VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

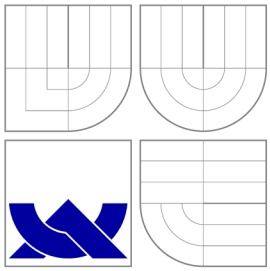
PROGRAM PRO NÁCVIK HMATOVÉ METODY PSANÍ NA KLÁVESNICI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

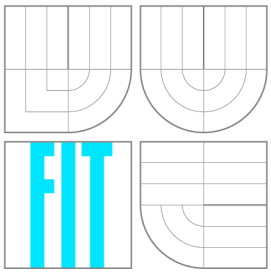
AUTOR PRÁCE
AUTHOR

JAROSLAV ADAMEC

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PROGRAM PRO NÁCVIK HMATOVÉ METODY PSANÍ NA KLÁVESNICI

APPLICATION FOR PRACTICING RAPID TYPING TECHNIQUES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROSLAV ADAMEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Tato práce se zabývá technikou nácviku hmatové metody psaní na klávesnici. Na základě této metody je vytvořen výsledný program, jehož snahou je učit uživatele hmatové metodě. Program používá techniku opisu textu bez kurzoru, který by označoval pozici uživatele v textu, a tím se odlišuje od jiných zkoumaných programů. Pro vytvoření reálnější situace opisu se na monitoru zobrazuje fyzický rozměr formátu A4 pokud je to možné. Do programu byl implementován Levenshteinův algoritmus pro porovnávání opisovaných textů. Výsledný program byl rozšířen mezi uživatele a jejich reakce byly zaznamenány a vyhodnoceny.

Abstract

This thesis considers the technique of practicing touch typing method on keyboard. Final program is based on this method and is trying to teach users the touch typing method. Program is using technique of text rewriting without the cursor, which would mark user's position in text. Another programs tested in this thesis are using cursor. To reach more real situation program shows the real physical size of A4 paper format. Program contains Levenshtein distance algorithm to compare source and target string. Final program was distributed to users and their reaction were written down and analysed.

Klíčová slova

Hmatová metoda, psaní na klávesnici, GUI, grafické uživatelské rozhraní, Levenshteinova vzdálenost, A4 formát, Qt

Keywords

Touch typing, typing on keyboard, GUI, graphical user interface, Levenshtein distance, A4 format, Qt

Citace

Jaroslav Adamec: Program pro nácvik hmatové metody psaní na klávesnici, bakalářská práce, Brno, FIT VUT v Brně, 2010

Program pro nácvik hmatové metody psaní na klávesnici

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Adama Herouta, Ph.D.

.....
Jaroslav Adamec
11. května 2010

Poděkování

Chtěl bych poděkovat panu Ing. Adamu Heroutovi, Ph.D. za cenné rady a připomínky při tvorbě práce.

Obsah

Kapitola 1

Úvod

Počítačová klávesnice je v dnešní době základním prostředkem pro komunikaci s počítačem. Klávesnice umožňuje vkládání znaků do počítače nebo jeho ovládání pomocí různých kláves a klávesových zkratek. Schopnost ovládat klávesnici všemi deseti prsty co nejvyšší rychlostí a s minimálním počtem chyb značně zvýší efektivitu práce s počítačem. Tuto schopnost však většina lidí nemá a neobratným ovládáním klávesnice zbytečně ztrácejí velké možství času. Čím delší text tito lidé do počítače zadávají, tím více klesá jejich pracovní výkonnost.

Tato práce se zabývá tvorbou programu, který by uživateli umožnil zlepšit schopnost ovládání klávesnice. Program by měl učit uživatele správným zásadám hmatové metody psaní na klávesnici, která umožní značně zrychlit a zefektivnit práci s počítačem. Práce se také zabývá průzkumem již existujících programů pro nácvik hmatové techniky či učení psaní všemi deseti prsty a snaží se zmapovat dobré a špatné vlastnosti těchto programů a z nich vycházet. Práce se při návrhu programu snaží nalézt inovativní řešení, která se nevyskytují v ostatních programech a která by mohla zlepšit způsob výuky hmatové metody psaní na klávesnici.

Kapitola 2 se zabývá zásadami hmatové metody a shrnuje její výhody a nevýhody. Kapitola 3 popisuje existující programy pro nácvik hmatové metody a shrnuje jejich vlastnosti. Návrhem a implementací se zabývá kapitola 4. V kapitole 5 se nachází vyhodnocení výsledného programu.

Kapitola 2

Nácvik hmatové metody

Hmatová metoda je velmi efektivní metodou pro psaní znaků na klávesnici. Člověk znalý psaní hmatovou metodou nejenže ovládá klávesnici pomocí všech deseti prstů a ovládá správný prstoklad, ale má i vytvořeny podmíněné reflexy a automatické vazby mezi jednotlivými znaky na klávesnici. Ovládání klávesnice se mění z vědomého ovládání na podvědomé. Znaky klávesnice jsou zafixované v podvědomí včetně směru a vzdálenosti pohybu příslušných prstů. Pisatel je schopen plně se soustředit na prováděnou činnost a nerozptylovat se způsobem ovládání klávesnice, protože je vytvořen dynamický stereotyp psaní. Schopnost ovládání klávesnice hmatovou metodou může být velmi výhodná ať už při programování či pouhém opisu dlouhého textu. Při psaní hmatovou metodou nedochází k přetržení tvůrčí myšlenky, která se může rozvíjet obdobně, jako při diktování, protože nemusíme přemýšlet, jak zadat myšlenku do počítače. Psaní hmatovou metodou také přispívá k ochraně zdraví. Námaha při psaní textu je rozložena na všech deset prstů. Při psaní hmatovou metodou jsou méně namáhány oči, protože odpadá potřeba neustále sledovat zapsaný text na obrazovce. Poznatky uvedené v tomto odstavci jsou převzaty z [7].

Lépe se bude učit hmatová metoda lidem, kteří se někdy učili hrát na klávesový hudební nástroj. Hůře na tom budou lidé, kteří již klávesnici ovládají pouze několika prsty, protože pro ně bude těžké zapojit zbývající prsty, které se doposud na zadávání znaků nepodílely. Lidé, kteří klávesnici ovládají pouze všemi deseti prsty bez použití hmatové metody, mohou mít při nácviku velké problémy, protože již mají vytvořené špatné návyky (*např.* sledování klávesnice), které se budou těžko odnaučovat. Informace v odstavci jsou převzaty z [6].

Na počátku výuky je lepší psát pomaleji a dodržovat správný prstoklad všech prstů, než se snažit psát pouze několika prsty co nejrychleji. Důležité je také dodržovat pravidelné tempo opisu. Zpočátku nám může psaní hmatovou metodou připadat pomalejší, ale v pozdějších fázích nácviku dojdeme k okamžiku, kdy již na písmena nebudeme muset myslet a začneme je psát automaticky velmi rychle. Ovládání klávesnice hmatovou metodou je vhodné pro praváky i leváky. Převzato z [9].

Člověk píšící hmatovou metodou při napsání chyby téměř vždy zjistí, že udělal chybu a tuto chybu okamžitě opraví, aniž by musel sledovat obrazovku či klávesnici. Naproti tomu člověk, který píše pouze deseti prsty zpaměti, musí neustále zrakem kontrolovat, zda neudělal chybu. Pokud si všimne, že chybu udělal, je nucen psaní přerušit a vědomě se soustředit na její opravu, čímž dochází ke ztrátě koncentrace a navazující myšlenky. Informace uvedené v tomto odstavci jsou čerpány z [7].

Výuka hmatové metody se zabývá nejen správným prstokladem, ale i správným sezením, technikou tisknutí kláves, držením rukou, psaním naslepo a především ovládáním klávesnice z podvědomí.

2.1 Správná poloha prstů, rukou a nohou

Prsty jsou při psaní rozmístěny do základní polohy na střední písemné řadě a každý prst zaujímá na alfanumerické klávesnici místo podle následující tabulky 2.1. Při psaní se pohybuje vždy pouze jedním prstem. Ostatní prsty čekají v základní poloze, dokud se prst nevrátí zpět do základní polohy. Prsty se jemně dotýkají kláves bříšky, aby nedošlo k nechtěnému stisknutí klávesy. Při pohybu prsty po klávesnici lehce klouzají z jedné klávesy na druhou. Paže jsou při psaní volně podél těla v lokti ohnuté do mírně stoupajícího úhlu a nepohybují se. Dlaně se nedotýkají klávesnice ani stolu, protože by omezovaly pohyb a dosah prstů. Zápěstí je při psaní v rovině s rukou. Nohy by neměly být ve vrchní části překříženy, křížení v oblasti kotníků je možné. Chodidla by se měla dotýkat země celou svou plochou. Většina poznatků tohoto odstavce je převzata z [6].

