VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

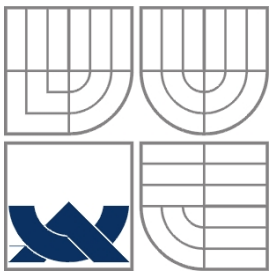
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DIKTOVACÍ SYSTÉM – UŽIVATELSKÉ ROZHRANÍ

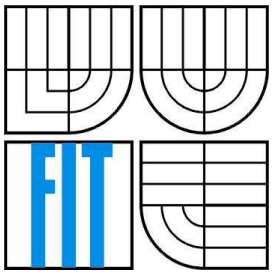
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Martin Svoboda
AUTHOR

BRNO 2009



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

DIKTOVACÍ SYSTÉM – UŽIVATELSKÉ ROZHRANÍ

DICTATION SYSTEM – USER INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Martin Svoboda
AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Schwarz
SUPERVISOR

BRNO 2009

Abstrakt

Pod pojmem diktovací systém chápeme software, který se skládá ze dvou hlavních částí. První částí je rozpoznávač pro rozpoznávání mluveného slova, druhou částí je uživatelské rozhraní pro interakci s uživatelem a zpracování výstupu rozpoznávače.
Tato práce se zaměřuje na uživatelské rozhraní diktovacího systému, popis komunikace mezi rozpoznávačem a uživatelským rozhraním, opravou slov přicházejících od rozpoznávače a převodem čísel ze slovní podoby na číselnou.

Abstract

The term dictation system we understand the software system, which consists of two main parts. The first part is recognizer spoken word, the second is the user interface for user interaction and processing output recognizer.
This work focuses on the user interface dictation system, a description of comunication between the recognizer and the user interface, the correction of words coming from the recognizer and transfer numbers from verbal to numerical form.

Klíčová slova

Diktovací systém, rozpoznávač, grafické uživatelské rozhraní, GUI, pravidla pro převody čísel, opravy slov.

Keywords

Dictation system, recognizer, graphical user interface, GUI, rules for transfers of numbers, repairs words.

Citace

Svoboda Martin: Diktovací systém – uživatelské rozhraní, bakalářská práce, Brno, FIT VUT v Brně, 2009

Diktovací systém – uživatelské rozhraní

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Schwarze. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Svoboda
10.05.2009

Poděkování

Rád bych poděkoval Ing. Petru Schwarzovi, který mi ochotně poskytl potřebné informace a materiály, za jeho rady a pomoc při řešení této bakalářské práce.

Obsah

1 Úvod

V současné době se s rychlostí vývoje nejnovějších počítačových technologií, tedy hardware i software, stává nejpomalejším článkem v používání počítače komunikace uživatele s počítačem.

Dnes se nejčastěji používá klávesnice s myší (případně jiné polohovací zařízení). Tato komunikace však není příliš efektivní, rychlost psaní na klávesnici je do značné míry omezená a je jen velmi malé procento uživatelů, kteří dokážou psát stejnou rychlostí jako diktovat. Navíc při psaní zaměstnávají obě ruce a zrakem kontrolují psaný text. Uživatel tedy musí být plně soustředěný na to, co píše. Z toho důvodu se věnuje velká pozornost rozpoznávání řeči, kdy počítač rozumí tomu, co uživatel říká a provádí požadovanou činnost. Ten již pak není natolik vytížen – nemusí sledovat obrazovku a má volné ruce. Může dělat více činností zároveň, čímž se znatelně zvýší efektivita jeho práce.

V této práci se zabývám grafickým uživatelským rozhraním takovéhoto rozpoznávače řeči vyvinutého na Fakultě informačních technologií VUT v Brně, vzájemnou komunikací rozpoznávače s jeho grafickým uživatelským rozhraním, úpravou a zpracováním výstupu rozpoznávače.

1.1 Cíl práce

Cíl této práce a můj úkol je ukázání směru, kterým se v nejbližší budoucnosti bude ubírat oblast komunikace člověka s počítačem. Nastudování a nastínění komunikace rozpoznávače řeči s okolím, návrh a implementace jeho grafického uživatelského rozhraní. V rámci tohoto rozhraní je také implementováno generování čísel ze slov přicházejících od rozpoznávače a provádění oprav slov spočívající v záměně slova za jiné, uživatelem definované, slovo. Dále otestování mnou navrženého a implementovaného řešení a zamyšlení se nad možnostmi využití rozpoznávače.

Výslednou aplikací bude kompletní diktovací systém, který bude zobrazovat text diktovaný uživatelem, umožní jeho formátování a editaci včetně uložení do souboru.

1.2 Struktura práce

Ve druhé kapitole jsou představeny diktovací systémy na dnešním trhu. Třetí kapitola se zabývá volbou operačního systému, pod nímž bude provedena implementace aplikace, programovacího jazyka a nástrojů potřebných k implementaci. Ve čtvrté kapitole je popsán rozpoznávač vyvinutý na Fakultě informačních technologií VUT v Brně a jeho komunikace s grafickým uživatelským rozhraním včetně komplikací s ní spojených. Pátá kapitola se zabývá zpracováním výstupu rozpoznávače. Návrh grafického uživatelského rozhraní zpracovává šestá kapitola a implementaci podle tohoto návrhu rozebírá kapitola sedmá. V osmé kapitole je popsáno ověření funkčnosti a celá práce je ukončena devátou kapitolou zabývající se zhodnocením této práce.

2 Diktovací systémy na trhu

Diktovacím systémům a ovládání počítačů hlasem se dnes věnuje mnoho firem, větších i menších. Největší přínos však mají dvě z nich. Jedná se o firmy Nuance se svým produktem Dragon NarurallySpeaking a IBM s technologií ViaVoice, na které se podíváme blíže.

2.1 Nuance – Dragon NaturallySpeaking

Nuance se soustřeďuje svým produktem Dragon NaturallySpeaking 10 právě na diktování a diktovací systémy. Tento produkt je nabízen se zaměřením na mnoho odvětví a v mnoha jazykových mutacích. Měl jsem možnost vyzkoušet si práci s jeho anglickou verzí pro domácí použití.

Obrázek 2.1: Dragon NaturallySpeaking 10 Preferred (převzato z [1])

2.1.1 Seznámení

Při prvním spuštění vyzve program uživatele k založení uživatelského účtu. Po vytvoření uživatele je pak potřeba vybrat zdroj diktování (výběr mikrofonu) a vybrat slovník, který chcete používat. Uživatel si může zvolit, zda chce používat slovník pro příkazy nebo všeobecný slovník.

Aplikace poté nabídne vstupní trénink, jehož součástí je také kalibrace mikrofonu. Tento trénink seznámí uživatele s celým programem, jeho ovládáním a také nastavením.

Vlastní kalibrace mikrofonu je tvořena průvodcem kalibrací. Ihned na začátku je zobrazeno upozornění, že by měl být mikrofon během kalibrace umístěn tam, kde bude během používání aplikace. V závěru kalibrace se provede také úprava nastavení směšovače a provede se test kvality mikrofonu.

2.1.2 Zkušenosti a zhodnocení

Aplikace má pěkné a příjemné grafické uživatelské rozhraní, na které si uživatel nemusí dlouho zvykat.

Bohužel již při kalibraci mikrofonu aplikace sdělila, že kvalita použitého mikrofonu je velice špatná a byla doporučena jeho výměna. Toto bylo způsobeno hlavně mou anglickou výslovností, kdy jsem se přesvědčil, že tyto diktovací systémy jsou na výslovnost velice citlivé. Většinou rozumí jen rodilému mluvčímu, což není vlastnost jen tohoto produktu, ale diktovacích systémů všeobecně.

2.2 IBM - ViaVoice

Firmu IBM a její rozsah působnosti na poli informačních technologií není třeba představovat, proto se podíváme na její vestavěnou technologii ViaVoice, která spadá právě do oblasti diktovacích systémů.

2.2.1 Vlastnosti technologie ViaVoice

Tato podkapitola byla převzata z [2]. Tuto technologie jsem neměl možnost osobně vyzkoušet a byl jsem odkázán na zdroje z internetu.

Technologie poskytuje plně integrované, automatické rozpoznávání řeči a schopnost čtení textů. Zahrnuje podporu netradičních příkazů pro přirozené a intuitivní fráze, které nemusí být známy v okamžiku diktování. Rozpoznávací slovník obsahuje přes 200 000 slov.

Její užití spadá do oblasti malých mobilních zařízení a automobilových telematických systémů. Vývojářům pomáhá snižovat čas a znalosti potřebné k vývoji pokročilých hlasových aplikací pro zařízení a vzdálené systémy.

