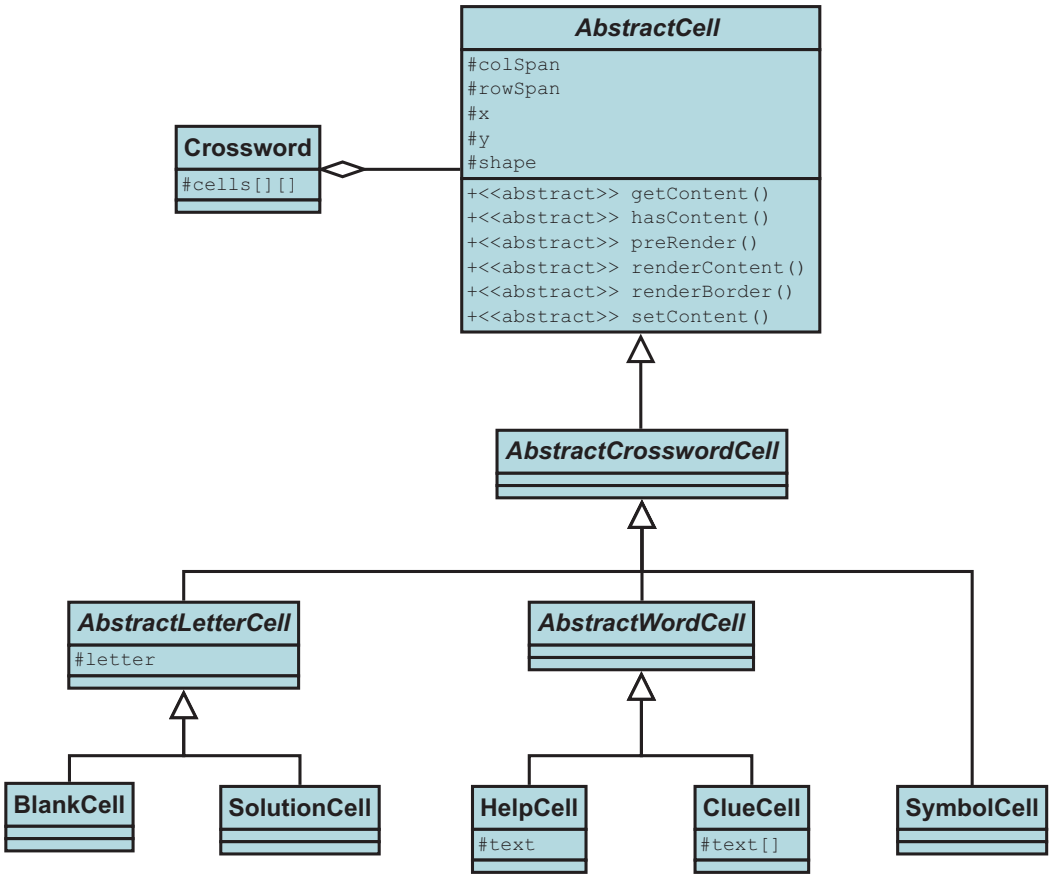
Na obrázku 5.1 vpravo je znázorněn balík `cz.itpro.cwdstudio.ui`. Tento balík obsahuje různé části uživatelského rozhraní programu. V diagramu je zobrazena bázová třída pro zobrazování hádanek, třídu `PuzzleViewer`, která je již vizuální komponentou odvozenou

[1] `www.jdom.org`
[2] DOM - Document Object Model

od třídy `JPanel` ze standardního balíku jazyka Java™ `javax.swing`. Komponenta třídy `PuzzleViewer` slouží k zobrazení hádanky a přizpůsobení její velikosti dostupné kreslicí ploše. Od této třídy se pro každý typ hádanky vytváří specializace, která ošetřuje uživatelské vstupy, zprostředkovává interakci s uživatelem a do hádanky vykresluje v další vrstvě doplňující informace (u švédských křížovek, např. pozici kurzoru, směr doplňování...). Podobně třída `PuzzleForm` reprezentuje obecné vnitřní okno aplikace s hádankou. Tato třída implementuje základní společné funkce jako je zmenšování a zvětšování hádanky, posunování jejím zobrazením, ukládání a načítání do souboru a na obecné úrovni i strukturu určenou k navracení se v činnosti (tzv. undo a redo funkce).

5.2 Model švédských křížovek

Věnujme se nyní samotné reprezentaci švédských křížovek. Na křížovku můžeme nahlížet jako na dvourozměrné pole buněk. Jednotlivé typy buněk byly popsány v kapitole 2.2. Obrázek 5.2 popisuje hierarchii tříd určených k datové reprezentaci jednotlivých políček.



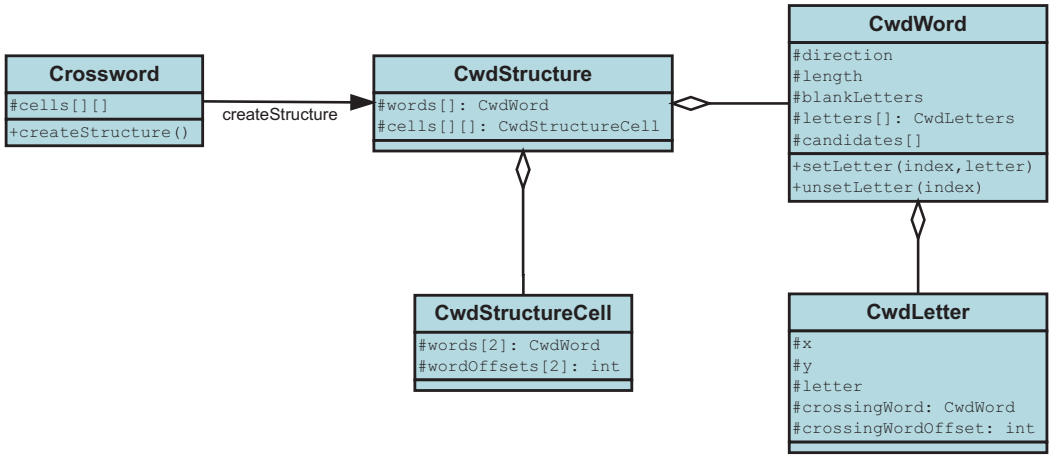
Obrázek 5.2: Model uchovávání a vykreslování políček švédských křížovek

Abstraktní bázová třída `AbstractCell` obsahuje základní společné rysy buněk, tj. pozici buňky, velikost v případě sloučení přes více políček a tvar buňky. Tvar buňky je instancí třídy `RectangularShape` a obsahuje kromě tvaru a velikosti i pozici pro vykreslení dané buňky. To výrazně urychluje vykreslení celé křížovky, protože tyto údaje není třeba při překreslení přepočítávat.