Prst	**Ruka**	
	levá	**pravá**
ukazováček	F	J
prostředníček	D	K
prsteníček	S	L
malíček	A	Ů
palec	mezerník	

Tabulka 2.1: Rozmístění prstů na klávesnici

2.2 Výhody hmatové metody

- zrychlení ovládání klávesnice
- zmenšení chybovosti
- odpadá nutnost neustálého vyhledávání znaků na klávesnici, což nejenom zrychluje samotné psaní, ale dochází také k menšímu namáhání očí (oči se nemusí neustále přemisťovat z obrazovky na klávesnici a přeostřovat)
- rozložení námahy rovnoměrně na všechny prsty
- ochrana zdraví (zraku, páteře, zamezení vzniku karpálního tunelu)

2.3 Nevýhody hmatové metody

- ze začátku nezáživná výuka
- dlouhá doba výuky

Kapitola 3

Existující programy

Tato kapitola stručně shrnuje vlastnosti zkoumaných programů. Obsahuje zamyšlení nad použitím některých prvků a obrázky uživatelských rozhraní jednotlivých aplikací. Je třeba podotknout, že některé programy jsou testovány v demoverzi, což ovšem nemá vliv na jejich funkčnost a systém testování uživatelů.

3.1 Deseti prsty

Základem tohoto programu je jednoduché a přehledné grafické uživatelské prostředí. Ve spodní části je zobrazena klávesnice a ruce, kde se zvýrazňuje klávesa a prst, kterým má být klávesa stisknuta. Ve vrchní části můžeme nalézt informace o rychlosti, chybovosti a času opisování. Prvek kurzoru zobrazuje aktuální pozici v textu. Písmena opisovaného textu jsou obarvena podle prstu, který má stisknout klávesu reprezentující opisované písmeno. Lekce jsou uživateli zpřístupňovány postupně, ale lze je přeskakovat.

Dalším prvkem v této aplikaci je metronom, což je nástroj určující rytmus, který je pro nácvik hmatové metody velmi důležitý. Pokud totiž opakovaně provádíme stejnou fyzickou akci, naše podvědomí si tuto akci snadněji zapamatuje a udrží tuto informaci v paměti. Program také dále obsahuje propracovanou statistiku včetně grafu rychlosti, uživatelské účty, možnost opisovat vlastní lekce a umožňuje měnit vzhled.

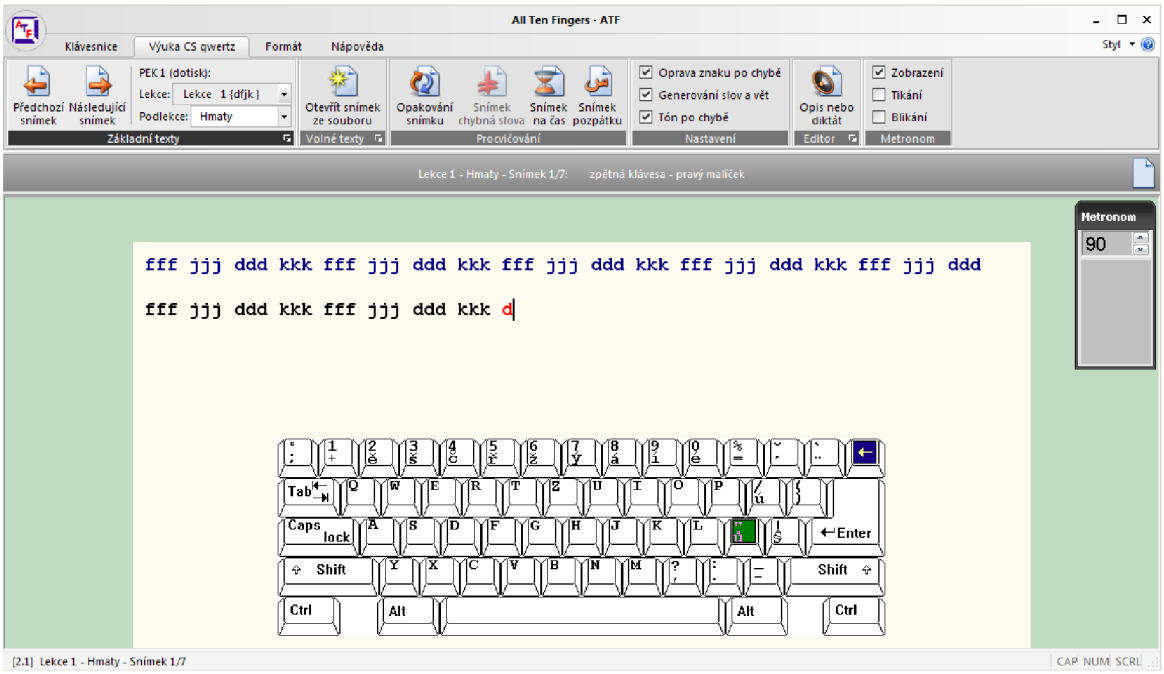
Jedná se o program lokalizovaný do češtiny. Lze jej stáhnout od výrobce na [10]. Program byl testován ve verzi 5.2.7. Grafické uživatelské rozhraní je zobrazeno na obrázku 3.1.

Obrázek 3.1: GUI aplikace Deseti prsty

3.2 All ten fingers

Aplikace působí na první pohled přehledně. Uživatelské prostředí je jednoduché, ve spodní části je zobrazena klávesnice, která zvýrazňuje klávesy. Chybí zde podpora uživatelských účtů, takže uživatelé musí sdílet jeden účet. Nechybí vlastní lekce a metronom, který určuje tempo nejen zvukovými signály, ale i problikáváním. U výběru lekce je uvedeno, které znaky se v ní vyskytují. Pozdější lekce jsou rozděleny do tří podlekcí – znaky, slova a věty. U každé lekce je možnost zvolit opisování na čas, ovšem čas není nikde zobrazen. Statistika lekce obsahuje rychlost a přesnost opisu, zobrazuje nejpomalejší a chybné znaky. Limit pro rychlost a přesnost opisu si může uživatel sám navolit. Pozice v textu je určena kurzorem.

Program lze získat od výrobce z [1]. Aplikace je lokalizována do češtiny. Testována byla verze 8.3. Grafické rozhraní je znázorněno na obrázku 3.2.



Obrázek 3.2: GUI aplikace All ten fingers

3.3 Type inspector

Způsob opisování textu se podobá předchozím aplikacím. Uživatel vidí pouze řádek nebo dva řádky textu, který opisuje. Program by mohl zobrazovat řádků více, aby uživatele učil orientovat se v delším textu. Opět se zde vyskytuje prvek klávesnice s zvýrazňováním právě opisovaného písmene a kurzor. Součástí aplikace je také metronom, uživatelské účty a objevují se i hry, které jsou založeny na opisování textu.

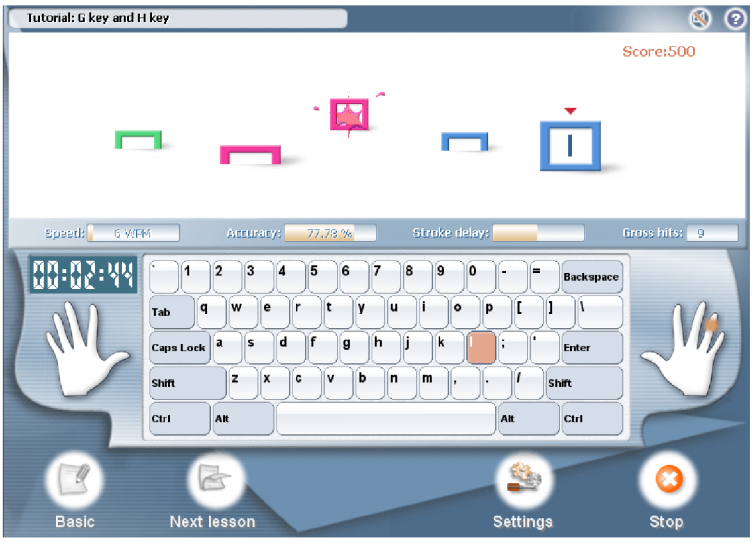
Program je lokalizován v anglickém jazyce. Testována byla verze 1.0.148.0. Grafické uživatelské rozhraní je znázorněno na obrázku 3.3. Oficiální stránky výrobce jsou dlohodobě nedostupné.

Obrázek 3.3: GUI aplikace Type inspector

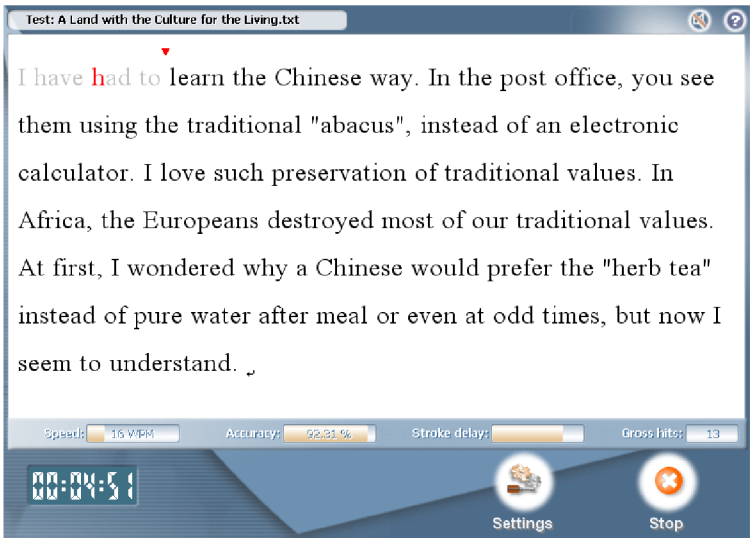
3.4 Typing tutor

Nácvik jednotlivých písmen v lekcích se zde liší od způsobu testování. V lekcích se objevuje prvek klávesnice se zvýrazňování písmen a opisovaná písmena se na obrazovce zobrazují v barevných čtvercích, které nejsou v jedné lince. Tento způsob tréninku příliš neodpovídá reálné situaci opisu textu. Při testování se však prvek klávesnice se zvýrazňováním písmen ztrácí a uživatel již vidí pouze opisovaný text a jednoduchou statistiku opisu. Po vyhodnocení testu se objevují doporučené kroky, které uživatele vedou dále, a podrobnější statistika. Opět se vyskytuje prvek kurzoru.