3 Volba nástrojů pro implementaci

Hlavním požadavkem na výslednou aplikaci je dostupnost pro co největší počet uživatelů. Tento požadavek byl hlavním a vlastně také jediným, který rozhodl o implementaci diktovacího systému pod operačním systémem Windows XP. Jedná se v současné době o nejpoužívanější operační systém.

V této kapitole také vybereme programovací jazyk a nástroje pro implementaci. Návrh bude poté proveden s ohledem na vybraný operační systém a programovací jazyk.

3.1 WinApi

WinApi je aplikační rozhraní pro programování aplikací pod operačním systémem Windows. Budeme – li uvažovat aplikace s grafickým uživatelským rozhraním, jsou ve Windows ovládány pomocí událostí. Programování takovýchto aplikací se nazývá událostmi řízené programování. Událostí je velké množství a můžou být vyvolány různými periferními zařízeními (myš, klávesnice, atd.), ale také programy samotnými.

Předpokládá se, že v případě implementace pod operačním systémem Windows XP bude výsledná aplikace přenositelná a spustitelná také na operačních systémech Windows Vista a Windows 7.

3.2 Volba programovacího jazyka

V dnešní době existuje spousta programovacích jazyků vyšší, či nižší úrovně. Každý jazyk má své výhody, ale také nevýhody. Některé z těchto jazyků jsou obecně použitelné pro implementaci téměř jakékoli aplikace, ostatní jsou zaměřeny spíše na specifické aplikace, pro které byly navrženy.

Naprogramování grafického uživatelského rozhraní diktovacího systému přichází v úvahu použitím jazyka C++, C#, Java nebo Python. Po zvážení všech požadavků na výslednou aplikaci a také ze zkušeností s programováním v jazyce C++ jsem se rozhodl právě pro implementaci v tomto jazyce.

3.3 Volba toolkitu

Po zamyšlení se nad návrhem grafického uživatelského rozhraní diktovacího systému jsem si stanovil čtyři požadavky, které musí splnit toolkit, v němž nakonec bude systém implementován:

- Multiplatformní toolkit
- Snadná implementace grafického uživatelského rozhraní
- Snadná synchronizace a ovládání vláken
- Kvalitní dokumentace a tutoriály

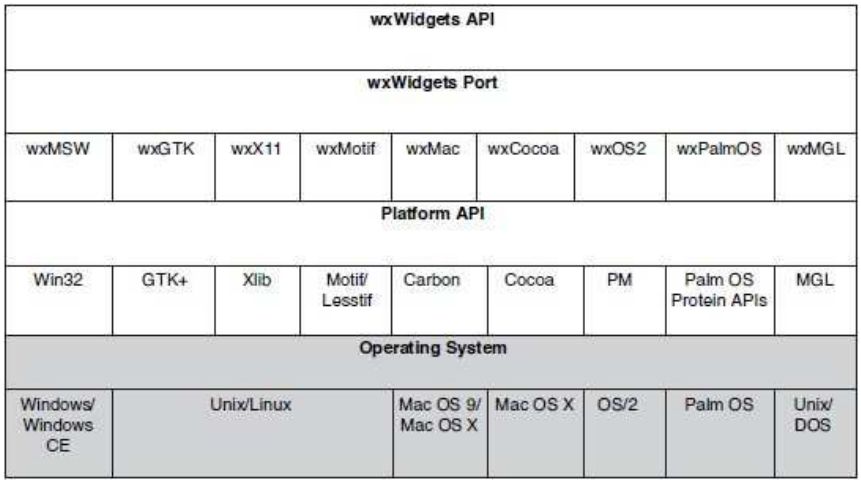
Těmto požadavkům vyhovuje spousta toolkitů jako je GTK+, QT, wxWidgets a mnoho dalších.

3.4 WxWidgets

Já jsem se nakonec rozhodl zvolit toolkit wxWidgets, protože odpovídá všem požadavkům uvedených v kapitole 3.3. Velkou roli při výběru tohoto toolkitu hrálo také to, že již s ním mám zkušenosti z dřívějšího programování grafických uživatelských rozhraní aplikací.

3.4.1 Popis wxWidgets

Jedná se o multiplatformní nástroj pro implementaci grafických uživatelských rozhraní jak pro desktopové aplikace, tak také pro mobilní zařízení [3]. Aplikace jsou snadno přenositelné mezi operačními systémy a vyžadují jen minimální nebo žádný zásah do kódu při přenosu. Podporuje implementaci pod spoustou programovacích jazyků (C++, Python, Java, C#, JavaScript, Perl aj.). Na obrázku 3.1 jsou zobrazeny porty wxWidgets.



Obrázek 3.1: Porty wxWidgets (převzato z [3])

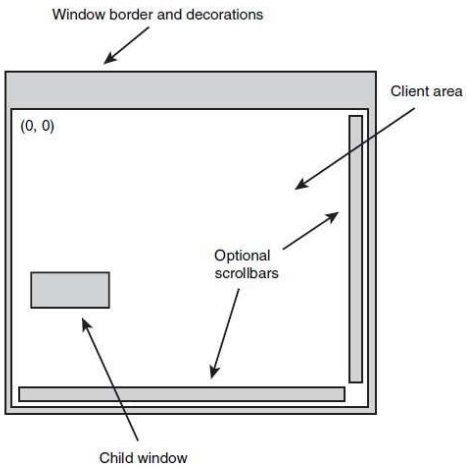
Tento framework však neobsahuje pouze třídy pro práci s grafickým uživatelským rozhraním, ale také mnoho dalších tříd pro práci se soubory, proudy, vlákny atd. [3].

Zakladatelem tohoto frameworku je Julian Smart, který se na jeho vývoji podílí dodnes. Původně se tento framework jmenoval wxWindow, ale na nátlak firmy Microsoft, byl přejmenován na dnešní název wxWidgets.

3.4.2 Programy ve wxWidgets

Aplikace implementované ve frameworku wxWidgets obsahují třídu, která dědí `wxApp`. V této třídě je vytvořena metoda `OnInit`, v níž je vytvořeno hlavní okno aplikace. Po vracení hodnoty `true` touto metodou, spustí wxWidgets smyčku událostí.

V hlavním okně aplikace jsou pak obsaženy další okna a ovládací prvky aplikace. Rozvržení prvků v hlavním okně je na obrázku 3.2.

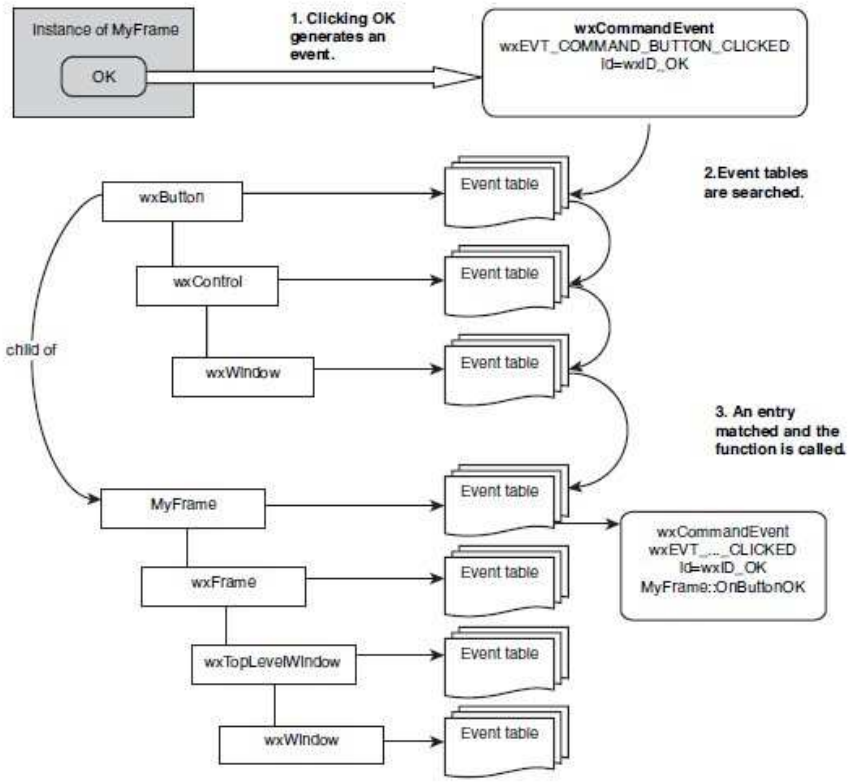


Obrázek 3.2: Rozvržení prvků hlavního okna (převzato z [3])

3.4.3 Události ve wxWidgets

Aplikace ve wxWidgets jsou řízené událostmi a proto je nutná také implementace obsluhy těchto událostí. V první řadě musíme ve třídě, v níž chceme provést obsluhu události deklarovat tabulku událostí. Tuto tabulku mohou deklarovat všechny třídy dědící třídu wxEvtHandler. V tabulce událostí se pak nastaví metoda třídy, kterou má být událost obsloužena.