V modelu si můžeme všimnout třídy `AbstractCrosswordCell`, která je však v modelu vložena především pro budoucí použití. Zastřešuje možnost do křížovky přímo vložit nějaká grafická data, např. reklamu. V odevzdané implementaci je však tato třída nepoužita, resp. její implementace je naprosto prázdná.

Vykreslení buněk odvozených od třídy `AbstractLetterCell` nelze příliš urychlit. Na rozdíl od toho buňky odvozené od třídy `AbstractWordCell`, které vykrelují víceřádkový text, je třeba předpočítávat a připravit takovým způsobem, aby výpočtů při vykreslování bylo co nejméně. Informace o řešení tohoto problému naleznete v kapitole 4.3.

V průběhu generování metodou, která byla popsána v kapitole 3.3, však není možné ke křížovce přistupovat jako k dvojrozměnému poli buněk. Práce s tímto pohledem na křížovku by totiž byla až příliš pomalá. Obrázek 5.3 znázorňuje model třídy `CwdStructure`, který slouží k rychlému přístupu k jednotlivým pozicím v křížovce. Operace `createStructure` třídy `Crossword` projde všechna políčka a sestaví na obrázku uvedenou strukturu. Pomocí ní je možné prostřednictvím souřadnice políčka okamžitě získat slova, která se v tomto políčku kříží. Zároveň je ale také možné iterovat přes všechna slova obsažená v křížovce. Volání metody pro zápis písmena ve třídě `CwdWord` způsobí zápis nejen do vybraného slova, ale i na příslušnou pozici do slova křížícího se. Zároveň je v této třídě obsažena hodnota určující aktuální počet volných pozic v daném slově, ale i jeho délku. Všechny hodnoty jsou předpočítané při vytvoření celého modelu a poté průběžně aktualizovány při změnách.

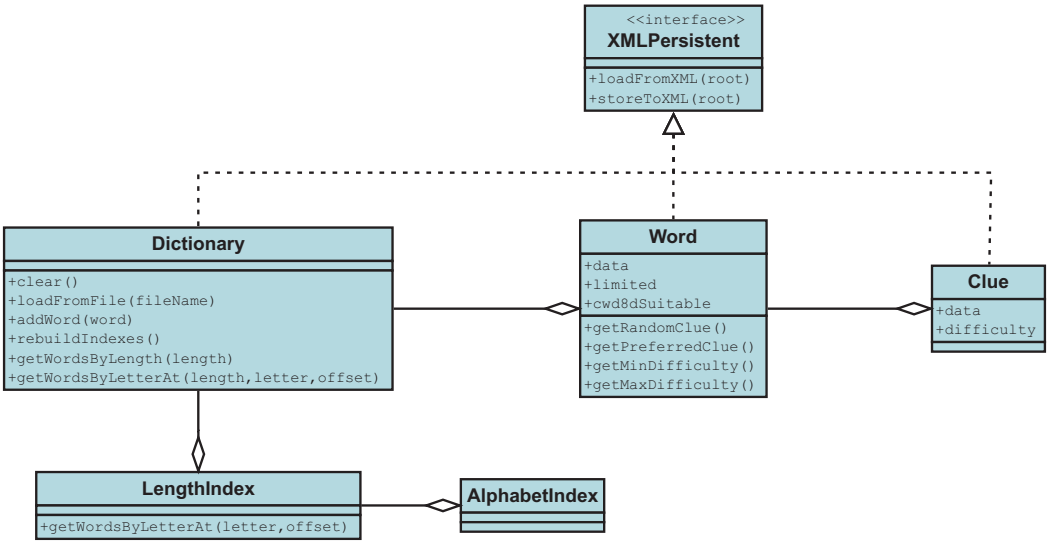


Obrázek 5.3: Vnitřní struktura křížovky určená pro generování

Tato struktura je vytvářena obvykle před spuštěním procesu generování. Vytváří se ale také za účelem získání některých statistických hodnot, jakými může být průměrná délka slova, procentuální délka tajenky apod.

5.3 Struktura slovníku

Pokud chceme ve slovníku používat vyhledávání pomocí indexu tak, jak byl definován na obrázku 3.7, a pokud chceme řešit problémy s obtížností automaticky generovaných křížovek, které byly popsány v kapitole 3.4.1, můžeme použít strukturu slovníku podle obrázku 5.4. Na tomto diagramu je mimo jiné znázorněno rozhraní `XMLPersistent`, které je určeno k zachování perzistence dat načítáním a ukládáním do hodnot do XML dokumentu. Toto rozhraní bylo detailněji popsáno v kapitole 5.1.



Obrázek 5.4: Struktura slovníku - diagram tříd

V implementaci je nutné počítat s tím, že uživatel bude chtít použít nějaký vlastní slovník. V takovém slovníku obvykle nebudou uvedeny koeficienty obtížnosti nebo nelze počítat s vícenásobnými legendovými výrazy pro stejné slovo. Někdy dokonce nejsou k dispozici ani samotné legendové výrazy a slovník obsahuje pouze slova. Navržený datový model a algoritmy musí s těmi situacemi počítat a být schopny například kontrolu obtížnosti křížovky (kapitola 3.4.1) odstavit z činnosti v případě, že slovník nedisponuje potřebnými daty.

Kapitola 6

Možnosti dalšího vývoje

Již od samotného počátku práce na tomto projektu byly brány v potaz možnosti dalšího rozšiřování a vylepšování aplikace. V objektovém modelu aplikace je hojně využíváno abstraktních tříd, které lze použít ke snadnému odvození dalších typů hádanek (jakým jsou například klasické křížovky nebo osmisměrky). Důležitou myšlenkou při návrhu bylo to, aby se pokud možno žádný kód nemusel při rozšiřování aplikace kopírovat na další místa.

Plánovaná rozšíření lze rozdělit do dvou kategorií. Rozšíření týkající se přímo švédských křížovek, která budou implementována přednostně a rozšíření, na které přijde řada teprve po dokončení a vyladění první části. Tato další rozšíření, která budou implementována až v druhé řadě, budou především doplňovat další typy podporovaných hádanek (osmisměrky, klasické křížovky,...). V první řadě se tedy počítá s těmito úpravami:

- *kontrola validnosti návrhu křížovky* - Návrh tvaru křížovky je při automatickém generování tím nejdůležitějším a také nejnáročnějším požadavkem na uživatele. Neopatrným návrhem se může stát, že danou křížovku v lepším případě nelze vygenerovat, v horším se vygeneruje, ale obsahuje chyby, které křížovky obsahovat nesmí. Kontrola tvaru křížovky by tak měla například kontrolovat zda:
  - před každým slovem je uvedeno políčko pro legendový výraz
  - v křížovce nejsou jednopísmenné výrazy
  - křížovka neobsahuje příliš velkou souvislou obdélníkovou nebo čtvercovou oblast, kterou je problematické vygenerovat
  - je celá oblast křížovky souvislá (tzn. neobsahuje dvě nebo více oddělených ploch)

  Políčka či pozice, ve kterých by byla jedna z výše uvedených podmínek porušena, by měla být graficky zvýrazněna tak, aby byl uživatel upozorněn na vzniklý problém.