Program byl testován ve verzi 2.7. Program je lokalizován do angličtiny. Nejnovější verzi programu je možné získat od výrobce z [3]. Grafické uživatelské rozhraní je znázorněno na obrázcích 3.4 a 3.5.



Obrázek 3.4: GUI aplikace Typing tutor – Lekce



Obrázek 3.5: GUI aplikace Typing tutor – Testy

3.5 Shrnutí

Ve všech zkoumaných aplikacích se vyskytuje prvek klávesnice se zvýrazňováním klávesy, která má být stisknuta. Otázkou je, zda se jedná o vhodný prvek. Uživatelova pozornost neustále směřuje na tento prvek a uživatel je schopen napsat celou lekci tím, že bude pouze pozorovat a stlačovat zvýrazňované klávesy. Další možností je, že uživatel bude pohybovat očima z opisovaného textu na zvýrazňované klávesy, čímž bude docházet k zvýšené námaze očí. Zkušenějším uživatelům, kteří již znají správný prstoklad, může prvek připadat nadbytečný. Lepší variantou by bylo uživateli pouze sdělit, kterým prstem má stlačit jakou

klávesu a nutit ho tak vytvářet si vazby mezi prsty a písmeny aniž by mu to program stále ukazoval.

Dalším prvkem vyskytujícím se ve všech aplikacích je kurzor. Kurzor ukazuje uživateli aktuální pozici v textu. Uživatel je však nucen kurzor neustále sledovat a opisovat písmeno, na jehož pozici se nachází kurzor.

Představme si nyní situaci, kdy uživatel opisuje slovo a při opisování udělá uprostřed slova chybu a vloží do něj písmeno navíc. Pokud by uživatel toto slovo dopsal a nesledoval kurzor, od chybně vloženého znaku by se všechny ostatní znaky počítaly jako chyba, jelikož by byl kurzor o jednu pozici napřed. Obdobně je to při vynechání jednoho písmene. Další možností je, že se kurzor zastaví na místě a čeká, dokud uživatel nezadá správné písmeno a narušuje tím plynulost opisu. Otázkou také je, zda je kurzor vhodným návykem pro uživatele, protože uživatel si neprocvičuje orientaci v textu. V reálné situaci nám při opisu z papíru také nebude kurzor ukazovat aktuální pozici v opisovaném textu.

Kapitola 4

Návrh a implementace

Pro vytvoření výsledné aplikace bychom měli zvolit vývojové prostředí, které je vhodné pro řešení zkoumaného problému. Aby byl výsledek co nejlepší, měli bychom zvolit takové prostředí, se kterým máme zkušenosti a které používá programovací jazyk nám známý.

Zkušenosti autora této práce vycházejí z programovacích jazyků C a C++, proto byl pro tvorbu aplikace zvolen jazyk C++. Jako vývojové prostředí byl zvolen Qt Creator verze 1.2.1, který vychází z Qt verze 4.5.3. S tímto prostředím má autor zkušenosti a splňuje cíl přenositelnosti programu na více operačních systémů. Program je primárně vyvíjen pod operačním systémem Windows XP, ale cílem je jeho spuštění také na OS Linux.

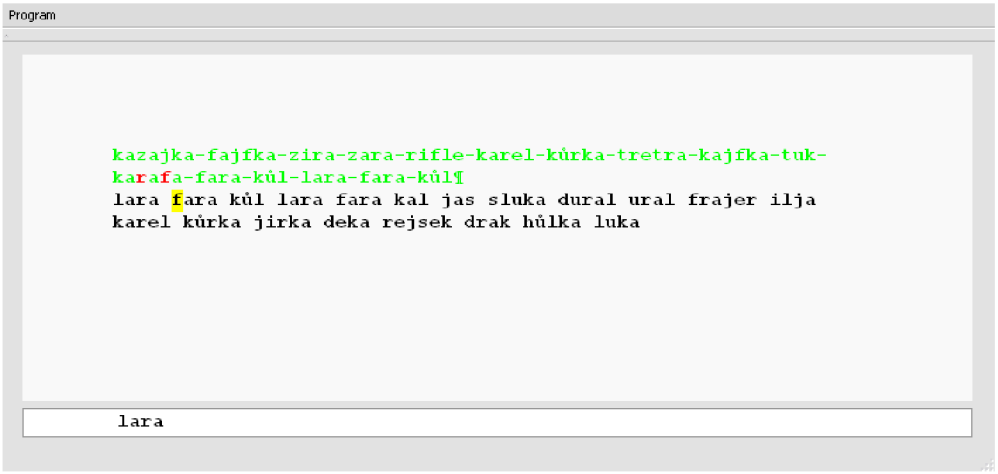
Qt je multiplatformní framework, který se používá nejen pro tvorbu programů s grafickým uživatelským rozhraním. Qt je napsáno v C++ a používá vlastní renderovací jádro, které dokáže emulovat vzhled systému, na kterém běží. Qt umožňuje pracovat s SQL databázemi, parsovat XML dokumenty, pracovat s vlákny a mnohé další. Qt podporuje tyto platformy: Embedded Linux, Mac OS X, Windows, Linux/X11, Windows CE/Mobile, Symbian a Maemo.

4.1 Navrhované prvky uživatelského rozhraní

Hlavní snahou při návrhu uživatelského rozhraní aplikace je vymyslet inovativní prvky, které se nevyskytují v ostatních programech a které by mohly zlepšit výuku hmatové metody. Dále je zde možnost vylepšit již existující prvky a zvýšit tak efektivitu učení hmatové metody.

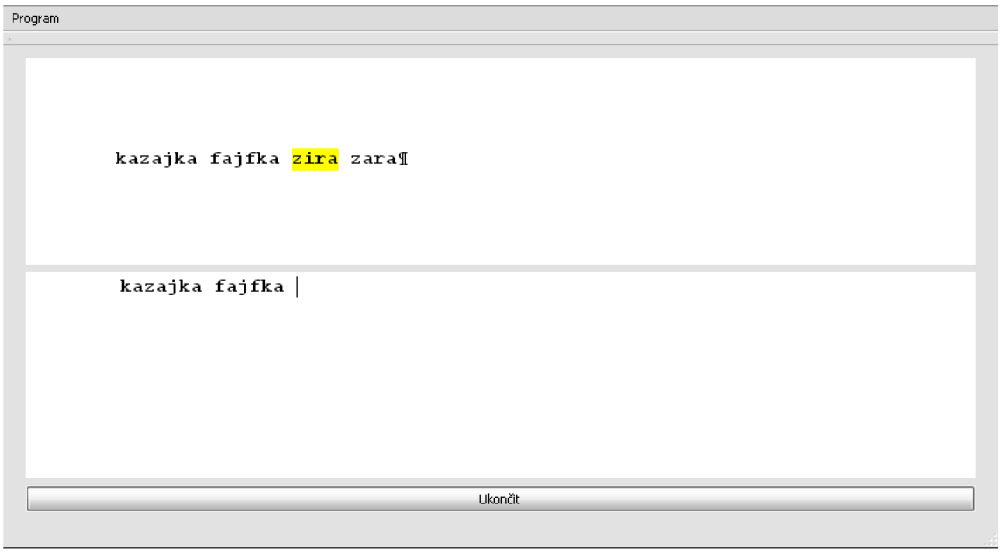
4.1.1 Opisování textu s využitím kurzoru

Jak již bylo popsáno výše, techniku opisování s využitím kurzoru používají všechny zkoumané aplikace. Zlepšení výuky by mohlo nastat, pokud bychom nezobrazovali chyby okamžitě, ale nechali bychom uživatele opsat celý řádek a teprve poté bychom vyznačili chyby. Nedocházelo by tak ke znejistění uživatele při každé chybě. Dále bychom mohli po zjištění chyby na řádku uživatele nechat opisovat celý řádek znovu, dokud by nebyl řádek opsán správně. Rozhodně bychom neměli kurzor zastavovat a čekat, dokud uživatel chybu opraví, což by vedlo ke ztrátě plynulosti opisu. Opravování chyb v opisovaném textu by nemělo být vůbec dovoleno. Uživatel by si neměl zvykat chyby opravovat a měl by být veden k bezchybnému opisu. Návrh uživatelského rozhraní s opisováním pomocí kurzoru po písmenech je zobrazen na obrázku 4.1. Některé zde uvedené zásady lze nalézt v [9].



Obrázek 4.1: Návrh uživatelského rozhraní s použitím kurzoru po písmenech

Možnost vylepšení tohoto způsobu opisování je použití kurzoru, který by se posouval po celých slovech. Předešlo by se tím problému nadbytečných či chybějících písmen v opsaném slovu. Uživatel by získal větší volnost při opisování. Problémem této možnosti by bylo programově zachytit, na kterém slovu se uživatel aktuálně nachází. Způsob zjišťování pozice kurzoru v navrhovaném uživatelském rozhraní (obrázek 4.2) je závislý na počtu mezer zadaných uživatelem. Pokud uživatel zadá mezeru, přeskočí kurzor na další slovo. Pokud ale uživatel vynechá mezeru mezi slovy a nevšimne si toho, kurzor zůstane na prvním slovu a i když bude uživatel psát další slovo správně, bude se mu počítat do chyb.