Kdykoli je vyvolána nějaká událost, vytvoří se objekt této události, který je odeslán aplikaci. Ta řadí tyto události do fronty událostí a odtud je v pořadí, ve kterém přišly, zpracovává. Na obrázku 3.3 je znázorněna obsluha stisku tlačítka.



Obrázek 3.3: Obsluha stisku tlačítka (převzato z [3])

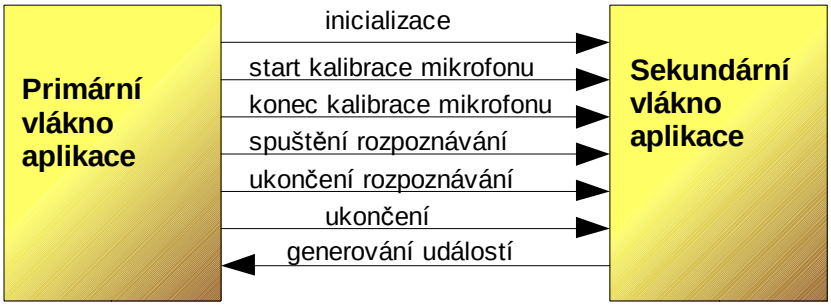
3.5 Výběr editoru a překladače

Jak uvádí kapitola 3.2, pro implementaci byl zvolen jazyk C++. Existuje spousta vývojových prostředí a překladačů pro tento jazyk. Já jsem zvolil na základě svých předešlých kladných zkušeností překladač MinGW32, který generuje kvalitní spustitelný kód. Součástí MinGW32 je také program make, který zjednodušuje kompilaci. Stačí vytvořit soubor Makefile s pravidly pro překlad a umístit jej do adresáře se zdrojovými kódy. Kompilace se potom spustí pomocí příkazu make.

V dalším kroku jsem volil editor pro psaní kódu. Pod operačním systémem Windows je mým oblíbeným editorem PSPad. Tento editor mi vyhovuje svou širokou nastavitelností a zvýrazňováním zdrojového kódu.

4 Diktovací systém

Celý diktovací systém se skládá ze dvou hlavních částí, které mezi sebou komunikují. Jedná se o rozpoznávač řeči a jeho grafické uživatelské rozhraní. Každá z těchto částí systému běží v samostatném vlákně, přičemž primárním vláknem celé aplikace je vlákno grafického uživatelského rozhraní. Pro ilustraci slouží obrázek 4.1.



Obrázek 4.1: Vlákna diktovacího systému

4.1 Rozpoznávač řeči

Vlastní rozpoznávač řeči je DLL knihovna obsahující soubor tříd pro rozpoznávání řeči vyvinutá na Fakultě informačních technologií VUT v Brně.

Rozpoznávač obsahuje statickou rozpoznávací síť a slovník. Slova jsou generována v podobě fonémů. V poslední fázi je kontrolováno, zda mohou tato slova následovat za sebou v pořadí, ve kterém byly rozpoznány, a vytváří se síť hypotéz. V této síti dochází k přezkórování a ořezání sítě o slova s nejnižším ohodnocením. Naopak slovo s nejvyšším ohodnocením je odesláno na výstup rozpoznávače.

4.2 Komunikace s uživatelským rozhraním

Jak již bylo zmíněno v kapitole 4.1, rozpoznávač řeči je DLL knihovna obsahující soubor tříd pro rozpoznávání řeči. Pro vytvoření spustitelného programu je proto nutné vytvořit vlastní zdrojový kód obsahující třídu, jenž dědí z této knihovny dvě třídy – `STransCallbackI` a `SErrorCallbackI`.

Třída `STransCallbackI` implementuje metodu `OnTransCription`, kterou musíme ve svém kódu přepsat pro zpracování výstupu rozpoznávače dle naší potřeby. Tato metoda je za běhu programu volána rozpoznávačem běžícím v sekundárním vlákně aplikace a je jí předáno jedno rozpoznané slovo.

Třída `SErrorCallbackI` implementuje metodu `OnTransMessage`, kterou musíme taktéž ve svém kódu přepsat pro zpracování předávaných informací o stavu rozpoznávače. Této metodě se

za běhu programu předávají logy, varovná hlášení a hlášení o chybách, které se vyskytly během práce rozpoznávače běžícího v sekundárním vlákně aplikace.

4.2.1 Komplikace se synchronizací vláken ve wxWidgets

I přes jednoduchou synchronizaci vláken ve wxWidgets pomocí volání metod `wxMutexGuiEnter`, volané při vstupu sekundárního vlákna do primárního, a `wxMutexGuiLeave`, volané při opuštění primárního vlákna, se vyskytly komplikace se synchronizací vlákna rozpoznávače s primárním vláknem grafického uživatelského rozhraní. Program zatuhl hned při spuštění a přitom alokoval paměť systému, dokud nebyla zcela vyčerpána.

4.2.2 Vyřešení synchronizace vláken ve wxWidgets

Problém se synchronizací vláken byl nakonec vyřešen zcela jiným způsobem a volání metod `wxMutexGuiEnter` a `wxMutexGuiLeave` bylo vynecháno. Sekundární vlákno aplikace nyní vůbec nepřistupuje ke grafickým prvkům grafického uživatelského rozhraní.

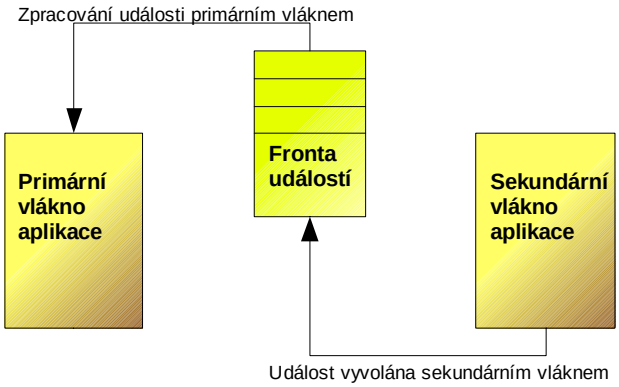
Metody `OnTransCription` a `OnTransMessage` převezmou data od rozpoznávače a uloží je do datové struktury. Pro metodu `OnTransCription` byla vytvořena datová struktura `MyMSWord` a pro metodu `OnTransMessage` datová struktura `MyMSMessage`. Tyto struktury vypadají následovně:

```
typedef struct
{
        message_type type;
        wxString message;
} MyMSMessage;

typedef struct
{
        wxString word;
} MyMSWord;
```

Ve výše zmíněných metodách se alokuje paměť pro danou strukturu a vytvoří se událost, do které se uloží ukazatel na tuto strukturu. Událost se odešle primárnímu vláknu aplikace, kde se zařadí do fronty událostí, která je postupně tímto vláknem zpracovávaná.

Na tyto události reagují funkce, které zpracují data uložená ve struktuře a poté uvolní paměť alokovanou pro danou strukturu. Na obrázku 4.2 je zobrazen princip synchronizace vláken.



Obrázek 4.2: Princip synchronizace vláken

4.3 Kalibrace mikrofonu

Pro správnou funkci rozpoznávače je nutné provést kalibraci mikrofonu. V opačném případě může docházet ke zkreslením a výstup rozpoznávače nebude odpovídat skutečnosti. Ke kalibraci mikrofonu obsahuje rozpoznávač metody `StartMikeCalibration` pro spuštění kalibrace a `FinishMikeCalibration` pro ukončení kalibrace.

Kalibraci mikrofonu stačí provést pouze jednou. Rozpoznávač si informace o kalibraci uloží a při příštím spuštění načte. Jestliže bude s aplikací pracovat více uživatelů, je pak nutné provést kalibrací také při každé změně uživatele.

Pro usnadnění ovládání diktovacího systému bude kalibrace mikrofonu implementována v uživatelském rozhraní jako průvodce kalibrací mikrofonu. Více se tomuto problému věnují kapitoly 6.3 a 7.2.

4.4 Možná rozšíření rozpoznávače do budoucna

V budoucnu by mohl rozpoznávač podporovat práci pro více uživatelů, aby se nemusela při každé změně uživatele provádět kalibrace mikrofonu. Rozpoznávač by podporoval správu uživatelů.

To by obnášelo implementaci funkce pro vytváření, editaci a mazání uživatelských účtů. Každý vytvořený uživatel by měl vlastní složku, do níž by rozpoznávač ukládal uživatelova nastavení, hlavně kalibraci mikrofonu.

V další řadě by byla nutná implementace funkcí pro získání seznamu uživatelů, označení aktivního uživatele a změnu aktivního uživatele.