- *kontrola přetečení obsahu v textových polích* - Při automatickém doplňování legend se leckdy stane, že zadaný text se do políčka s legendou nevejde. Aby uživatel nemusel všechna políčka ručně kontrolovat, je vhodné, aby program automaticky graficky zvýraznil všechna políčka, která se dožadují uživatelova zásahu.

- *zabezpečení aplikace a slovníku proti neoprávněnému šíření* - Program CwdStudio bude komerčně šířen za licenční poplatek. Proto je třeba program zabezpečit tak, aby nebylo zcela triviální šířit jeho kopie. Dále je třeba vytvořit zkušební verzi, která bude funkční pouze po stanovenou časovou lhůtu a umožní uživateli vyzkoušet všechny vlastnosti produktu. Slovník je třeba udržovat v šifrované podobě, aby nedošlo k jeho

zneužití např. pro konkurenční produkty. Zabezpeční musí být takové, aby legitimního uživatele neobtěžovalo a zároveň neoprávněného uživatele odradilo od získání nebo alespoň znepříjemnilo cestu k získání nelegální kopie.

- *pokročilá editace legendy* - Uživateli by měly být k dispozici nástroje, které umožní hromadnou manipulaci s legendami, například převod všech písmen v legendových výrazech na malá. Dále bude vhodné vytvořit formulář pro pokročilé editování obsahu, ve kterém půjde text kopírovat, vpisovat na pozici kurzoru a podobně. V tomto formuláři by také měly být vidět legendy navržené na základě načtených informací ze slovníku.

- *pokročilá editace políček pomůcky* - Program by měl být automaticky schopen navrhnout uživateli slova, která jsou vhodná pro vložení do pomůcky na základě jejich obtížnosti uvedené ve slovníku. Políčko pomůcky navíc může být sloučeno přes více buněk. Tato funkce již je podporována, není však propojena s uživatelským rozhraním.

- *podpora obrázků pro symbolová políčka* - Do políček, která jsou v křížovce nevyužita se mnohdy nedává pouze nějaký dekorační symbol nebo nápis, ale v praxi je tento prostor využíván pro zobrazení loga časopisu, tiskoviny nebo produktu, který je prostřednictvím křížovky propagován. V nastavení vzhledu křížovky by tedy mělo být možné zvolit obrázek ve vektorovém nebo rastrovém formátu a jeho relativní velikost k políčku.

- *export do negrafických formátů* - Křížovky jsou natolik populární záležitostí, že se často objevují i na webových stránkách. Podstatnou funkcí je tedy i export do jiných formátů než jsou klasické rastrové a vektorové obrázky. Požadovaným exportním formátem je export na web. V předchozí verzi systému byl export realizován pomocí Java^{TM} appletu, který křížovku zobrazoval a umožňoval její luštění. Problémem však v tehdejší době byla nepříliš rozšířená podpora spouštění Java^{TM} appletů. V tomto produktu bude zobrazování a luštění křížovky realizováno přímo pomocí HTML, CSS a Javascriptu. To by mělo rozšířit kompatibilitu na maximum. Podporované prohlížeče budou Microsoft Internet Explorer 6, 7, 8 (beta), Mozilla Firefox 2, 3, Opera 9 a Apple Safari.

  Pro tvůrce křížovek je občas také důležité mít možnost si křížovku zpracovat zcela vlastním způsobem. Za tímto účelem se jako ideální jeví export do formátu CSV (Comma separated values), který lze načíst jako tabulku téměř do libovolného programu, který tabulky podporuje.

- *nástroje pro manuální tvorbu křížovek* - Jak již bylo uvedeno v úvodních kapitolách, křížovka vygenerovaná počítačem nedosahuje takové kvality, jako křížovka vytvořená přemýšlejícím člověkem. Již předchozí produkt byl používán řadou uživatelů pouze jako nástroj pro sazbu a asistenci při ruční tvorbě křížovek. Pro tvůrce, kteří chtějí křížovky vytvářet ručně, bude v programu CwdStudio dostupný vlastní nástrojový panel tak, aby měl tvůrce všechny potřebné nástroje vždy po ruce.

- *propojení s online nástrojem pro sdílení slovníku DictShare* - Základním slovníkem, který bude s aplikací dodávan je komunitní slovník. Je tedy třeba, aby úpravy ve slovníku provedené jedním uživatelem byly sdíleny i s ostatními uživateli. K tomu slouží služba pro sdílení křížovkářského slovníku DicShare, která je veřejně dostupná na adrese: `http://dictshare.itpro.cz`. V budoucím vývoji produktu se počítá s přímým

napojením programu CwdStudio na tuto službu, aby bylo možné stahovat aktuální verzi komunitního slovníku, ale i sdílet provedené změny v lokálním slovníku s ostatními uživateli.

Poté, co budou dokončeny výše uvedené úpravy, bude CwdStudio dále rozšířeno o podporu:

- *klasických křížovek* - Klasické křížovky jsou založeny na velmi podobném principu jako křížovky švédské. V podstatě se jedná pouze o jinou formu zobrazení křížovky. Generátor tohoto typu křížovek bude tedy sdílený s křížovkami švédskými. Interní datová reprezentace generátoru totiž není přímo napojena na konkrétní typ hádanky.
- *osmisměrek* - Osmisměrky jsou u nás zřejmě druhým nejpopulárnějším typem hádanky. Problematika generování osmisměrek je od generování křížovek značně odlišná. V tomto ohledu se tedy bude jednat o velmi rozsáhlé rozšíření zahrnující nejen vlastní třídu pro zobrazování, ale i generátor pracující na zcela jiném principu.

Termín dokončení celého projektu je stanoven na konec roku 2009, a protože je již od počátku zamýšlen jako komerční a počítá se s jeho praktickým použitím, očekává se ještě řada dalších námětů na vylepšení, které se objeví v průběhu vývoje na základě zkušeností uživatelů se zkušební verzí.

Kapitola 7

Závěr

V první části práce jsem čtenáře tohoto dokumentu seznámil s historií křížovek a termíny používanými v oblasti problematiky tvorby a luštění křížovek. Čtenář také mohl získat přehled o rozsáhlosti stavového prostoru a problémech, které při strojovém generování křížovek vznikají.

V dalších částech jsem zdokumentoval použitý princip generování křížovek, ale i zajímavé problémy, které bylo třeba v průběhu vývoje vyřešit, především podporu CMYKových barev v různých grafických formátech a v programech v Javě.

Povedlo se vytvořit generátor křížovek, který na základě metrik rychlosti generování i kvality generovaných křížovek podle pravidel SČHAK výrazně předčí svého předchůdce Crosswords - ITPro CZ, ale i jiné konkurenční produkty, které umožňují automatické generování křížovek.