Obrázek 4.2: Návrh uživatelského rozhraní s použitím kurzoru po slovech

Jelikož uvažovaná vylepšení nejsou nijak zásadní pro zlepšení výuky hmatové metody, nakonec od nich bylo upuštěno.

4.1.2 Opisování textu bez využití kurzoru

Opisování textu bez využití kurzoru nevyužívá žádná ze zkoumaných aplikací, což otevírá prostor pro inovaci. Kontrola chyb při opisování textu je zde náročnější než u opisování textu s kurzorem. V navrhovaném uživatelském rozhraní je porovnávání zdrojového textu a textu napsaného uživatelem řešeno až po ukončení opisu lekce. Nemůžeme však kontrolovat a porovnávat tyto texty písmeno po písmenu, protože nevíme, zda uživatel v některých slovech nepřidal písmena navíc nebo naopak mu nějaká písmena nechybí. Také nevíme, zda uživatel některá slova úplně nevynechal nebo neopsal vícekrát. Musíme proto zvolit algoritmus, který je schopen porovnat dva řetězce a vrátit počet znaků, ve kterých se oba řetězce liší. V návrhu je zvolen Levenshteinův algoritmus, který bude popsán podrobněji v podkapitole 4.3.1

Protože opisování bez využití kurzoru přináší nejvíce prostoru pro inovaci výuky hmatové metody psaní, bylo toto uživatelské prostředí zvoleno výchozím pro tvorbu výsledné aplikace.

4.2 Práce s uživatelem

Mezi hlavní cíle výsledné aplikace patří, aby se uživatel učil správným zásadám hmatové metody aniž by musel studovat materiály zabývající se touto tématikou. Aplikace by měla uživatele vést, radit mu jak dále postupovat a zobrazovat statistiku opisu jednotlivých lekcí.

4.2.1 Systém testování a hodnocení uživatele

Ve výsledném programu je testování uživatele rozděleno do dvou částí – lekcí a testů. Tento systém, je převzatý ze zdroje [9]. Uživateli jsou pro opisování nabídnuty všechny lekce a testy po první nedokončený test. Teprve po dokončení testu se zobrazí další lekce a další test. Výhodou je, že uživatel znalý základů hmatové metody nemusí nutně absolvovat všechny úvodní lekce, ale pouze test. Pokud neprojde úvodním testem, je to pro něj známkou, že jeho schopnosti neodpovídají požadavkům programu a je třeba se více procvičit. Díky tomuto způsobu postupného zobrazování lekcí a testů je nedočkavým uživatelům zabráněno spustit si závěrečný test a cvičit ho neustále dokola až do jeho splnění. Uživatel by tímto mylně nabyl dojmu, že již ovládá plně hmatovou metodu psaní, protože splnil nejnáročnější test. Došlo by také ke ztrátě motivace pracovat s programem, která je pro udržení zájmu uživatele pracovat s programem velmi důležitá. Uživatel by si mohl pomyslet, že již dosáhl toho nejvyššího cíle a jeho zájem o program by opadl.

V lekcích se uživatel učí používat nové klávesy. Po vybrání lekce se uživateli zobrazí instrukce, kterým prstem má stlačit klávesy, které se v dané lekci procvičují. Jednotlivé lekce na sebe navazují a platí, že pozdější lekce v sobě obsahují nácvik písmen z dřívějších lekcí v kombinaci s aktuálními písmeny. Aby došlo k plnému osvojení nových písmen, musí uživatel napsat celou lekci bezchybně. Teprve potom je lekce označena jako správně opsaná a uživateli je nabídnut přechod do další lekce.

Testy v sobě obsahují písmena, která se vyskytují ve všech předcházejících lekcích. V testech se již nevyskytují instrukce, které by uživateli radily se stiskem kláves. Uživatel může v testech udělat několik chyb (narozdíl od lekcí, kde chyby dovoleny nejsou), ale počet chyb se projeví v celkovém hodnocení.

Uživatel může při opisu lekce či testu zadat maximálně třikrát delší text než je délka textu, který opisuje. Pokud dosáhne této hranice, je opisování automaticky ukončeno. Ma-

ximální doba opisu je čtyřiadvacet hodin.

Předtím než přejdeme k hodnocení lekcí a testů, musíme si nejprve ujasnit, co počítáme za chybu:

- chybějící znak
- nesprávný znak
- znak navíc
- použití klávesy **Enter**

Dále budeme potřebovat vzorce (převzaté z [9]) pro výpočet míry přesnosti opisu (rovnice 4.1) a výpočet čistých úhozů za minutu (rovnice 4.2).

$$MPO = 100 - \left(E * \frac{100}{HU} \right) \tag{4.1}$$

$$CUmin = \frac{HU - (E * 50)}{t} \tag{4.2}$$

Legenda:

- E – chyby v opsanám textu
- t – doba opisu
- HU – hrubé úhozy
- MPO – míra přesnosti opisu
- $CUmin$ – čisté úhozy za minutu

Hodnocení testů se liší od hodnocení lekcí. Lekce hodnotíme podle následující jednoduché tabulky 4.1. Hodnocení testů zohledňuje dva faktory – rychlost opisu a přesnost opisu.

Počet chyb	**Hodnocení**
0	Prošel
vetší než 0	Neprošel

Tabulka 4.1: Hodnocení lekcí

Pro hodnocení rychlosti se používá tabulka 4.2. Hodnocení dle přesnosti opisu vychází z tabulky 4.3. Obě tabulky jsou převzaté z [9]. Celková známka má vždy horší hodnotu z obou faktorů.

Hodnocení	do 15. lekce	do 25. lekce	do 35. lekce	do 45. lekce	od 46. lekce
Výborný	80 a více	110 a více	140 a více	170 a více	200 a více
Chvalitebný	70	100	130	160	190
Dobrý	60	90	120	150	180
Dostatečný	50	80	110	140	170
Nedostatečný	méně				

Tabulka 4.2: Hodnocení dle rychlosti opisu (čisté úhozy)

Hodnocení	Míra přesnosti opisu
Výborný	99,8%
Chvalitebný	99,6%
Dobrý	99,4%
Dostatečný	99,2%
Nedostatečný	méně

Tabulka 4.3: Hodnocení dle přesnosti opisu

4.2.2 Systém vedení uživatele

Ve výsledné aplikaci je implementován systém navigování uživatele pomocí doporučených kroků. V aplikaci se rozlišují tyto základní doporučené kroky:

1. přechod do lekce, kde se vyskytuje písmeno, ve kterém bylo při opisu uděláno nejvíce chyb
2. opakování lekce
3. přesun na další lekci
4. přesun na další lekci, která ještě nebyla zhotovena (přeskočení hotových lekcí)

Možnost 2 je nabídnuta vždy. Možnost 1 je nabízena, pokud udělal uživatel chybu v písmenu, které se vyskytuje v jedné z předchozích lekcí. Možnost 3 je nabídnuta, pokud uživatel projde lekcí či testem a existuje ještě další lekce. Možnost 4 je nabídnuta, pokud uživatel projde lekcí či testem a existuje další nezhotovená lekce či test, na které je možno skočit. Pokud uživatel lekci či test neprojde, je mu nabídnuta pouze možnost 1 a 2.

4.2.3 Statistika

Statistika ve výsledné aplikaci se skládá ze tří částí. V první části jsou uživateli zobrazeny informace o rychlosti opisu, přesnosti opisu, celkovém počtu hrubých a čistých úhozů, počtu chyb, chybovosti a výsledném hodnocení.

Ve druhé části statistiky je zobrazena tabulka nesoucí informace o chybovosti jednotlivých prstů. V aplikaci se vypočte, jaký je počet stisků každého prstu ve zdrojovém textu a jaký je počet stisků každého prstu v opsaném textu a z těchto údajů se získá chybovost každého prstu. Protože rozložení kláves na klávesnici není vždy stejné, je potřeba uvažovat více rozložení. V aplikaci jsou podporována dvě rozložení – české a americké. Pokud je zjištěno jiné rozložení klávesnice než české nebo americké, uvažuje se české rozložení.

Třetí část obsahuje graf rychlosti opisu. Uživatel z tohoto grafu rozezná jaké bylo tempo jeho opisu. Čím více připomíná výsledný graf přímku, tím bylo tempo rovnoměrnější.

4.2.4 Systém nápovědy programu

Nápověda se v programu vyskytuje v podobě krátkých informací v textových štítcích. Cílem tohoto řešení je, aby uživatel nemusel před použitím programu číst rozsáhlý manuál k jeho používání. Hlavní nápověda se vyskytuje v pravé části hlavního okna programu a obsahuje informace týkající se vybrané lekce.