5 Zpracování výstupu rozpoznávače

Požadavkem na diktovací systém byla schopnost upravit výstup rozpoznávače, a to následovně:

- Umožnit nahrazování slov
- Převod diakritiky z doslovného přepisu na symboly
- Obsahovat základní pravidla pro převody čísel

5.1 Nahrazování slov

Tato kapitola se zabývá hned prvníma dvěma požadavky na zpracování výstupu rozpoznávače. Princip, na němž je založen jejich návrh a implementace, je totožný.

Přijde slovo od rozpoznávače, systém zjistí, zda se toto slovo nachází v databázi nahrazovaných slov. Jestliže ano, vezme slovo a nahradí jej slovem, které uživatel definoval jako nahrazující slovo. Systém nerozlišuje, zda se jedná o slova diakritiky (čárka, tečka, otazník, atd.) jakékoli jiné znaménko nebo slovo, které chceme nahradit.

Více se nahrazování slov věnují kapitoly 6.4 a 7.3, neboť nahrazování slov je implementováno jako součást grafického uživatelského rozhraní.

5.2 Generování čísel

Posledním z požadavků na zpracování výstupu rozpoznávače bylo generování čísel z doslovného přepisu na číslice a čísla. Jedná se o schopnost generování čísel z přicházejících slov. Tímto požadavkem se zabývá tato kapitola.

5.2.1 Analýza problému

Pro vytvoření tohoto generátoru čísel jsem zvažoval návrh a implementací dvěma možnými přístupy:

- Pomocí gramatik
- Pomocí konečného automatu

Vzhledem k tomu, že pro generování čísel byl zvolen rozsah 0 – 999 999, bylo by vytvoření gramatiky mnohem náročnější než návrh konečného automatu a rychlost výsledného generátoru by nebyla nijak ovlivněna, byl nakonec zvolen návrh a implementace pomocí konečného automatu.

5.2.2 Popis konečného automatu

Navržený konečný automat má celkem 14 stavů. Počáteční stav „s“ je zároveň také jediným koncovým stavem. To znamená, že při ukončení generování čísla přejde automat do počátečního stavu a je připraven generovat další číslo. Konečný automat je v tabulkové formě zobrazen v tabulkách 5.1, 5.2 a 5.3.

stav / vstup	nula	jedna	jeden	dvě	dva	tři	čtyři	pět	šest	sedm	osm	devět	deset	jedenáct	dvanáct
s	s / 0	s / 1	p1 / 1	p2 / 2	p3 / 2	p4 / 3	p4 / 4	p5 / 5	p5 / 6	p5 / 7	p5 / 8	p5 / 9	p1 / 10	p5 / 11	p5 / 12
p1	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p2	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p3	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p4	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p5	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p6	s / word	p1 / 1	s / word	s / word	p1 / 2	p1 / 3	p1 / 4	p1 / 5	p1 / 6	p1 / 7	p1 / 8	p1 / 9	s / word	s / word	s / word
p7	s / word	p1 / 01	s / word	s / word	p1 / 02	p1 / 03	p1 / 04	p1 / 05	p1 / 06	p1 / 07	p1 / 08	p1 / 09	p1 / 10	p1 / 11	p1 / 12
p8	s / 000	s / 001	s / 000	p9 / x	s / 002	p10 / x	p10 / x	p11 / x	p11 / x	p11 / x	p11 / x	p11 / x	s / 010	s / 011	s / 012
p9	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p10	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p11	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p12	s / 0	s / 1	s / 0	s / 0	s / 2	s / 3	s / 4	s / 5	s / 6	s / 7	s / 8	s / 9	s / 0	s / 0	s / 0
p13	s / 00	s / 01	s / 00	s / 00	s / 02	s / 03	s / 04	s / 05	s / 06	s / 07	s / 08	s / 09	s / 10	s / 11	s / 12

Tabulka 5.1: Konečný automat generátoru čísel 1/3

stav / vstup	třináct	čtrnáct	patnáct	šestnáct	sedmnáct	osmnáct	devatenáct	dvacet	třicet	čtyřicet	padesát
s	p5 / 13	p5 / 14	p5 / 15	p5 / 16	p5 / 17	p5 / 18	p5 / 19	p6 / 2	p6 / 3	p6 / 4	p6 / 5
p1	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p2	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p3	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p4	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p5	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p6	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word
p7	p1 / 13	p1 / 14	p1 / 15	p1 / 16	p1 / 17	p1 / 18	p1 / 19	p6 / 2	p6 / 3	p6 / 4	p6 / 5
p8	s / 013	s / 014	s / 015	s / 016	s / 017	s / 018	s / 019	p12 / 02	p12 / 03	p12 / 04	p12 / 05
p9	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p10	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p11	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p12	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0
p13	s / 13	s / 14	s / 15	s / 16	s / 17	s / 18	s / 19	p12 / 2	p12 / 3	p12 / 4	p12 / 5

Tabulka 5.2: Konečný automat generátoru čísel 2/3

stav / vstup	šedesát	sedmdesát	osmdesát	devadesát	sto	sta	stě	set	tisíc	tisíce	jiné
s	p6 / 6	p6 / 7	p6 / 8	p6 / 9	p7 / 1	s / word	s / word	s / word	p8 / 1	s / word	s / word
p1	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	p8 / " "	s / word	s / word
p2	s / word	s / word	s / word	s / word	s / word	s / word	p7 / ""	s / word	s / word	s / word	s / word
p3	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	p8 / " "	s / word
p4	s / word	s / word	s / word	s / word	s / word	p7 / ""	s / word	s / word	s / word	p8 / " "	s / word
p5	s / word	s / word	s / word	s / word	s / word	s / word	s / word	p7 / ""	p8 / " "	s / word	s / word
p6	s / word	s / word	s / word	s / word	s / word	s / word	s / word	s / word	p8 / " "	s / word	s / word
p7	p6 / 6	p6 / 7	p6 / 8	p6 / 9	s / word	s / word	s / word	s / word	p8 / 00	s / word	s / word
p8	p12 / 06	p12 / 07	p12 / 08	p12 / 09	p13 / 1	s / 000	s / 000	s / 000	s / 000	s / 000	s / 000
p9	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	p13 / i	s / 00i	s / 00i	s / 00i	s / 00i
p10	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	p13 / i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i
p11	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	s / 00i	p13 / i	s / 00i	s / 00i	s / 00i
p12	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0	s / 0
p13	p12 / 6	p12 / 7	p12 / 8	p12 / 9	s / 00	s / 00	s / 00	s / 00	s / 00	s / 00	s / 00

Tabulka 5.3: Konečný automat generátoru čísel 3/3

Sloupce těchto tabulek obsahují vstupní abecedu konečného automatu a řádky aktuální stav. Jednotlivé buňky jsou ve tvaru: <*příští stav*> / <*výstup automatu*>. Příštím stavem je myšlen stav, ve kterém bude konečný automat při příchodu dalšího slova od rozpoznávače. Výstup automatu může nabýt těchto hodnot:

- ***word*** – výstupem generátoru bude přicházející slovo, nedojde ke změně
- ***x*** – pozice získané číslice není známa a bude potřeba tuto číslici uložit a umístit ji na svou pozici až bude její pozice známa
- ***i*** – pozice číslice uchované v ***x*** je již známa, výstupem je tato číslice
- **" "** – výstupem je mezera, která odděluje tisíce od stovek (např. 999 999)
- **""** – výstupem je prázdný string, protože příchozí slovo je součástí čísla, ale nejedná se o žádnou číslici ani nevkládá mezeru (set, stě, sta)
- **<*číslo*>** – výstupem je číslo nebo číslice
- **<*číslo*>*i*** – výstupem je spojení čísla **<*číslo*>** a číslice uložené v ***x***

5.2.3 Implementace generátoru čísel

Implementace generátoru čísel vychází z tabulek 5.1, 5.2 a 5.3. Celý tento konečný automat je realizován třídou `GenNumbers`. Tato třída má pouze jednu veřejnou metodu `Generate`, která přebírá vždy jedno slovo. V závislosti na současném stavu a na předaném slovu se generuje na výstup posloupnost číslic.

Každému stavu konečného automatu odpovídá jedna privátní metoda, která implementuje chování přiřazeného stavu. Automat si uchovává informaci o stavu, ve kterém bude při příštím volání metody `Generate`.

U některých posloupností slov není rozhodnutelné, na kterou pozici se má číslovka umístit (statisíce, desetitisíce, tisíce, stovky, desítky, jednotky). V takovém případě si konečný automat získanou číslici uloží a umístí jí na svou pozici až tehdy, kdy bude její pozice známa.

5.2.4 Možná rozšíření generátoru čísel

V budoucnu by se mohlo generování čísel rozšířit také do záporných čísel a desetinných čísel. Mohlo by se implementovat generování zlomků a zápis čísel pomocí mocniny čísla 10.