Výsledky práce vytvořeného generátoru si lze vyzkošet v příloze D. Zde je uvedeno několik křížovek, které byly v programu CwdStudio vytvořeny. Na těchto ukázkových křížovkách nebyly provedeny žádné další grafické úpravy a jsou uvedeny přímo ve tvaru, v jakém byly vyexportovány z programu. Uvedené časy ukazují také dobu návrhu křížovky uživatelem a samotnou dobu vygenerování obsahu.

Pro podporu snadného ovládání generátoru jsem navrhl uživatelské prostředí, které je připraveno pro další rozšiřování a maximálně využívá výhod dědičnosti, rozhraní a potažmo celého objektově orientovaného návrhu aplikace v Javě.

V rámci diplomového projektu bylo kromě samotného generátoru křížovek vytvořeno i kvalitní jádro pro studio na tvorbu křížovek. Na tomto jádře se bude dále stavět a vývoj tohoto projektu bude pokračovat i mimo rámec diplomového projektu. Jeho dokončení je naplánováno na konec roku 2009.

Vlastním přínosem práce je nejen vytvoření programu, který svými funkcemi výrazně předčí podobný dostupný software a některé jeho funkce jsou dokonce zcela unikátní. Přínosem je také rozšíření knihovny FreeHEP VectorGraphics o podporu barevného modelu CMYK a praktické vyzkoušení možností formátu SVG z hlediska implementace v různých programech.

Seznam obrázků

Seznam tabulek

Literatura

[1] Adobe Systems Incorporated: *PostScript® Language Reference - third edition*. Addison-Wesley Publishing Company, 1999, ISBN 0-201-37922-8.

[2] Adobe Systems Incorporated: PDF Reference - sixth edition [online]. `http://www.adobe.com/devnet/pdf/pdf_reference.html`, 2006-11.

[3] CoJeCo®: křížovka, druh hádanky - CoJeCo - Vaše encyklopedie [online]. `http://www.cojeco.cz/index.php?detail=1&id_desc=49894`, 2000-08-15 [cit. 2008-12-16].

[4] Dietrich, H.: XCmyk - CMYK to RGB Calculator. `http://www.codeproject.com/KB/applications/xcmyk.aspx`, 2003-07-05 [cit. 2008-05-06].

[5] Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[6] Josefsson, S.: RFC 4648: The Base16, Base32, and Base64 Data Encodings [online]. `http://tools.ietf.org/html/rfc4648`, 2006-10-01 [cit. 2009-05-05].

[7] Mařík, V., Štěpánková, O., Lažanský, J.: *Umělá inteligence*. Academia, 1993, ISBN 80-200-0496-3.

[8] Šoba, P.: Historie křížovek [online]. `http://petrsoba.webnode.cz/news/historie-krizovek/`, 2008-04-20 [cit. 2008-12-17].

[9] Svaz českých hádankářů a křížovkářů: Směrnice pro tvorbu křížovek [online]. `http://www.e-rebus.cz/Krizovky/SmerK.htm`, 2007-01-13 [cit. 2008-12-14].

[10] Svaz českých hádankářů a křížovkářů: Kritéria kvality křížovek [online]. `http://www.e-rebus.cz/Krizovky/Kvalita.htm`, 2008-11-12 [cit. 2008-12-14].

[11] Trnková, J.: Naučte se luštit [online]. `http://www.lidovky.cz/naucte-se-lustit-0kz-/ln_noviny.asp?c=A080806_000133_ln_noviny_sko`, 2008-08-06 [cit. 2008-12-17].

[12] Wikipedia: Crossword — Wikipedia, The Free Encyclopedia [online]. `http://en.wikipedia.org/w/index.php?title=Crossword&oldid=257458372`, 2008-12-12 [cit. 2008-12-17].

[13] Wikipedie: CMYK — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/w/index.php?title=CMYK&oldid=3293019, 2008-11-18 [cit. 2008-05-05].

[14] Wikipedie: Abeceda — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/w/index.php?title=Abeceda&oldid=3368837, 2008-12-09 [cit. 2008-12-14].

[15] Wikipedie: Vesmír — Wikipedie: Otevřená encyklopedie [online]. http://cs.wikipedia.org/w/index.php?title=Vesm%C3%ADr&oldid=3382546, 2008-12-12 [cit. 2008-12-14].

Příloha A

Tabulka CMYK barev pro SVG

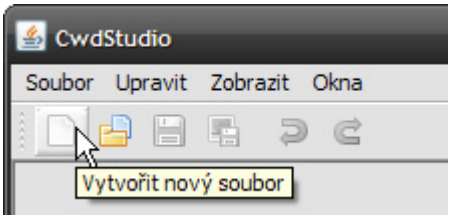
Převzato z `http://www.december.com/html/spec/colorsvgcmyk.html`