4.3 Použité algoritmy a výpočty

Tato podkapitola podrobněji popisuje použité algoritmy a výpočty ve výsledném programu. Pro výpočet chyb, kterých se uživatel dopustil při opisu textu se používá Levenshteinava vzdálenost popsaná v 4.3.1. Do programu byl implementován algoritmus pro zobrazení fyzického rozměru šířky A4 na monitoru popsaný v 4.3.2. Problém posouvání dlouhých textů, které přesahují velikost okna, v němž je text zobrazen, je řešen v 4.3.3. Formátování textů v programu je vysvětleno v 4.3.4.

4.3.1 Levenshteinova vzdálenost

Levenshteinova vzdálenost označuje míru podobnosti dvou řetězců. Jeden z řetězců je označen jako zdrojový, druhý jako cílový. V našem případě bude zdrojovým řetězcem text lekce určený k opsání, cílovým řetězcem bude text napsaný uživatelem. Vzdálenost označuje celkový počet operací vložení znaku, mazání znaku a substituce znaku potřebný k transformaci jednoho řetězce na druhý. Čím větší je Levenshteinova vzdálenost mezi oběma řetězci, tím více se řetězce liší.

Princip algoritmu je založen na sestavení matice. Pseudokód algoritmu je znazorněn v tabulce 4.4. Protože budeme v cílovém řetězci potřebovat vyznačit uživatelovy chyby, budeme provádět transformaci cílového řetězce. Proto bude počet sloupců matice odpovídat počtu znaků v cílovém řetězci zvětšeným o jedna. Počet řádků matice je v použitém algoritmu dán počtem znaků v zdrojovém řetězci zvětšeným o jedna. Řádky matice jsou indexovány znaky zdrojového řetězce, sloupce jsou indexovány znaky cílového řetězce.

Pohyb v matici se řídí několika pravidly. Pokud je znak z cílového řetězce na indexu `m` roven znaku ze zdrojového řetězce na indexu `n`, použije se operace identita, která odpovídá diagonálnímu pohybu v matici. Pokud se znaky na průsečíku textů `(m, n)` nerovnají, je potřeba editovat cílový řetězec a provést jednu z následujících transformací:

- **mazání znaku** z cílového řetězce, což odpovídá horizontálnímu pohybu v matici
- **vložení znaku** do cílového řetězce, což odpovídá vertikálnímu pohybu v matici
- **substituce znaku** v cílovém řetězci, což odpovídá diagonálnímu pohybu v tabulce

Do aktuálního místa v matici se vždy dosadí hodnota, která odpovídá té transformaci, jejíž provedení bude mít pro dané místo v matici minimální hodnotu.

```
int LevenshteinDistance(QString zdroj, QString cíl)
{
  \\ Krok 1
  n = délka zdroje (zdrojového řetězce);
  m = délka cíle (cíloveho řetězce);
  if(n == 0) return m;
  if(m == 0) return n;
  vytvoř matici M[n][m];
  \\ Krok 2
  M[n][0] = cena za mazání znaku vynásobená indexem řádku;
  M[0][m] = cena za vložení znaku vynásobená indexem řádku;
  \\ Krok 3
  for (i = 0; i <= n; i++)
  {
    \\ Krok 4
    for (j = 0; j <= m; j++)
    {
      \\ Krok 5
      if(zdroj[i] == cíl[j]) váha = 0;
      else váha == 1;
      \\ Krok 6
      above = M[i-1][j]; \\vertikání pohyb
      left = M[i][j-1]; \\horizontální pohyb
      diag = M[i-1][j-1]; \\diagonální pohyb
      M[i][j] = minimum (above + 1, left + 1, diag + váha);
    }
  }
  \\ Krok 7
  return M[n][m];
}
```

Tabulka 4.4: Pseudokód Levenshteinova algoritmu

Abychom mohli začít matici vyplňovat, je třeba si nejprve určit ceny za vložení, mazání a substituci znaku. Vyplňování matice začíná v levém horním rohu matice. Nultý řádek matice odpovídá ceně vložení znaku vynásobené indexem sloupce, ve kterém se nacházíme. Nultý sloupec odpovídá ceně mazání znaku vynásobené indexem řádku, na kterém se nacházíme. Zbytek matice se vyplní tak, že pro každou buňku matice se vyhledá nejmenší hodnota, kterou je možno dosáhnout vložením, mazáním nebo substitucí znaku. Matici vyplňujeme po řádcích. Výslednou hodnotu, která označuje počet odlišností ve dvou řetězcích, nalezneme v pravém dolním rohu matice.

Princip tvorby matice je zřejmý z následujícího příkladu. Za zdrojový řetězec považujme slovo `kazajka`. Uživatel při opisu zadá slovo `kaazojka`, což značí cílový řetězec. Cíl se liší od zdroje ve dvou znacích, uživatel tedy udělal 2 chyby. Uvažujme cenu za mazání, za vložení a za substituci rovnou 1. Průběh vyplňování matice je naznačen v tabulce 4.5 a 4.6. Výsledná matice je znázorněna v tabulce 4.7.

		k	**a**	**a**	**z**	**o**	**j**	**k**	**a**
	0	1	2	3	4	5	6	7	8
k	1								
a	2								
z	3								
a	4								
j	5								
k	6								
a	7								

Tabulka 4.5: Vyplnění nultého řádku a sloupce

		k	**a**	**a**	**z**	**o**	**j**	**k**	**a**
	0	1	2	3	4	5	6	7	8
k	1	0	1	2	3	4	5	6	7
a	2								
z	3								
a	4								
j	5								
k	6								
a	7								

Tabulka 4.6: Vyplnění prvního řádku

Abychom mohli v cílovém řetězci vyznačit chyby, musíme výslednou matici projít zpětně. Tím zjistíme, jaké operace se provedli nad cílovým řetězcem. Průchod matice začíná v pravém dolním rohu. Vybíráme vždy ze tří možných pohybů – nahoru, doleva a diagonálně. Pohneme se tím směrem, kde se v matici nachází minimální hodnota. Zpětný průchod maticí je vyznačen v tabulce 4.8.

U každého diagonálního pohybu porovnáváme hodnotu místa v matici, ze kterého jsme se pohnuli, s místem, na které jsme se pohnuli. Pokud nejsou hodnoty stejné, jedná se o chybu a proto je nutné znak cílového řetězce zvýraznit. V příkladu také vidíme, že došlo i k pohybu horizontálnímu. Jak již bylo vysvětleno výše, horizontální pohyb značí mazání znaku. Znamená to, že znak v cílovém řetězci je nadbytečný a proto ho můžeme označit za chybný.

Levenshteinovu vzdálenost vymyslel v roce 1965 ruský vědec Vladimir Levenshtein. Dnes se používá pro hledání podobných řetězců DNA, pro kontrolu pravopisu nebo například pro odhalování plagiátů. Některé definice uvedené v této podkapitole lze nalézt zde [2], které byly ověřeny v [5] a v [11]. V programu byl použit zápis algoritmu ze zdroje [4].

		k	**a**	**a**	**z**	**o**	**j**	**k**	**a**
	0	1	2	3	4	5	6	7	8
k	1	0	1	2	3	4	5	6	7
a	2	1	0	1	2	3	4	5	6
z	3	2	1	1	1	2	3	4	5
a	4	3	2	1	2	2	3	4	4
j	5	4	3	2	2	3	2	3	4
k	6	5	4	3	3	3	3	2	3
a	7	6	5	4	4	4	4	3	2

Tabulka 4.7: Výsledná matice

		k	**a**	**a**	**z**	**o**	**j**	**k**	**a**
	0	1	2	3	4	5	6	7	8
k	1	**0**	1	2	3	4	5	6	7
a	2	1	**0**	**1**	2	3	4	5	6
z	3	2	1	1	**1**	2	3	4	5
a	4	3	2	1	2	**2**	3	4	4
j	5	4	3	2	2	3	**2**	3	4
k	6	5	4	3	3	3	3	**2**	3
a	7	6	5	4	4	4	4	3	**2**

Tabulka 4.8: Zpětné projití matice

4.3.2 Rozměr A4

Pro dosažení co nejreálnější situace, kdy uživatel opisuje text z papíru, byl do výsledné aplikace implementován systém, který dokáže na obrazovku zobrazit skutečný rozměr šířky A4. Výhodou pro uživatele je možnost učit se lépe orientovat v textu napsaném na papíře o formátu A4.

Abychom dosáhli skutečného rozměru šířky A4, musíme nastavit šířku (zadanou v pixelech) požadoveného prvku programu v závislosti na DPI. DPI označuje počet pixelů na palec. Při vývoji programu bylo snahou zjistit DPI programově. Qt nabízí metody `logicalDpiX()` a `physicalDpiX()` pro zjištění logického a fyzického DPI monitoru na ose X. Po otestování těchto metod na více rozlišeních a více monitorech se však ukázalo, že tyto metody nevracejí spolehlivé výsledky a proto nebyly použity. Dalším způsobem jak dosáhnout A4 rozměru je dopočítat ho pomocí fyzických rozměrů monitoru a rozlišení. Nabízela se možnost využití metod `widthMM()` a `heightMM()`, které by měly vrátit šířku a výšku monitoru v milimetrech, ovšem opět nebylo dosaženo spolehlivých výsledků. Programového zjištění rozlišení bylo dosaženo.