V takovém případě by se již oplatilo navrhnout gramatiku, protože konečný automat by měl příliš velké množství stavů, byl by nepřehledný a snadno by mohlo dojít k chybám jak při návrhu tak implementaci takto složitého konečného automatu.

5.3 Jiné možnosti zpracování výstupu

Zpracování výstupu rozpoznávače nabízí opravdu velkou škálu možností. Ve většině případů se zatím jedná jen o vyhlídky do budoucnosti, ale byla by škoda se nepozastavit alespoň nad dvěma:

- Ovládání operačního systému a programů
- Ovládání strojů a přístrojů

5.3.1 Ovládání operačního systému a programů

Dnes se k ovládání operačního systému a většiny programů používá myš, případně jiné polohovací zařízení. Představme si však, že bychom celý operační systém mohli ovládat hlasem. Přepínání mezi programy, vyplňování textů atd. Kolik práce a času by nám takovéto ovládání hlasem ušetřilo.

Tato myšlenka je pěkná, ale má svá omezení. Úspěšnost rozpoznávání by se musela přiblížit 100%, nároky na hardware by byly mnohem větší, protože rozpoznávač spotřebovává velké množství zdrojů operačního systému. V neposlední řadě nesmíme zapomenout na práci v kancelářích, kde vedle sebe sedí spousta kolegů. Místo tichého klikání myší by se navzájem rušili hlasovým ovládáním.

Největším problémem ovládání jiných programů jsou však programy samy o sobě. Každý program je napsán jiným stylem, v jiném jazyce a je určen k jinému použití. K ovládání by bylo potřeba znát vnitřní strukturu těchto programů, nebo alespoň jejich grafického rozhraní. To není zcela reálné, protože ne všechny programy jsou open-source a firmy vyvíjející komerční software své zdrojové kódy jen tak neuvolní.

Řešením by mohl být mezičlánek mezi rozpoznávačem a programem, kde by si jednotlivé aplikace registrovaly události, které jim mají být zaslány, když uživatel vysloví příkaz určený právě této aplikaci.

5.3.2 Ovládání strojů a přístrojů

Ovládání strojů a přístrojů hlasem je v dnešní době velice aktuální myšlenkou. Vzpomeňme například na hlasové ovládání mobilních telefonů. Této oblasti se věnuje spousta odborníků, kteří se tuto technologii snaží zdokonalit.

Kdo ví, třeba se v blízké budoucnosti bude většina přístrojů v domácnosti (světla, televize, DVD atd.) ovládat hlasem.

6 Návrh uživatelského rozhraní

Hlavním cílem této práce je návrh a implementace grafického uživatelského rozhraní diktovacího systému. Návrhem tohoto rozhraní se zabývá tato kapitola.

Výsledné uživatelské rozhraní má být jednoduché a pro uživatele snadno pochopitelné. Celá aplikace má pak sloužit k prezentování rozpoznávače širší veřejnosti.

6.1 Uživatelský vstup

Samozřejmostí je, že aplikace musí být schopná reagovat na vstup uživatele. Protože se jedná o diktovací systém, program bude obsluhovat:

- Vstup z klávesnice
- Ovládání myší
- Hlasový vstup – diktování

Celý systém bude ovládán myší, stejně jako u většiny jiných aplikací. Očekává se také implementace klávesových zkratek pro urychlení práce s programem.

Vstup z klávesnice bude sloužit k opravám diktovaného textu, klasickému psaní dokumentů a k vyplňování formulářů (dialogů). Samozřejmostí bude posun kurzoru na řádku i po řádcích.

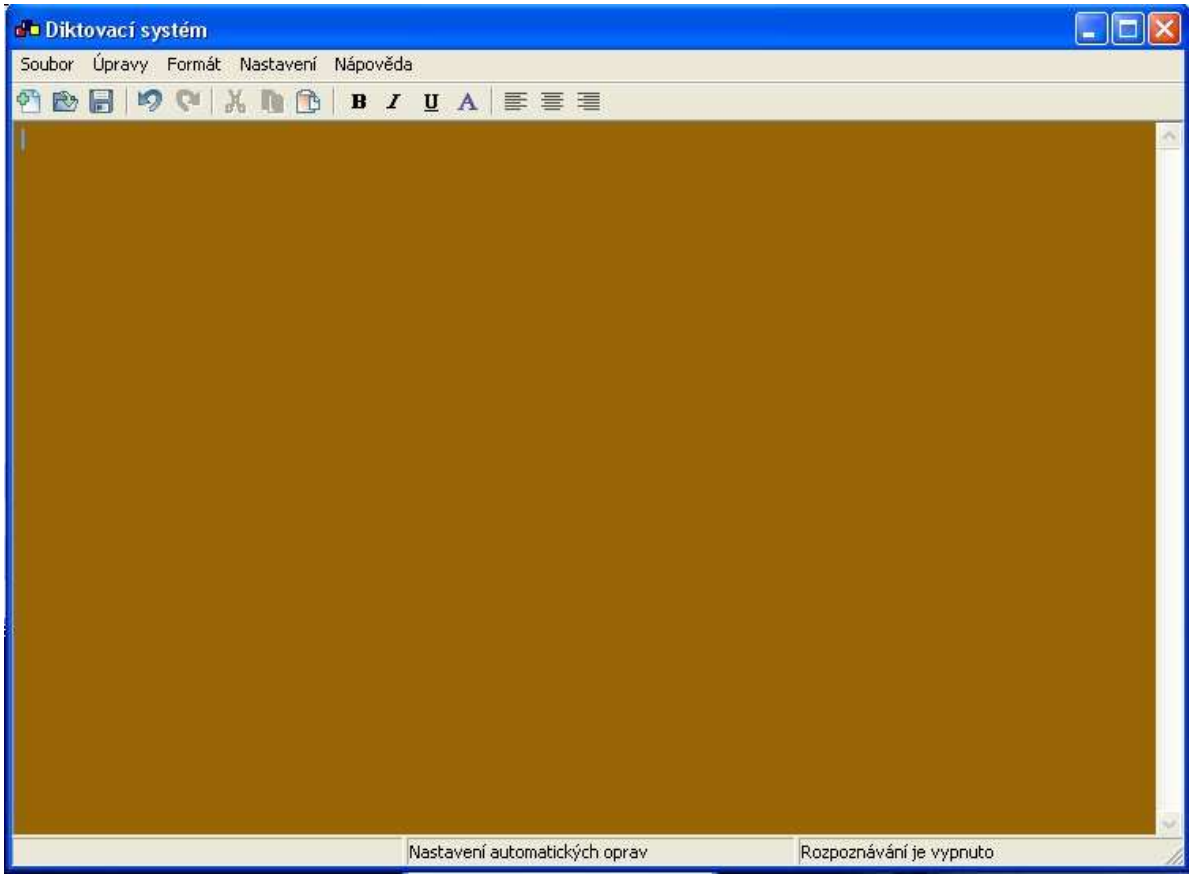
Hlasový vstup bude použit pouze k diktování.

6.2 Hlavní okno aplikace

Hlavní okno aplikace je okno, které se uživateli zobrazí při spuštění programu a v něm budou obsaženy všechny ostatní prvky grafického uživatelského rozhraní. Proto musí splňovat určité požadavky:

- Obsahuje jednoduché a přehledné menu
- Obsahuje toolbar – nástrojovou lištu
- Obsahuje statusbar – stavový řádek
- Je jednoduché a přehledné

Samozřejmostí jsou standardní ovládací prvky operačního systému Windows – rám okna obsahující tyto prvky. Na obrázku 6.1 je zobrazeno hlavní okno aplikace.



Obrázek 6.1: Hlavní okno aplikace

6.2.1 Menu

Menu zaujme pozici standardně v levém horním rohu aplikace. Jeho hlavní nabídka sestává z pěti položek – „Soubor“, „Úpravy“, „Formát“, „Diktování“ a „Nápověda“. Na obrázku 6.2 lze toto menu vidět s otevřenou záložkou „Soubor“.

Obrázek 6.2: Hlavní nabídka menu aplikace

V záložce „Soubor“ budou umístěny ovládací prvky pro vytvoření nového dokumentu, otevření dokumentu, uložení dokumentu a ukončení aplikace. Samozřejmostí je zobrazení klávesových zkratek pro jednotlivé prvky.

Menu „Úpravy“ bude obsahovat prvky pro práci s textem. Jedná se o krok zpět, krok vpřed, vyjmutí textu, kopírování textu a vložení textu.

Pod záložkou „Formát“ se bude nacházet nastavení písma a nastavení pozadí uživatelské plochy.

V menu „Diktování“ pak budou prvky pro ovládání rozpoznávače a nastavení úprav jeho výstupu. Své místo zde zaujme zapnutí a vypnutí diktování, kalibrace mikrofonu, povolení úprav výstupu rozpoznávače, jejich nastavení a také nastavení písma, kterým se bude zobrazovat diktovaný text.