white cmyk(0,0,0,0)		**whitesmoke** cmyk(0,0,0,4)		**gainsboro** cmyk(0,0,0,14)	
lightgray cmyk(0,0,0,17)		**lightgrey** cmyk(0,0,0,17)		**silver** cmyk(0,0,0,25)	
darkgray cmyk(0,0,0,34)		**darkgrey** cmyk(0,0,0,34)		**grey** cmyk(0,0,0,50)	
gray cmyk(0,0,0,50)		**dimgray** cmyk(0,0,0,59)		**dimgrey** cmyk(0,0,0,59)	
black cmyk(0,0,0,100)		**ivory** cmyk(0,0,6,0)		**beige** cmyk(0,0,10,4)	
lightyellow cmyk(0,0,12,0)		**lightgoldenrodyellow** cmyk(0,0,16,2)		**yellow** cmyk(0,0,100,0)	
olive cmyk(0,0,100,50)		**snow** cmyk(0,2,2,0)		**floralwhite** cmyk(0,2,6,0)	
lemonchiffon cmyk(0,2,20,0)		**oldlace** cmyk(0,3,9,1)		**cornsilk** cmyk(0,3,14,0)	
palegoldenrod cmyk(0,3,29,7)		**darkkhaki** cmyk(0,3,43,26)		**seashell** cmyk(0,4,7,0)	
linen cmyk(0,4,8,2)		**khaki** cmyk(0,4,42,6)		**lavenderblush** cmyk(0,6,4,0)	
antiquewhite cmyk(0,6,14,2)		**papayawhip** cmyk(0,6,16,0)		**blanchedalmond** cmyk(0,8,20,0)	
wheat cmyk(0,9,27,4)		**mistyrose** cmyk(0,11,12,0)		**bisque** cmyk(0,11,23,0)	
moccasin cmyk(0,11,29,0)		**thistle** cmyk(0,12,0,15)		**navajowhite** cmyk(0,13,32,0)	
tan cmyk(0,14,33,18)		**peachpuff** cmyk(0,15,27,0)		**gold** cmyk(0,16,100,0)	
burlywood cmyk(0,17,39,13)		**rosybrown** cmyk(0,24,24,26)		**goldenrod** cmyk(0,24,85,15)	
pink cmyk(0,25,20,0)		**darkgoldenrod** cmyk(0,27,94,28)		**plum** cmyk(0,28,0,13)	
lightpink cmyk(0,29,24,0)		**sandybrown** cmyk(0,33,61,4)		**peru** cmyk(0,35,69,20)	
orange cmyk(0,35,100,0)		**darksalmon** cmyk(0,36,48,9)		**lightsalmon** cmyk(0,37,52,0)	
violet cmyk(0,45,0,7)		**darkorange** cmyk(0,45,100,0)		**lightcoral** cmyk(0,47,47,6)	
orchid cmyk(0,49,2,15)		**palevioletred** cmyk(0,49,33,14)		**salmon** cmyk(0,49,54,2)	
sienna cmyk(0,49,72,37)		**coral** cmyk(0,50,69,0)		**chocolate** cmyk(0,50,86,18)	
saddlebrown cmyk(0,50,86,45)		**indianred** cmyk(0,55,55,20)		**hotpink** cmyk(0,59,29,0)	
tomato cmyk(0,61,72,0)		**orangered** cmyk(0,73,100,0)		**brown** cmyk(0,75,75,35)	
firebrick cmyk(0,81,81,30)		**mediumvioletred** cmyk(0,89,33,22)		**crimson** cmyk(0,91,73,14)	
deeppink cmyk(0,92,42,0)		**fuchsia** cmyk(0,100,0,0)		**magenta** cmyk(0,100,0,0)	
darkmagenta cmyk(0,100,0,45)		**purple** cmyk(0,100,0,50)		**red** cmyk(0,100,100,0)	
darkred cmyk(0,100,100,45)		**maroon** cmyk(0,100,100,50)		**ghostwhite** cmyk(3,3,0,0)	
mintcream cmyk(4,0,2,0)		**azure** cmyk(6,0,0,0)		**honeydew** cmyk(6,0,6,0)	
aliceblue cmyk(6,3,0,0)		**lavender** cmyk(8,8,0,2)		**lightcyan** cmyk(12,0,0,0)	
mediumorchid cmyk(12,60,0,17)		**darkolivegreen** cmyk(21,0,56,58)		**lightsteelblue** cmyk(21,12,0,13)	
lightslategrey cmyk(22,11,0,40)		**lightslategray** cmyk(22,11,0,40)		**slategrey** cmyk(22,11,0,44)	
slategray cmyk(22,11,0,44)		**powderblue** cmyk(23,3,0,10)		**darkseagreen** cmyk(24,0,24,26)	
olivedrab cmyk(25,0,75,44)		**yellowgreen** cmyk(25,0,76,20)		**lightblue** cmyk(25,6,0,10)	
darkorchid cmyk(25,75,0,20)		**paleturquoise** cmyk(26,0,0,7)		**darkviolet** cmyk(30,100,0,17)	
greenyellow cmyk(32,0,82,0)		**mediumpurple** cmyk(33,49,0,14)		**palegreen** cmyk(39,0,39,2)	
lightgreen cmyk(39,0,39,7)		**blueviolet** cmyk(39,81,0,11)		**darkslategray** cmyk(41,0,0,69)	
darkslategrey cmyk(41,0,0,69)		**cadetblue** cmyk(41,1,0,37)		**indigo** cmyk(42,100,0,49)	
skyblue cmyk(43,12,0,8)		**lightskyblue** cmyk(46,18,0,2)		**mediumslateblue** cmyk(48,56,0,7)	
slateblue cmyk(48,56,0,20)		**darkslateblue** cmyk(48,56,0,45)		**aquamarine** cmyk(50,0,17,0)	
mediumaquamarine cmyk(50,0,17,20)		**chartreuse** cmyk(50,0,100,0)		**lawngreen** cmyk(51,0,100,1)	
cornflowerblue cmyk(58,37,0,7)		**steelblue** cmyk(61,28,0,29)		**mediumturquoise** cmyk(66,0,2,18)	
mediumseagreen cmyk(66,0,37,30)		**seagreen** cmyk(67,0,37,45)		**turquoise** cmyk(71,0,7,12)	
royalblue cmyk(71,53,0,12)		**limegreen** cmyk(76,0,76,20)		**forestgreen** cmyk(76,0,76,45)	
midnightblue cmyk(78,78,0,56)		**lightseagreen** cmyk(82,0,4,30)		**dodgerblue** cmyk(88,44,0,0)	
aqua cmyk(100,0,0,0)		**cyan** cmyk(100,0,0,0)		**darkcyan** cmyk(100,0,0,45)	
teal cmyk(100,0,0,50)		**mediumspringgreen** cmyk(100,0,38,2)		**springgreen** cmyk(100,0,50,0)	
lime cmyk(100,0,100,0)		**green** cmyk(100,0,100,50)		**darkgreen** cmyk(100,0,100,61)	

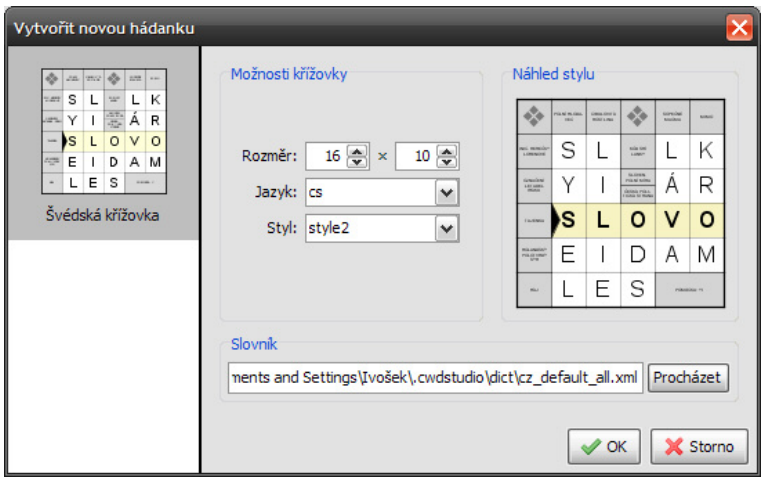
Příloha B

Vytvoření křížovky krok za krokem

1. Spusťte program CwdStudio a klikněte na ikonku „Vytvořit nový soubor“.

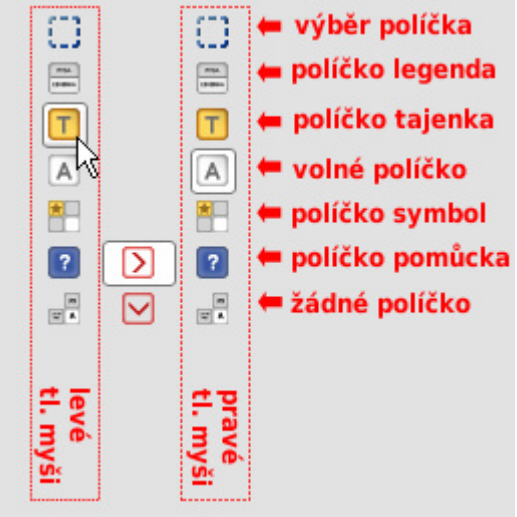


2. Na zobrazeném formuláři zvolte typ křížovky „Švédská křížovka“, zadejte požadovaný rozměr a styl křížovky. Zvolte také slovník, který budete pro generování a práci s touto křížovkou chtít používat. Slovník můžete případně zvolit i později.