Výsledným řešením je vytvoření dialogu, ve kterém uživatel zadá úhlopříčku monitoru a poměr stran. Díky těmto údajům můžeme pomocí Pythagorovy věty dopočíst fyzické rozměry monitoru. Pomocí programově zjištěného rozlišení vypočteme počet zobrazených pixelů na milimetr. Když už známe počet pixelů na milimetr, vynásobíme ho požadovanou šířkou v milimetrech a získáme výslednou šířku v pixelech potřebnou pro zobrazení reálné šírky A4 formátu.

Ve výsledné aplikaci je uvažována i možnost, že uživatel nezná rozměr úhlopříčky nebo

poměr stran svého monitoru. V takovém případě je uživateli nabídnuta možnost pro nastavení rozměru šířky A4 ručně, kdy uživatel nastaví šířku pomocného okna programu podle papíru A4 přiloženého k obrazovce.

Jelikož na menší monitory by se nevlezl formát A4, musíme u malých monitorů nastavovat konstantní šířku okna v pixelech. Skutečného rozměru šířky A4 tedy není dosaženo, ale funkčnost programu není nijak ovlivněna. Za malý monitor je v aplikaci považován monitor s úhlopříčkou menší patnáct palců. Důvodem je, že monitory menší patnácti palců nebyly při tvorbě programu k dispozici.

4.3.3 Posouvání při opisování dlouhých textů

Při opisování lekce či testu je text zobrazen v oblasti s omezenou výškou. Výška oblasti je omezena velikostí zobrazovací plochy monitoru. Pokud se již text do oblasti nevejde, zobrazí se posuvník. Protože je nepřípustné, aby byl uživatel nucen posouvat posuvníkem během opisování textu, je nutné zajistit posouvání programově. Posouvání textu by nemělo být skokové, protože by to vedlo ke ztrátě orientace uživatele.

Posouvání je v programu realizováno pomocí dvou posuvníků a časovače. První posuvník se vztahuje k oblasti s opisovaným textem, druhý posuvník posouvá oblastí s textem, který píše uživatel. První posuvník má dvakrát větší rozsah než druhý posuvník. Oba posuvníky jsou mezi sebou propojeny tak, že první posuvník dorovnává hodnotu druhého posuvníku. Aby bylo posouvání plynulé, využívá se časovače, který posouvá prvním posuvníkem po malých úsecích za určitý čas. Čím větší je rozdíl hodnot mezi aktuálními pozicemi obou posuvníků, tím větší je krok, kterým se posunuje první posuvník.

4.3.4 Formátování textů

Textové widgety v Qt umožňují zobrazit *rich text*. *Rich text* je formát textu, do kterého je možné zapisovat některé HTML tagy. Widgety automaticky detekují HTML značkování, které je připojeno k textu. Výsledkem je zobrazení naformátovaného textu bez HTML značek. Pokud například budeme chtít nastavit text některého widgetu, který podporuje *rich text*, na tučný, použijeme řetězec ve tvaru `<b>Hello world!</b>`. Výsledkem bude zobrazení textu **Hello world!**.

V programu je formátování pomocí HTML tagů využito pro obarvení řetězce, který napsal uživatel při opisu zdrojového textu. Jednotlivé znaky v řetězci jsou formátovány pomocí tagu `font` a vlastnosti `color`. Pro formátování bílých znaků je použit tag `span` a vlastnost `background`, která umožní nastavit barvu pozadí u bílých znaků.

V programu existuje pět kategorií, u kterých je možno nastavovat barvu:

- správný znak
- špatný znak
- chybějící znak
- chybějící mezera
- chybějící Enter

4.4 Uložení dat

Pro uchovávání kontextu programu bylo nutné ukládat data do souboru na pevný disk. Díky ukládání dat do souboru na disku můžeme pracovat s uživatelskými účty, ukládat statistiku jednotlivých lekcí, ukládat změny v nastavení aplikace pro konkrétního uživatele *apod.*

Data programu jsou uložena v souboru ve formátu XML. Qt obsahuje modul pro snadnou práci se soubory ve formátu XML, což je výhodou. Pro zpracování XML je použit DOM parser. Po spuštění aplikace je otevřen soubor XML s uloženými daty a vytvořen obraz XML dokumentu v paměti. Výhodou tohoto řešení je, že nemusíme neustále otevírat a zavírat soubor s uloženými daty pokud potřebujeme zjistit, editovat či uložit informace v něm uložené. Teprve po ukončení programu je při volání destruktoru třídy pro práci s XML celý obraz uložen z paměti do souboru. V následující tabulce 4.9 můžeme vidět základní navrženou strukturu XML dokumentu.

```
<?xml version='1.0' encoding='UTF-8'?>
<users>
        <user>
              <name>jmeno</name>
              <password/>
              <monSizeX>331.703</monSizeX>
              <monSizeY>207.314</monSizeY>
              <style>blue.style</style>
              <colors>#00ff00;#ff0000;#ff0000;#ffff00;#ffff00</colors>
              <created>31.03.2010;19:01:15</created>
              <lessons>
                        <lesson id="9">
                        ⋮
                        </lesson>
                        ⋮
              </lessons>
        </user>
        ⋮
</users>
```

Tabulka 4.9: Základní struktura XML dokumentu

4.4.1 Uživatelské účty

Aby mohlo aplikaci používat více uživatelů nezávisle na výsledcích ostatních, byl implementován systém pro tvorbu uživatelských účtů. Jednotlivé uživatelské účty se do XML ukládají do tagů `<user>`. Po startu programu se projde celý strom XML v paměti a načtou se všechny existující uživatelské účty. Po vybrání účtu se ze souboru s daty načtou nastavení a dosažené výsledky pro konkrétní uživatelský účet.

Při tvorbě nového uživatelského účtu je možno zadat jeho heslo. Heslo je uloženo v XML souboru do tagu `<password>`. Aby nebylo heslo při prohlížení souboru na první pohled

čitelné, je jeho obsah převeden do formátu `base64`. Algoritmus `base64` převede každé tři znaky ASCII (což odpovídá třem bajtům) na čtyři kódované znaky ASCII. Pokud počet znaků nevychází přesně na trojici, zakóduje se zbývající počet znaků a přidá se znak `=`, který označuje chybějící znak.

4.4.2 Statistika lekcí a testů

Statistika všech lekcí či testů, které již uživatel splnil, je uložena do XML souboru. Programově tak snadno zjistíme, které lekce či testy můžeme označit jako hotové. U každého tagu `<lesson>` najdeme i identifikátor, jehož číslo určuje, ke které lekci se vztahuje uložená statistika. Pokud uživatel splní lekci nebo test, zjistí se, zda již není uložena statistika této lekce či testu v XML souboru. Pokud je statistika po splnění lekce lepší jako statistika v XML souboru, automaticky se přepíše. Pokud je však statistika po splnění lekce horší než statistika v XML, je uživatel dotázán, zda si přeje nahradit lepší statistiku tou horší.

Uložení statistiky lekce nebo testu se skládá ze čtyř tagů. Tag `<labelStat>` v sobě obsahuje textové informace, které se nastavují do jednotlivých textových štítků v programu. Jednotlivé informace jsou od sebe odděleny středníkem. Počet sekcí oddělených středníkem se nemění. Příklad uložených informací:

```
<labelsStat>Test1;8;8;240.0;0;0.0;00:00:02;100.0;
Výborný;Výborný;Výborný;</labelsStat>
```

Dalším tagem je `<fingerStat>`, který v sobě obsahuje informace o použitém rozložení klávesnice a informace pro vyplnění tabulky s chybovostí prstů. Informace jsou od sebě opět odděleny středníkem. Na prvním místě řetězce je informace o rozložení klávesnice. Pak již stačí cyklit po jednotlivých sekcích řetězce a v programu po řádcích vyplňovat tabulku. Počet sekcí řetězce se nemění. Příklad uložených informací:

```
<fingerStat> České;3;3;0;0.0%;0;0;0;0.0%;0;0;0;0.0%;0;0;0;0.0%;
0;0;0;0.0%;0;0;0;0.0%;0;0;0;0.0%;3;3;0;0.0%;2;2;0;0.0%;</fingerStat>
```

Tag `<graphStat>` v sobě nese informace o počtu hrubých úhozů za minutu v určitém časovém úseku a používá se pro vykreslení grafu rychlosti. Narozdíl od předchozích dvou tagů je počet sekcí oddělených středníkem v řetězci proměnlivý, maximálně však obsahuje dvacet sekcí. Abychom mohli začít vykreslovat souřadnice do grafu, musíme nejprve zjistit sekci s maximální hodnotou. Tímto krokem zjistíme, že délka osy Y odpovídá nalezené maximální hodnotě a můžeme začít nanášet ostatní souřadnice v poměru k maximální hodnotě. Dále musíme zjistit, jaký je krok na ose X. Tato informace je uložena v tagu `<graphModulo>` a označuje, po kolika sekundách byly hodnoty rychlosti zaznamenávány. Příklad uložení:

```
<graphStat>0000.0;0324.0;0360.0;0360.0;</graphStat>
<graphModulo>5</graphModulo>
```

4.4.3 Soubory lekcí a testů

Texty lekcí a testů nejsou uloženy v XML souboru, ale v samostatných textových souborech. Texty byly převzaty z [9] a slouží pouze pro demonstraci funkčnosti programu (pro distribuci programu a jeho širší použití je nutno navrhnout lekce a testy vlastní). Abychom rozeznali, jaké je pořadí lekcí a testů, byl vytvořen pomocný textový soubor. Pomocný soubor obsahuje seznam lekcí a testů zapsaný v tvaru uvedeném v tabulce 4.10.

```
Lekce!:F J!:lesson1.fit
Lekce!:D K!:lesson2.fit
Lekce!:A Ů!:lesson3.fit
⋮
Test1!:!:test1.fit
⋮
```

Tabulka 4.10: Obsah pomocného souboru

Roli oddělovače sekcí v řetězci zde plní vykřičník následovaný dvojtečkou. První sekce obsahuje informace o názvu lekce či testu. Ve druhé sekci nalezneme písmena, která se nacvičují v dané lekci. Druhá sekce se v programu používá při navádění uživatele do lekce, která obsahuje písmeno, ve kterém uživatel nejvíce chybuje. Ve třetí sekci je název souboru, který obsahuje zdrojový text lekce nebo testu. Výhodou pomocného souboru je, že můžeme snadno měnit pořadí, přidávat nové lekce a měnit názvy jednotlivých lekcí.