V poslední záložce „Nápověda“ bude umístěn stručný popis programu a nápověda.

6.2.2 Toolbar – nástrojová lišta

Nástrojová lišta bude umístěna v hlavním okně aplikace ihned pod menu. V této liště jsou zpřístupněny nejdůležitější ovládací prvky aplikace a formátování textu. Tuto nástrojovou lištu můžete vidět na obrázku 6.3.

Obrázek 6.3: Toolbar hlavního okna aplikace

6.2.3 Statusbar – stavový řádek

V dolní části hlavního okna aplikace bude umístěn statusbar rozdělený do tří částí. Na obrázku 6.4 je tento statusbar zobrazen.

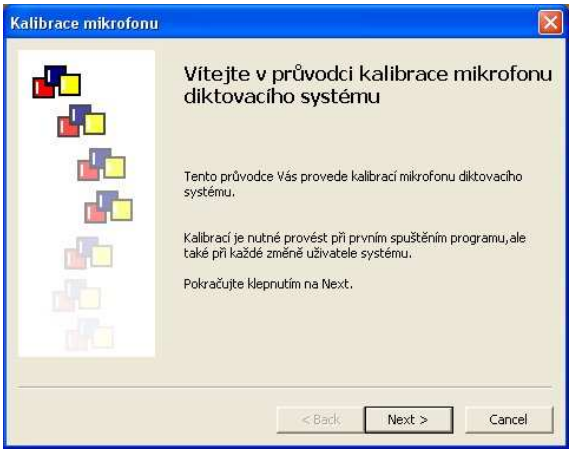
Obrázek 6.4: Statusbar hlavního okna aplikace

V levé části statusbaru se bude zobrazovat popisek aktuální položky v menu. V prostřední části je umístěna informace o provedené akci uživatele a její úspěšnosti. V pravé části uživatel uvidí informaci, zda je rozpoznávání spuštěno, či nikoli.

6.3 Průvodce kalibrací mikrofonu

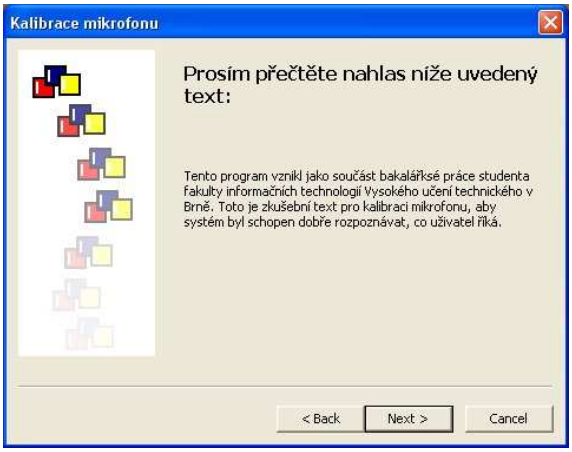
Kalibrace mikrofonu bude implementována jako uživatelský průvodce kalibrací mikrofonu. Tento přístup byl zvolen s ohledem na co největší usnadnění práce s aplikací. Uživatel bude proveden kalibrací krok za krokem a všechny důležité informace mu budou sděleny průvodcem.

Tento průvodce se bude skládat ze tří oken. V prvním okně bude zobrazena uvítací zpráva s informací, že se jedná o průvodce kalibrací mikrofonu a také informace o tom, kdy je potřeba kalibraci provést. Toto okno je na obrázku 6.5.



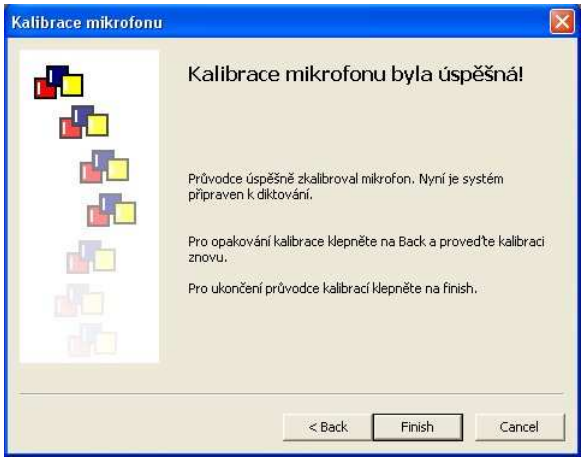
Obrázek 6.5: První okno průvodce kalibrací mikrofonu

V druhém okně průvodce bude zobrazen text určený k přečtení uživatelem. Zároveň bude zobrazena výzva k přečtení tohoto textu. Druhé okno je zobrazeno na obrázku 6.6.



Obrázek 6.6: Druhé okno průvodce kalibrací mikrofonu

Ve třetím a zároveň i posledním okně se zobrazí informace, zda kalibrace mikrofonu proběhla úspěšně, či nikoli. Toto okno je na obrázku 6.7.



Obrázek 6.7: Třetí okno průvodce kalibrací mikrofonu

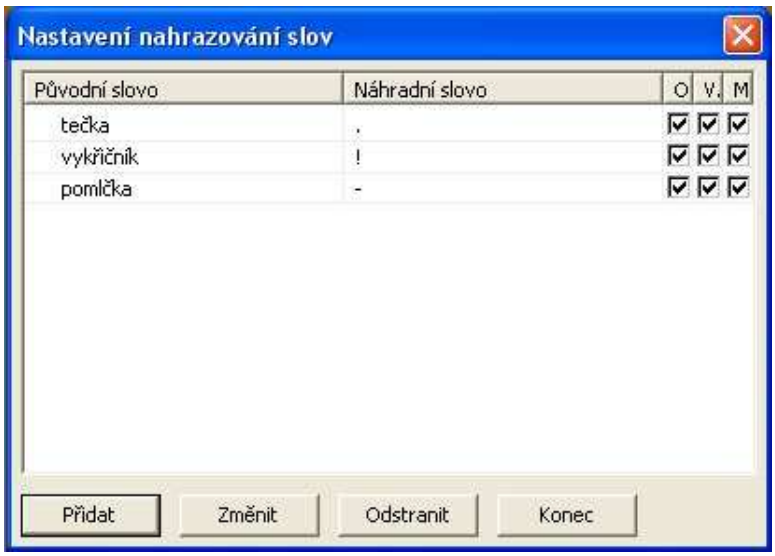
6.4 Nastavení nahrazování slov

V kapitole 5.1 bylo zmíněno, že nahrazování slov bude implementováno jako součást grafického uživatelského rozhraní. Proto je této problematice věnována právě tato kapitola.

Uživatel bude moci definovat vlastní pravidla pro nahrazování slov. Veškerá nastavení těchto pravidel budou ukládána do souboru, ze kterého se při spuštění aplikace načtou. Uživatel si bude moct zobrazit seznam všech pravidel, upravovat je i odstraňovat.

6.4.1 Seznam pravidel

Seznam všech pravidel definovaných uživatelem bude možné zobrazit přes položku v menu nebo v nástrojové liště programu. Po otevření se zobrazí okno na obrázku 6.8.



Obrázek 6.8: Okno seznamu pravidel nahrazování

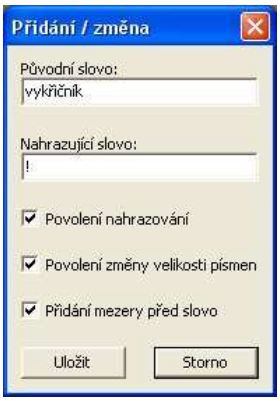
První sloupec bude obsahovat slovo, které má být nahrazeno nahrazujícím slovem (znakem nebo také prázdným řetězcem) ve druhém sloupci. Třetí, čtvrtý a pátý sloupec budou obsahovat zaškrtávací pole, jejichž zaškrtnutí bude možné měnit přímo ze seznamu. Třetí sloupec povoluje použití pravidla, tzn., pravidlo může být definované, ale nemusí být používané. Čtvrtý sloupec povoluje změnu velikosti písmen a pátý sloupec povoluje vložení mezery před toto slovo. Tento sloupec je zde pro diakritická znaménka, která pak budou umístěna hned za předchozí slovo.

Vzhledem k tomu, že uživatel může definovat nová pravidla a velikost okna seznamu nebude muset stačit k zobrazení všech pravidel, bude celý seznam skrolovatelný pomocí scrollbarů, které se zobrazí až v situaci, kdy by nebyly zobrazeny všechny pravidla.

Spodní část tohoto okna bude obsahovat čtyři tlačítka. Jedno pro přidávání pravidel, druhé pro jejich editaci, třetí pro mazání pravidel a čtvrté pro zavření seznamu.

6.4.2 Přidávání a editace pravidel

Přidávání a editace pravidel budou mít společné okno, které je zobrazeno na obrázku 6.9.