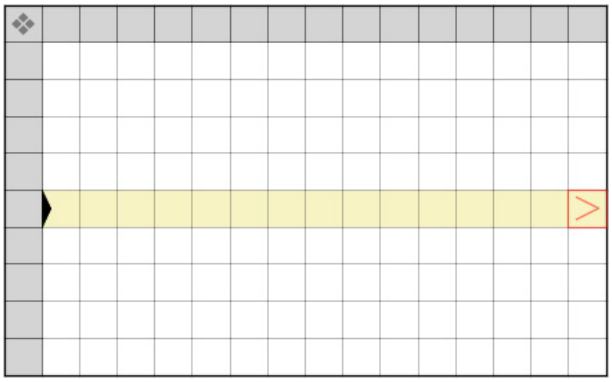


3. Po vytvoření křížovky vám bude automaticky zobrazen nástrojový panel pro návrh křížovky. Nástrojový panel se skládá ze dvou sloupců. Sloupec vlevo určuje nástroj vybraný pro levé tlačítko myši. Pravý sloupec zase určuje nástroj pro pravé tlačítko myši. Která ikona zastupuje který nástroj je popsáno na obrázku. Nástroj výběr políčka při kliknutí pouze nastavuje pozici kurzoru na políčko, na které je kliknuto. Nástroj výběr políčka je napevno nastaven na prostřední tlačítko myši bez ohledu na to, jaký nástroj je zvolen na ostatních tlačítkách myši. Ostatní nástroje po kliknutí změní typ políčka na zvolený typ.

Zvolíme tedy nástroj „políčko tajenka“.

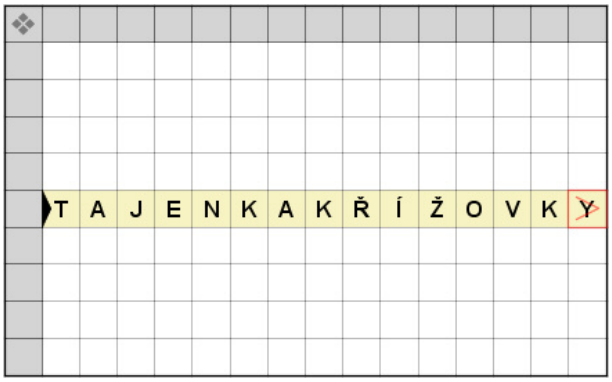


4. Nyní v křížovce označíme políčka, na kterých se bude nacházet tajenka.

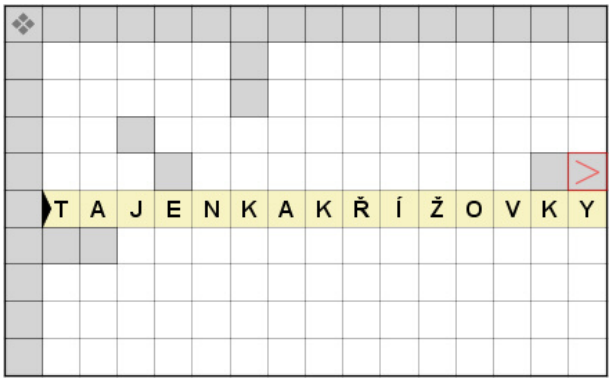


Na panelu nástrojů nalezneme i přepínač směru doplňování. Zvolte takový směr doplňování, v jakém směru jste označili pozici pro tajenku. V tomto případě vodorovný směr.

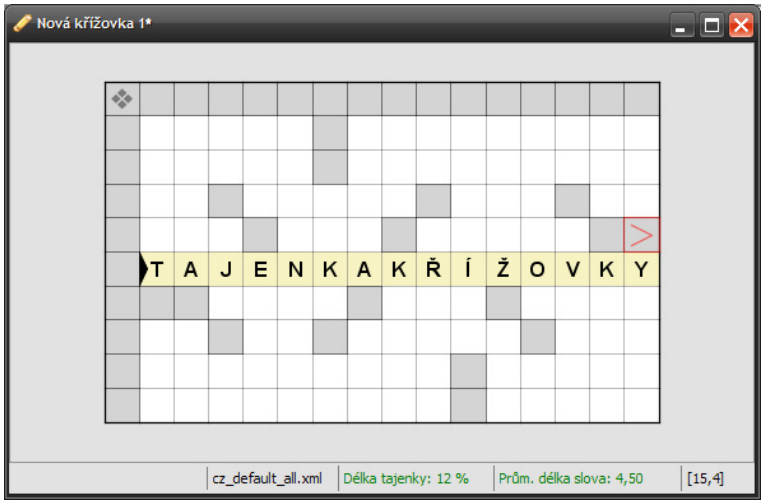
Přemístíme kurzor na první políčko tajenky (např. kliknutím prostředním tlačítkem myši) a zadáme text tajenky.



Nyní na panelu nástrojů zvolíme nástroj „Políčko legenda“ a označíme políčka, která mají být legendová. Při návrhu tvaru dbáme na to, aby nevznikala jednopísmenná slova a zároveň se snažíme rozdělit příliš velké souvislé plochy volných políček.

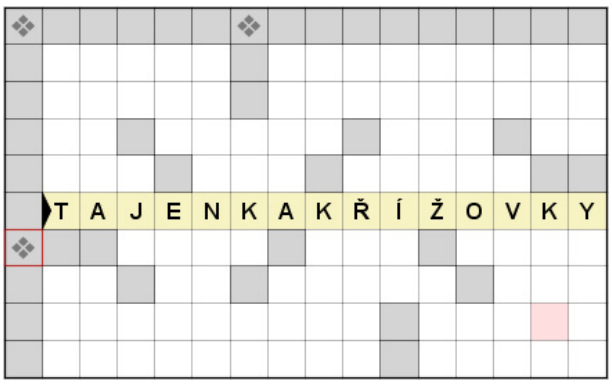


Rozmístění políček pro legendu by mělo být alespoň částečně symetrické. Neměla by se vyskytovat samostatná políčka. Ve stavovém pruhu okna můžeme sledovat průměrnou délku slova. Popisek by ideálně měl svítit zeleně. Optimální průměrná délka slova by se měla pohybovat okolo hodnoty 4,5.