4.5 Nastavení programu

Snahou výsledného programu je nabídnout uživateli co nejvíc možností pro jeho přizpůsobení. Program umožňuje uživateli nastavovat vlastní vzhledy a barvy, zobrazování některých prvků a nebo nastavovat metronom.

4.5.1 Vzhledy

Qt umožňuje měnit vzhled jednotlivých widgetů v programu pomocí Qt Style Sheets, které jsou inspirované kaskádovými styly. Kaskádové styly se používají pro grafickou úpravu webových stránek. Syntaxe stylového předpisu se u kaskádových stylů zapisuje následovně:

```
selektor { deklarace vlastností }
selektor { deklarace vlastností }
...
```

Selektor určuje, na které elementy se pravidlo použije. Vlastnosti definují vizuální vlastnosti dotyčných elementů. Ne jinak je tomu u Qt Style Sheets. Selektor zde ale označuje, na který widget se pravidlo uplatní. Pokud bychom chtěli například obarvit všechna tlačítka programu červeně, použili bychom následující předpis:

```
QPushButton { background-color: red; }
```

Widgety, které umožňují nastavovat pozadí, okraje, vnitřní odsazení a lemování dodržují box model známý z CSS. Ve výsledném programu je navrženo několik základních vzhledů pro demostraci. K programu je i přiložena šablona, ve které jsou vypsané identifikátory jednotlivých widgetů aplikace. Uživatelé znalí Qt Style Sheets si tak mohou vytvářet vlastní vzhledy programu úpravou zmiňované šablony. Stačí pouze zkopírovat její obsah do textového souboru s koncovkou `.style` a vložit tento soubor do složky *Styles*.

4.5.2 Metronom

Metronom je důležitým prvkem implementovaným ve výsledné aplikaci. Umožňuje uživateli dodržovat pravidelné tempo. Při nácviku hmatové metody bychom se měli snažit psát v pravidelném tempu i za cenu snížení rychlosti psaní. Metronom programu umožňuje nastavit počet úderů za minutu. Uživatel dodržující toto tempo přesně ví, kolik hrubých úhozů napíše za minutu. V metronomu lze nastavit kombinaci až čtyř různých zvuků, které je možno střídat po jednom až deseti úderech. Metronom programu přehrává zvuky ve formátu WAV. Pokud chce uživatel do programu přidat vlastní zvuky, stačí nakopírovat WAV soubor do adresáře *Sounds* k ostatním zvukům. Doporučeno je používat krátké zvuky s okamžitým nástupem, protože delší zvuky se při vyšším počtu úderů za minutu nestihnou celé přehrát.

Kapitola 5

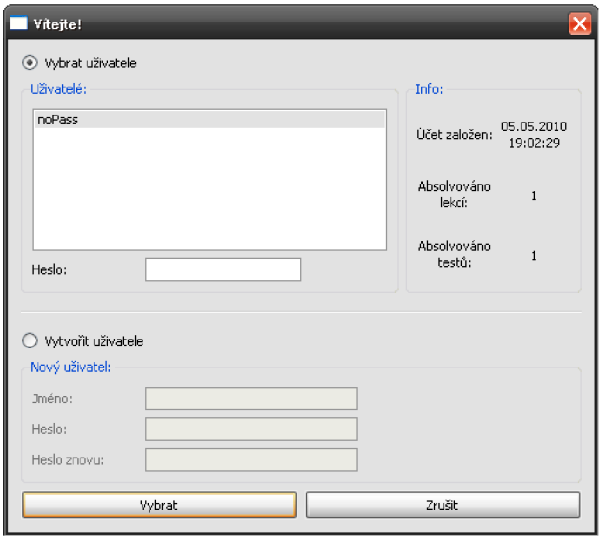
Vyhodnocení

Tato kapitola se zabývá vyhodnocením výsledného programu. V podkapitole 5.1 je popisováno výsledné uživatelské rozhraní programu. Podkapitola 5.2 shrnuje výsledky dotazníků. Podkapitola 5.3 se věnuje testování programu.

5.1 Výsledné uživatelské rozhraní

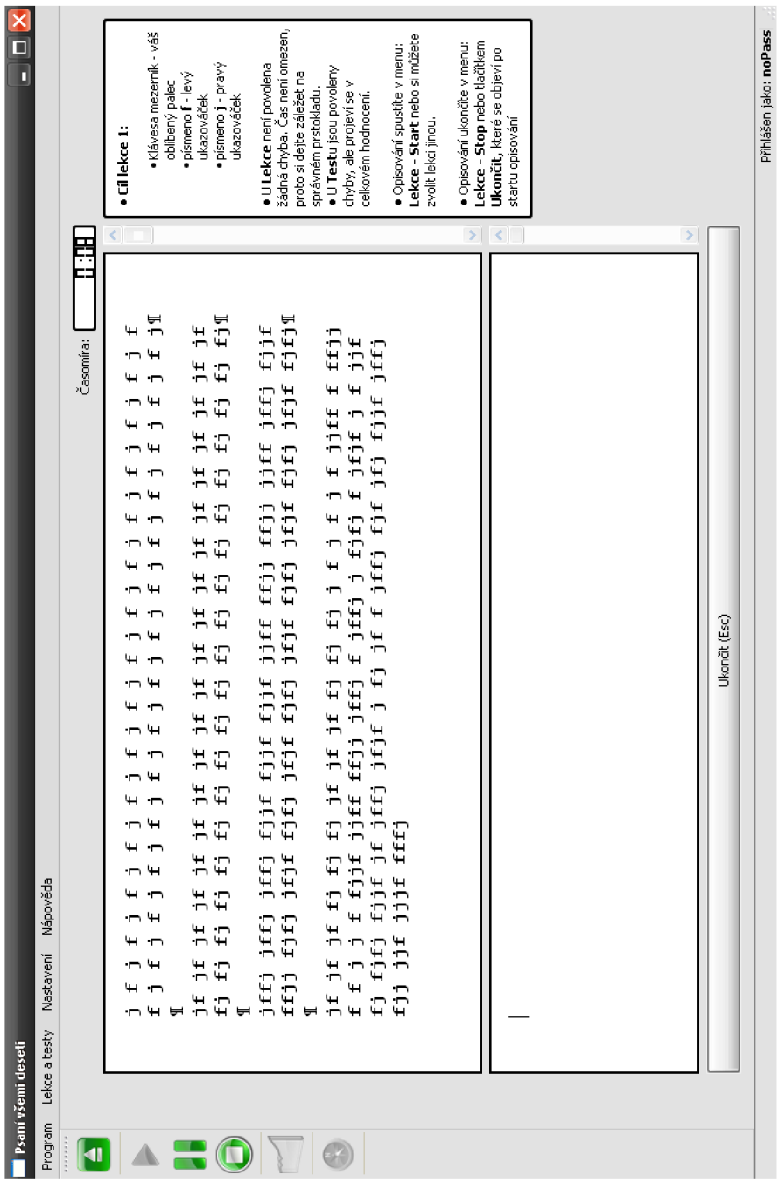
Cílem výsledného uživatelského rozhraní programu je, aby bylo co nejpřehlednější a na ovládání co nejintuitivnější. Důvodem je, aby program byli schopni ovládat i lidé méně zdatní při práci na počítači.

Úvodní obrazovka (obrázek 5.1) slouží pro výběr již existujícího uživatelského účtu nebo pro vytvoření nového účtu. U existujících účtů jsou zobrazeny základní informace – datum založení, počet hotových lekcí a počet hotových testů. Pokud uživatel vytváří nový uživatelský účet, další obrazovkou, která se mu zobrazí, je dialog pro zadání rozměrů monitoru. Teprve poté se zobrazí hlavní obrazovka programu. Pokud uživatel vybere již existující účet, zobrazí se rovnou hlavní obrazovka, protože informace o velikosti monitoru jsou již známy z XML souboru.