Obrázek 6.9: Okno pro přidání /editaci pravidla

V tomto okně budou dvě textová pole, první pro nahrazované slovo a druhé pro nahrazující slovo. Pod těmito poli budou umístěny tři zaškrtávací pole – pro povolení pravidla, změnu velikosti písma a vložení mezery před slovo.

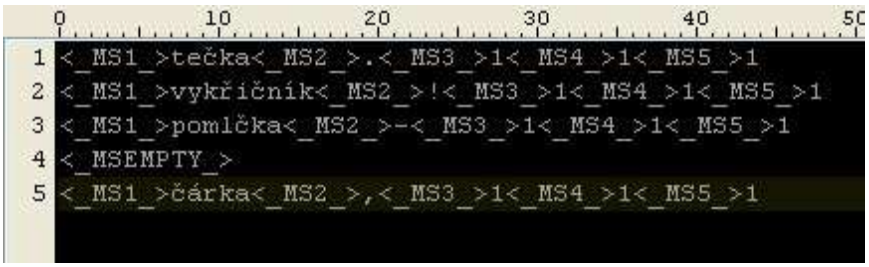
Jestliže se bude přidávat nové pravidlo, budou zaškrtnuta všechna zaškrtávací pole a textová pole budou prázdná. Když se bude pravidlo editovat, budou všechny položky v tomto okně vyplněny aktuálním nastavením editovaného pravidla.

V dolní části budou dvě tlačítka. Jedno pro potvrzení a uložení změn (nebo nového pravidla) a druhé pro zahození změn.

6.4.3 Ukládání pravidel nahrazování slov do souboru

Aby mohly být pravidla uchovávaná také po ukončení programu, musí se jejich nastavení ukládat do souboru, ze kterého se při spuštění aplikace načtou. Tato kapitola se zabývá návrhem formátu tohoto souboru.

Na každém řádku bude obsaženo právě jedno pravidlo. Na tomto řádku se pak vyskytuje pět značek, za nimiž bezprostředně následuje aktuální hodnota. Pořadí značek na řádku je zobrazeno na obrázku 6.10.



Obrázek 6.10: Příklad souboru s uloženými pravidly

Za značkou <_MS1_> následuje nahrazované slovo, za <_MS2_> je umístěno nahrazující slovo, za <_MS3_> je hodnota pro povolení vložení mezery před slovo. Po značce <_MS4_> následuje hodnota pro povolení pravidla a konečně za <_MS5_> je umístěna hodnota pro povolení změny velikosti písmen.

Na samostatném řádku se může vyskytnout značka <_MSEMPTY_>. Tato značka se umístí na řádek pravidla, které bylo uživatelem odstraněno. Při příštím spuštění programu se pak tento řádek ze souboru zcela odstraní. Tento princip byl zvolen, protože si aplikace bude uchovávat informaci o číslu řádku každého pravidla. Kdyby se tento řádek odstranil ihned při odstranění pravidla, byla by neplatná čísla všech řádků, která následují za odstraněným řádkem.

6.5 Přenositelnost vytvořených textů

Aplikace bude schopná přenášet texty mezi jinými programy dvěma způsoby:

- Přenos textu přes schránku do jiného programu
- Uložení dokumentu do souboru

Přenos textu přes schránku do jiného programu bude implementován pomocí klávesových zkratek Ctrl+C a Ctrl+X. Implementováno bude také vkládání textu pomocí klávesové zkratky Ctrl+V a také klávesová zkratka Ctrl+A bude podporována.

7 Implementace uživatelského rozhraní

Na základě návrhu grafického uživatelského rozhraní provedeného v 6. kapitole byla provedena implementace diktovacího systému. Pro implementaci byly použity nástroje popsány ve 3. kapitole.

7.1 Hlavní okno aplikace

Hlavní okno aplikace, které se zobrazí uživateli po spuštění programu je implementováno třídou `MyFrame`, jejíž uživatelskou oblast pro psaní textů implementuje třída `TextWindow`.

7.1.1 Třída `MyFrame`

Tato třída dědí hned tři třídy – `wxFrame`, `STransCallbackI` a `SErrorHandlerI`. Třídy `STransCallbackI` a `SErrorHandlerI` zajišťují komunikaci grafického uživatelského rozhraní s rozpoznávačem popsanou v kapitole 4.2. Třída `wxFrame` je součástí frameworku wxWidgets a je to jedna ze tříd, které mohou být použity jako hlavní okno aplikace.

Třída `MyFrame` obsahuje instanci rozpoznávače, stará se o jeho inicializaci, práci s ním a při ukončení programu také o uvolnění jím zabírané paměti. Také implementuje ukládání logů, varovných hlášení a chyb přicházejících od rozpoznávače do souborů pro pozdější analýzu jeho činnosti. Obsahuje instanci třídy `Translator` popsané v kapitole 7.3.1, které předává voláním její veřejné metody `RepairWord` všechna slova přicházející od rozpoznávače.

Protože se jedná o hlavní okno aplikace, je v této třídě umístěno menu, toolbar, statusbar. Také zajišťuje správu těchto prvků.

7.1.2 Třída `TextWindow`

Třída `TextWindow` implementuje uživatelskou oblast třídy `MyFrame`. Dědí třídu `wxSplitterWindow` a obsahuje instanci třídy `wxRichTextCtrl` usnadňující implementaci textových editorů. Obě tyto třídy jsou součástí frameworku wxWidgets.

Text je v paměti uchováván ve formátu RTF a je podporováno ukládání vytvořeného dokumentu ve třech typech souborů (txt, html a xml). Otevření dokumentu je podporováno ve dvou typech souborů (txt a xml). Jestliže je text uložen jako xml soubor, uloží se také formátování celého dokumentu, které se při znovuotevření dokumentu načte a aplikace toto formátování zobrazí.

Ve třídě je implementována změna fontu rozpoznávače i vstupu z klávesnice. Tyto fonty jsou na sobě navzájem nezávislé, takže diktovaný text může být zobrazen jiným fontem než text psaný na klávesnici.

Také implementuje změnu barvy pozadí a v neposlední řadě veškeré formátování psaného i diktovaného textu.

7.2 Průvodce kalibrací mikrofonu

Kalibrace mikrofonu byla implementována pomocí čtyř tříd – `MicWizard`, `FirstPage`, `SecongPage` a `ThirdPage`. Vlastní průvodce kalibrací je tvořen třídou `MicWizard` a jednotlivé stránky tohoto průvodce mají na starosti zbylé tři třídy.

Třída `MicWizard` dědí `wxWizard` z frameworku wxWidgets, která usnadňuje programování uživatelských průvodců. Ve třídě `MicWizard` jsou instance jednotlivých stránek (jejich tříd) a ta má na starost jejich správu.

Další tři třídy dědí `wxWizardPageSimple` pro implementaci stránek uživatelského průvodce. Při přechodu na druhou stránku se spustí kalibrace mikrofonu, která se ukončí při přechodu na poslední stránku průvodce. Té je předán výsledek kalibrace, na jehož základě je zobrazena zpráva o úspěšnosti, či neúspěšnosti kalibrace mikrofonu.

7.3 Nastavení nahrazování slov

Nastavení nahrazování slov je implementováno pomocí tří tříd. Jedná se o třídy `Translator`, `MyListCtrl` a `MyListItem`.

7.3.1 Třída `Translator`

Hlavní třída implementující okno nastavení nahrazování a veškerou logiku nahrazování. Třída dědí `wxDialog` obsažený ve frameworku wxWidgets.

Protože se jedná o hlavní třídu nahrazování slov a vlastně také o hlavní třídu implementující zpracování výstupu rozpoznávače, obsahuje instanci třídy `GenNumbers` naprogramovanou jako generátor čísel popsaný v kapitole 5.2. Tomuto generátoru čísel předává voláním metody `Generate` třídy `GenNumbers` všechna slova přicházející od rozpoznávače a ten na základě těchto slov generuje čísla nebo vrací mu předaná slova zpět.

Dle návrhu v kapitole 6.4.3 implementuje ukládání pravidel nahrazování slov do souboru a také jejich opětovné načtení při spuštění aplikace. V neposlední řadě je v rámci třídy naprogramováno chování tlačítek pro přidávání, změnu a mazání pravidel. Také je implementováno tlačítko pro uzavření okna.

Dále obsahuje instance tříd `MyListCtrl` a `MyListItem`, které jsou popsány dále v kapitolách 7.3.2 a 7.3.3.

7.3.2 Třída `MyListCtrl`

Třída `MyListCtrl` má na starost vlastní zobrazení seznamu nahrazování a obsluhu zaškrtávacích polí, kdy stačí pouze kliknout na zaškrtávací pole, u nějž chceme změnit jeho hodnotu.