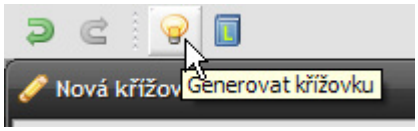


Na panelu nástrojů zvolte nástroj „Políčko symbol“ a toto políčko nastavte na místa,

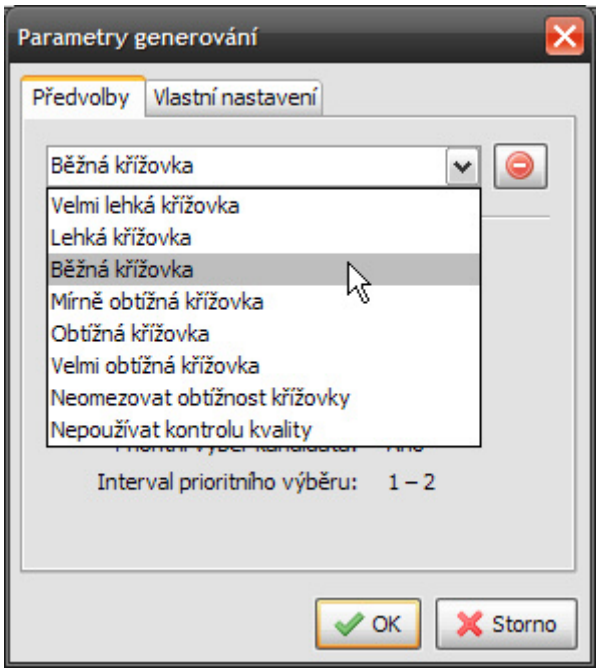
kde nebude umístěna žádná legenda. Tímto typem políčka doladíte symetričnost obrazce.



Po dokončení návrhu tvaru křížovky klikněte na ikonu „Generovat křížovku“.

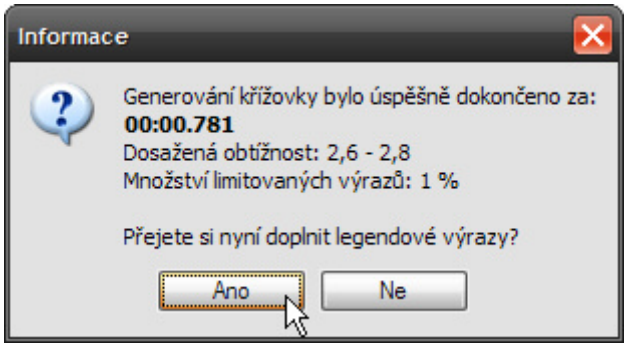


Na zobrazeném dialogu zvolte požadované parametry generování. Parametry lze vybrat z přednastavených profilů.



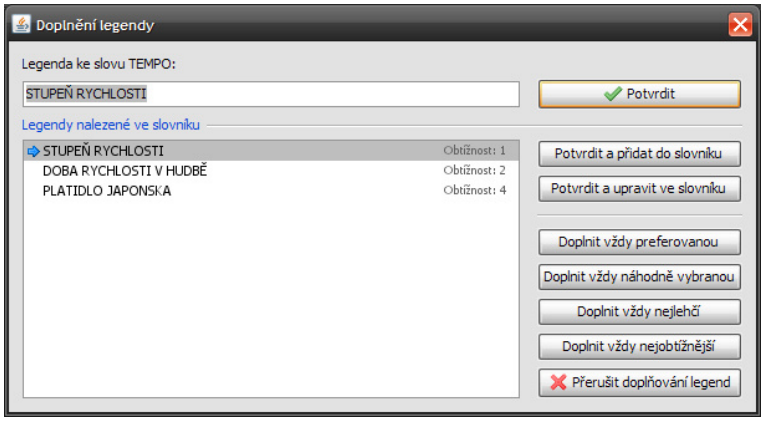
Poté vyčkejte na vygenerování křížovky. Průběh je vizualizován. Čím červenější barva, tím vícekrát generátor přes toto políčko prošel a pokoušel se ho doplnit. Pokud se generátoru křížovku nedaří dokončit do několika minut, generování přerušte a upravte ty pozice, přes které provedl generátor nejvíce průchodů, a spusťte generování znovu.

O dokončení generování budete informováni vyskakovacím oknem. Rovnou vám bude nabídnuta možnost automatického doplnění legendových výrazů. Na zobrazeném dialogu klikněte na tlačítko „Ano“.



Na formuláři pro automatické doplnění legedových výrazů zvolte vždy nejvhodnější legendu k zadanému výrazu. Modrou šipkou je vždy označen výraz, který je ve slovníku označen jako nejvhodnější (pokud slovník těmito informacemi disponuje).

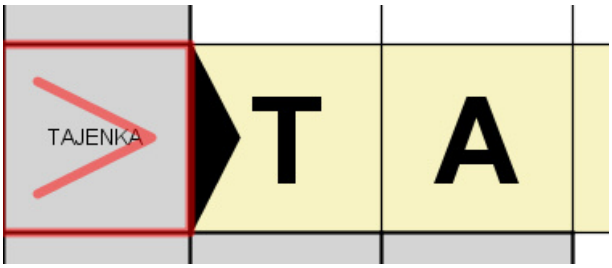
Zvolit můžete také doplnění všech legendových výrazů náhodně nebo podle kritéria (obtížnost, preferované…).



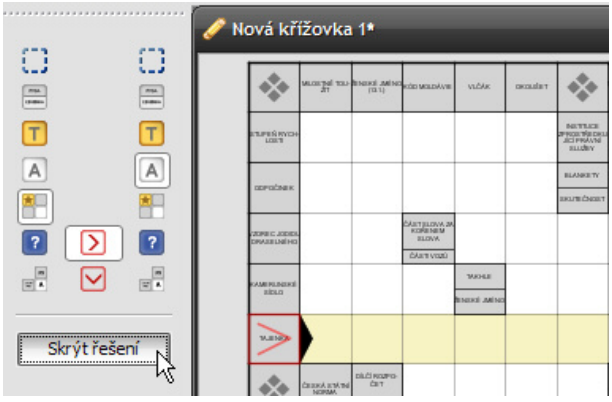
V křížovce máte poté doplněné všechny legendy. Je vhodné legendy zkontrolovat, zda

se všechny vejdou do zadaných políček a zda není třeba některé legendové výrazy upravit.

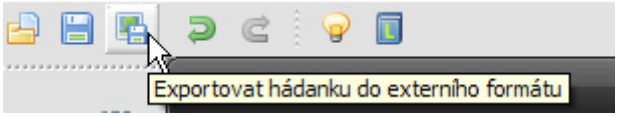
Pomocí nástroje „Výběr políčka“ zvolte legendové políčko, které náleží k tajence a do něj vepište např. slovo „TAJENKA“.