Obrázek 5.1: Úvodní obrazovka výsledného programu

Hlavní obrazovka programu je zobrazena na obrázku 5.2. Nejdůležitějšími prvky hlavního okna jsou dvě textové oblasti umístěné nad sebou. Ve vrchní textové oblasti se zobrazuje text určený k opisu a ve spodní se zobrazuje text opsaný uživatelem. Šířka těchto oblastí odpovídá šířce A4 papíru (pokud je možné ho fyzicky na monitoru zobrazit). Uživatelské rozhraní je navrženo tak, aby tyto textové oblasti zabíraly co nejvetší prostor hlavní obrazovky a nebyly omezovány ostatními prvky uživatelského rozhraní. Ve vrchní části obrazovky se nachází časomíra. Nápověda je umístěna vpravo. Toolbar s ikonami byl umístěn doleva, protože při umístění nahoře zabírá místo textovým oblastem, ale uživatel může jeho polohu měnit. Vpravo dole se po ukončení opisu objeví legenda. Prvky, které by zobrazovaly různé průběžné statistiky (*např.* rychlost opisu, počet chyb), se na hlavní obrazovce nevyskytují, aby zbytečně neodváděly pozornost uživatele od textu.



Obrázek 5.2: Hlavní obrazovka výsledného programu

5.2 Vyhodnocení dotazníků

Po vytvoření celkového programu je nutné získat odezvu od více uživatelů a zjistit jeho použitelnost. Pro získání zpětné vazby od uživatelů byly vytvořeny dva dotazníky. Uživatelům byly poskytnuty verze programu pro spuštění na operačních systémech Windows a Linux.

V prvním dotazníku byla zjišťována funkčnost zobrazení šířky formátu A4 na obrazovce uživatele, dále první dojem aplikace a uživatelovy schopnosti psaní při opisu prvního testu. Z výsledků dotazníku je patrné, že se uživatelům formát A4 zobrazoval korektně. Uživatelé vyzdvyhovali jednoduchý vzhled programu a jeho snadné ovládání. Uživatelům se také líbila možnost měnit vzhled programu pomocí implementovaných stylů, nápověda programu v pravé části a graf rychlosti opisu. Co se uživatelům na první dojem nelíbilo byla absence kurzoru. Zvláště v prvních lekcích, kdy se střídají opisovaná písmena a jsou oddělena mezerou, se uživatelům v textu špatně orientovalo. Jelikož většina uživatelů nezná zpaměti rozložení kláves na klávesnici, často uhýbali pohledem z opisovaného textu a vyhledávali klávesy, čímž ztráceli orientaci v textu. Postupem času však uživatelé uznali, že tento způsob opisu textu bez kurzoru je nutil učit se rozložení kláves a postupně se jejich orientace v textu zlepšovala. Po vyplnění prvního dotazníku byli uživatelé požádáni, aby pracovali s programem alespoň 15 minut denně po dobu jednoho týdne. Poté vyplňovali druhý dotazník.

Ve druhém dotazníku byli uživatelé dotazováni zda je program při jeho používání naváděl a zda si myslí, že je program schopen zlepšovat jejich schopnost psaní na klávesnici. Uživatelé odpovídali na tyto otázky kladně. V tomto dotazníku se také opět zjišťoval výsledek prvního testu po týdnu používání programu, z kterého je patrné, že uživatelé se v opisu zlepšovali. U otázky, zda uživatelé někdy zkoušeli jiný program pro nácvik psaní na klávesnici a zda se jim na nich líbil nějaký prvek, uživatelé uváděli, že se jim líbilo okamžité zvýrazňování již napsaného textu. Jak bylo uvedeno výše, okamžité zvýrazňování chyb může narušit uživatelovu jistotu při opisování. Proto se text ve výsledném programu zvýrazní až po ukončení opisu.

Z dotazníků bylo získáno také několik dobrých připomínek. Uživatelé si stěžovali, že při vybírání zvuků metronomu si nemohou zvuky okamžitě přehrávat a vybraný zvuk slyší až po spuštění lekce či testu se zapnutým metronomem. Proto byla do programu přidána tlačítka na okamžité přehrání zvuku. Další připomínkou byla nemožnost obarvovat výsledný text vlastními barvami. Nastavení barev bylo na základě těchto připomínek do programu také implementováno.

5.3 Testování

Výsledný program byl otestován pod operačními systémy Windows XP SP3, Windows Vista SP1, Windows 7 a Ubuntu 8.04. Při testování na operačním systému Ubuntu se nepřehrávaly zvuky metronomu. V dokumentaci Qt (zdroj [8]) se uvádí, že zvuky jsou na platformě X11 třídou `QSound` přehrávány pomocí Network Audio System. Pokud není NAS nainstalován, zvuky se nepřehrají. Činnost programu však není nijak ovlivněna. Na některých linuxový distribucích se také v Qt nedaří zjistit rozložení klávesnice. V takovýchto případech se uvažuje české rozložení. Zobrazení reálné šířky A4 formátu bylo otestováno na monitorech o rozměrech – 15.6 palců 16:10, 19 palců 4:3, 21 palců 16:10, 22 palců 16:10, 24 palců 16:9.

Kapitola 6

Závěr

Cílem práce bylo vytvořit program pro nácvik hmatové metody. Implementaci výsledného programu předcházelo nastudování zásad hmatové metody psaní na klávesnici a tvorba několika dílčích uživatelských rozhraní pro opis textu. Ve výsledném řešení je opis textu realizován bez použití kurzoru. Tento způsob opisu nepoužívá žádná z aplikací zabývající se nácvikem hmatové metody, které byly v rámci práce zkoumány.

Při použití kurzoru, který vyznačuje aktuálně opisované písmeno, je kontrola správnosti opisu jednoduchá. Po stisku klávesy se zjistí, zda stisknutý znak na klávesnici odpovídá znaku, na jehož pozici se nachází kurzor. Pokud se znaky nerovnají, došlo při opisu k chybě. Kontrola chyb ve výsledné aplikaci je složitější. Protože aplikace nepoužívá kurzor, nemůžeme porovnávat při opisu textu znak po znaku. Proto byl do aplikace implementován Levenshteinův algoritmus, který porovná opisovaný řetězec s řetězcem, který opsal uživatel, a vrátí počet chyb.

Ve výsledné aplikaci je implementován algoritmus pro zobrazení fyzického rozměru šířky formátu A4 na obrazovku, čímž dosáhneme reálnější situace opisu textu. Je počítáno s možností, že program bude spouštěn i na monitorech, na které se formát A4 fyzicky nevleze.

Jelikož nácvik hmatové metody je časově náročný proces, vliv programu na zlepšování schopnosti psaní na klávesnici musí být ještě podroben dlouhodobějším testům. Poté by mohla být provedena srovnání, který z programů nejlépe učí zásadám hmatové metody.

Možným pokračováním projektu by bylo vytvoření webových stránek, kam by se automaticky odesílaly statistiky uživatelů. Mohly by tak vznikat online žebříčky nejrychleji píšících uživatelů. Uživatelé by mohli být oceňováni různými tituly čímž by byla probuzena jejich soutěživost a motivace pracovat s programem a zlepšovat tak svou hmatovou techniku. Na těchto stránkách by také mohli uživatelé vytvářet a sdílet vlastní vzhledy a zvuky programu. Dalším možným rozšířením výsledné aplikace by mohlo být implementování her procvičující hmatovou metodu.

Při tvorbě práce autor získal zkušenosti s tvorbou uživatelských prostředí pomocí vývojového prostředí Qt Creator. Autor dále prohloubil své znalosti programovacího jazyka C++ a objektově orientovaného programování a naučil se pracovat s odbornými texty.

Literatura

[1] Firma SWX: ATF – Psaní všemi deseti prsty [online]. http://www.swx.cz/index.php?pg=1, 2002-06-30 [cit. 2009-10-11].

[2] Gilleland, M.: Levenshtein Distance [online]. http://www.merriampark.com/ld.htm, 2001-01-08 [cit. 2009-12-02].

[3] GrassSoftware: GS Typing tutor – typing tutor and typing test sofware [online]. http://www.typingstar.com/, 2003-08-14 [cit. 2009-10-15].

[4] Johansen, A. S.: Levenshtein Distance Algorithm: C++ Implementation [online]. http://www.merriampark.com/ldcpp.htm, 2002-09-30 [cit. 2009-12-17].

[5] Kleiweg, P.: Levenshtein Demo [online]. http://odur.let.rug.nl/ kleiweg/lev/, 2001-01-31 [cit. 2009-12-02].

[6] Neugebauer, T.: Desetiprstová hmatová metoda [online]. http://www.deseti-prsty.cz/hmatova_metoda.html, 2000-09-07 [cit. 2009-10-15].

[7] Neugebauer, T.: Psaní na počítači - profesionální ovládání klávesnice [online]. http://www.psani-vsemi-deseti.cz/2045/psani_na_pocitaci_profes_01.html, 2008-02 [cit. 2009-10-15].

[8] Nokia Corporation: Online Reference Documentation [online]. http://doc.qt.nokia.com/, 2010 [cit. 2010-04-27].

[9] Nádběla, J.: *Deseti prsty na klávesnici*. Computer Media s.r.o., druhé vydání, 2009, iSBN 978-80-7402-014-8.

[10] pmq SOFTWARE s.r.o.: Psaní všemi deseti [online]. http://www.deseti-prsty.cz/instalace.html, 2009-09-07 [cit. 2009-10-11].

[11] WWW Stránky: Levenshteinova vzdálenost [online]. http://www.algoritmy.net/article/1699/Levenshteinova-vzdalenost, 2010-02-06 [cit. 2010-04-27].

Dodatek A

Obsah CD

- **bin**: spustitelné verze programu
- **doc**: soubory pro vytvoření textové části bakalářské práce, výsledky dotazníků, plakát, nápověda programu
- **src**: zdrojové soubory programu