Třída dědí `wxListCtrl` frameworku wxWidgets. Jedná se o vysokoúrovňovou třídu pro snadnou implementaci takovýchto seznamů.

7.3.3 Třída `MyListItem`

Poslední ze tříd grafického uživatelského rozhraní pro nahrazování slov dědící `wxDialog`. Je volána ze třídy `MyListCtrl` pro přidání nebo změnu pravidla nahrazování.

Její součástí je veřejná metoda `MyShow` pro zobrazení dialogu. Této metodě se předají reference na proměnné, do nichž bude uloženo nové nastavení nebo se z nich načte a zobrazí aktuální nastavení pravidla, které se bude měnit.

Třída také implementuje chování tlačítka pro uložení a pro stornování změn.

7.4 Postup při implementaci

V první řadě se vytvořila třída `MyFrame` spravující hlavní okno aplikace. Do tohoto okna poté byly přidány hlavní ovládací prvky aplikace – menu, toolbar (nástrojová lišta) a statusbar (stavový řádek). Uživatelská oblast pro psaní textů byla do tohoto okna přidána až po jejím naprogramování.

V dalším kroku byla naprogramována třída `TextWindow`, která spravuje uživatelskou oblast v hlavním okně aplikace. Po implementování této třídy, byla v `MyFrame` naprogramovaná komunikace s rozpoznávačem. Komunikace s rozpoznávačem se implementovala až v tomto kroku pro snadné ověření správné komunikace vláken aplikace – výstup rozpoznávače se již zobrazoval přímo v uživatelské oblasti hlavního okna.

Poté přišla na řadu třída generátoru čísel (`GenNumbers`) a vzápětí byly naimplementovány také třídy `MyListCtrl` a `Translator`. V závěru byla vytvořena třída `MyListItem`.

7.5 Komplikace při implementaci

Během implementace se nevyskytly žádné závažnější komplikace, které by zásadním způsobem ovlivnily programování. Jediný složitější problém se týkal komunikace grafického uživatelského rozhraní s rozpoznávačem, kterému se věnují kapitoly 4.2.1 a 4.2.2.

Drobnější komplikace byly způsobeny kódováním výstupu rozpoznávače, který nebyl v unicodu. Tento problém byl vyřešen přeinstalováním wxWidgets bez kódování v unicodu a překódováním všech zdrojových textů do kódování CP-1250.

Poslední drobnou komplikací, která stojí za zmínku, bylo nesprávné zpracování vstupu z klávesnice třídou `wxRichTextCtrl`, kdy se nezobrazovaly některé znaky s diakritikou. Tento problém byl však vyřešen současně s vyřešením kódování výstupu rozpoznávače.

8 Ověření funkčnosti

Vlastní testování programu bylo rozděleno do čtyř hlavních částí:

- Otestování komunikace s rozpoznávačem
- Otestování oprav slov
- Otestování generování čísel
- Otestování grafického uživatelského rozhraní

8.1 Otestování komunikace s rozpoznávačem

Primárním cílem této fáze testování bylo ověření, zda přicházejí od rozpoznávače slova, aniž by docházelo k zatuhnutí aplikace kvůli problémům se synchronizací vláken. Dále bylo potřeba ověřit, jestli je v pořádku kódování přicházejících slov – správná diakritika.

Sekundárním cílem bylo ověření podávání zpráv o stavu rozpoznávače primárnímu vláknu aplikace.

Vlastní testování proběhlo spuštěním programu, kalibrací mikrofonu, spuštěním rozpoznávání a následným diktováním. Tím ovšem testování nekončí, neboť může dojít k chybě také při ukončení rozpoznávání. Proto bylo rozpoznávání ukončeno a následně byl ukončen i program. V úplném závěru se provedla kontrola souborů, do nichž se ukládají log záznamy, varovná hlášení a hlášení o chybách.

8.2 Otestování nahrazování slov

Testování oprav slov proběhlo také v několika krocích. Nejdříve se pomocí grafického uživatelského rozhraní nadefinovaly pravidla pro opravy slov, poté se spustilo diktování a v diktovaném textu se vyskytovala slova, pro která byla v prvním kroku definovaná pravidla oprav.

V rámci této fáze testování se také provedla kontrola souboru, do nějž se ukládají uživatelem definovaná pravidla. Kontrola se týkala přítomnosti všech pravidel přidaných v prvním kroku fáze.

8.3 Otestování generování čísel

Tato fáze testování proběhla zároveň s první a druhou fázi testování, kdy se v rámci diktovaného textu vyskytovaly čísla. Tato čísla byla volena tak, aby prošla všemi stavy generátoru čísel.

8.4 Otestování uživatelského rozhraní

Testování grafického uživatelského rozhraní bylo poslední fází testování a bylo zaměřeno na otestování funkčnosti menu a všech jeho položek a následně i toolbaru a všech jeho tlačítek.

Testování proběhlo opět v několika krocích. Spustil se program a zkontrolovala se funkčnost všech tlačítek umístěných na toolbaru. Poté se prošla jednotlivá menu a zkontrolovaly se všechny jeho položky. Zároveň se v rámci tohoto testování provedlo ověření funkčnosti všech dialogů a tlačítek v grafickém uživatelském rozhraní.

9 Závěr

Myšlenka ovládaní počítačů, strojů a přístrojů hlasem je v současnosti ve velkém zájmu odborníků a proto se jí oplatí zabývat. V této práci byla představena jedna z aplikací, která v této oblasti není jedinečná, ale je dostupná široké veřejnosti. Byl rozebrán návrh a implementace grafického uživatelského rozhraní tohoto diktovacího systému a byl nastíněn směr, kterým by se mohl ubírat další vývoj této aplikace.

9.1 Zhodnocení výsledné aplikace

Aplikace byla vytvořena s ohledem na požadavky zadání práce, jednoduchost používání a snadné pochopení práce s ní.

Celý program je navržen a naprogramován tak, aby byl snadno rozšiřitelný o další funkčnost. Uživatelské rozhraní bylo implementováno nezávisle na rozpoznávači. Jestliže se změny v rozpoznávači nebudou týkat tříd implementujících komunikaci s uživatelským rozhraním (`STransCallBackI` a `SErrorHandlerI`), nebude potřeba žádným způsobem zasahovat do kódu uživatelského rozhraní.

Jednoduché grafické uživatelské rozhraní má snadné ovládání a velké možnosti nastavení. Osobně jsem s výsledným programem spokojen.

9.2 Přínos pro mě

Vypracování této práce mě zdokonalilo v programování v jazyce C++ a v používání frameworku wxWidgets, kde jsem se seznámil s třídou `wxRichTextCtrl` a `wxListCtrl`, které mi hodně ulehčily implementaci grafického uživatelského rozhraní.

Naučil jsem se rozvrhnout si práci a čas na projektu rozsahu, jenž byl mnohem větší než projekty zadané v rámci povinných i volitelných předmětů.

Hlavním přínosem pro mě však bylo osvojení si technik návrhu a implementace grafického uživatelského rozhraní.

V neposlední řadě také musím zmínit přiučení se zpracování technických textů, dodržování stylistických, typografických a jazykových zásad, pravidel a norem.

9.3 Možná rozšíření do budoucna

Přestože je aplikace kompletní a obsahuje spoustu ovládacích prvků, nabízí mnoho možností pro rozšíření v budoucnu. Může se zvážit hlasové ovládání celého programu včetně formátování diktovaného textu. Za úvahu také stojí hlasové ovládání operačního systému a jiných programů rozebírané v kapitole 5.3.1, které sebou však nese určité komplikace popsané v téže kapitole.

Možná rozšíření grafického uživatelského rozhraní se můžou týkat editace více dokumentů zároveň pomocí záložek. Formátování textu může být rozšířeno o další možnosti formátování, podporu vkládání obrázků. Nakonec by mohlo být podporováno více formátů (typů souborů) pro uložení vytvořených dokumentů. Vzhledem k tomu, že spuštění aplikace trvá poměrně dlouhou dobu, bylo by vhodné zobrazit při spuštění aplikace informaci o stavu načítání dat rozpoznávače.

Literatura

[1] Nuance: Dragon NaturallySpeaking 10 Preffered [online]. Nuance Communications c2002-2009, [cit. 2009-05-12]. Dostupné na URL: <http://www.nuance.com/naturallyspeaking/products/preferred.asp>

[2] IBM: Embedded ViaVoice [online]. IBM c1997-2008, [cit. 2009-05-12]. Dostupné na URL: http://www-01.ibm.com/software/pervasive/embedded_viavoice/

[3] Smart, J.; Hock, K.; Csomor, S.: Cross-Platform GUI Programming with wxWidgets. První vydání, 2005. ISBN 0-13-147381-6.

Seznam příloh

Příloha 1. CD obsahující zdrojové kódy, programovou dokumentaci a uživatelský manuál