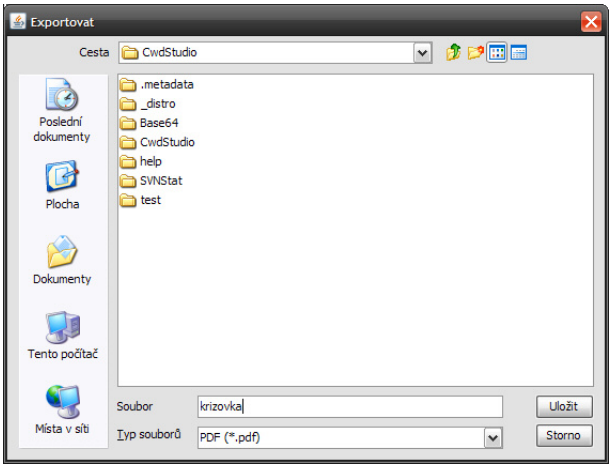
Kliknutím na tlačítko „Skrýt řešení“ připravíte křížovku pro výstup, v jakém uvidí křížovku luštitelé.



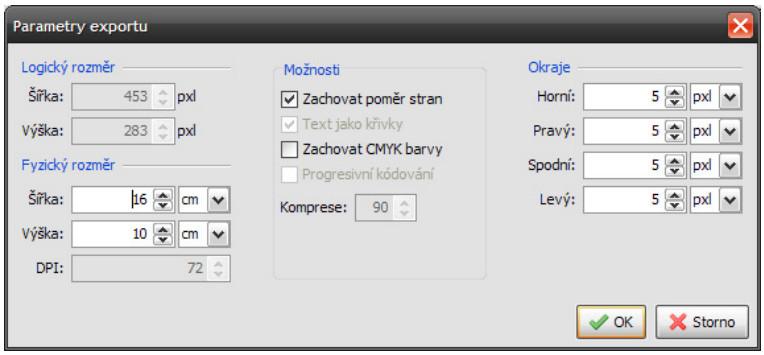
Klikněte na ikonu „Exportovat hádanku do externího formátu“.



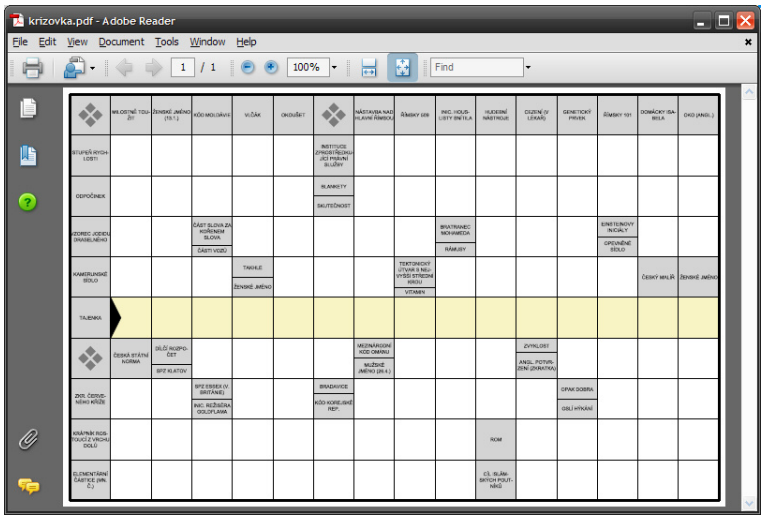
Zvolte formát, cestu a jméno souboru, do kterého chcete provést export.



Nastavte parametry exportu. Důležité je nastavení exportní velikosti. Další parametry jsou volitelné. Tyto parametry lze nastavovat v závislosti na vybraném exportním formátu, podle možností, které tento formát poskytuje. Nastavit lze například převod textu na křivky nebo zachování barev v barevném modelu CMYK.



Použijte externí program pro další manipulaci s vytvořeným exportem.



Příloha C

Obsah CD

`bin`	Spustitelné binární soubory distribuce
`doc`	Dokumentace
`doc/projekt.pdf`	Technická zpráva práce
`doc/javadoc`	Úplná programová dokumentace
`src`	Zdrojové kódy
`src/cz/itpro`	Zdrojové kódy diplomové práce
`src/org/freehep`	Upravené zdrojové kódy FreeHEP VectorGraphics
`src/LaTeX`	Zdrojový kód technické zprávy

Spuštění programu

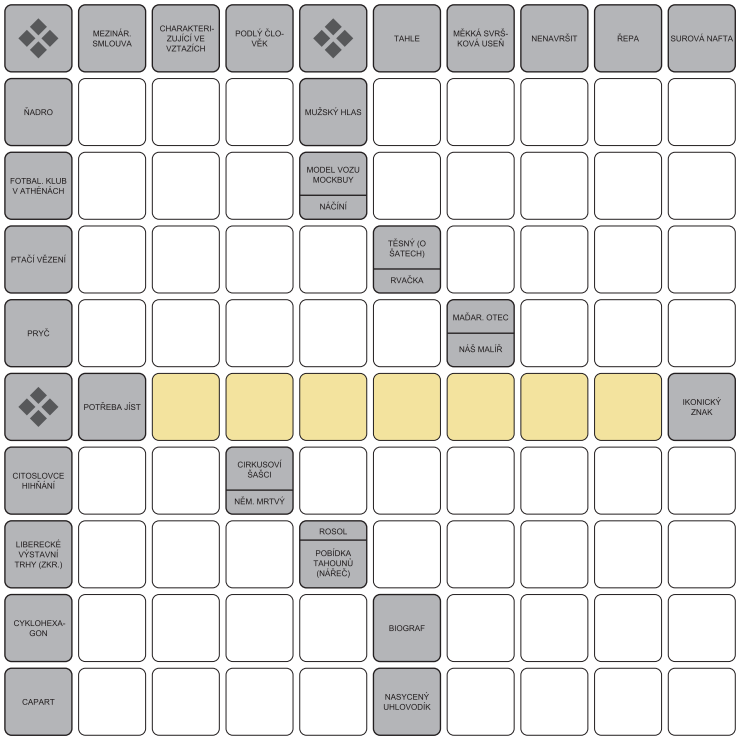
Vyžaduje nainstalovaný Java™ Runtime Enviroment 1.6.0 nebo vyšší. Program se spouští v adresáři `bin` zadáním příkazu:

win32	`CwdStudio.exe`
ostatní	`java -jar CwdStudio.jar`

Příloha D

Ukázkové výstupy

Tato příloha zobrazuje ukázky výstupních souborů přesně ve tvaru, v jakém byly vyexportovány z programu. Na tyto soubory nebyly aplikovány žádné dodatečné úpravy v grafickém editoru.



Obrázek D.1: Křížovka 10×10 - průměrná délka slova: 4,44, doba generování: 0,1 s

Obrázek D.2: Křížovka 20×14 - průměrná délka slova: 4,45, doba generování: 4,5 s

Obrázek D.3: Křížovka 40×28 - průměrná délka slova: 4,37, doba generování: 6,9 s