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

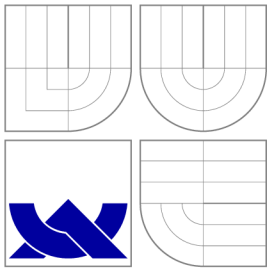
HLASEM OVLÁDANÁ KALKULAČKA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

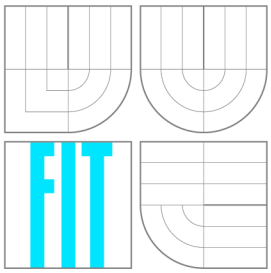
AUTOR PRÁCE
AUTHOR

OTA PAVELEK

BRNO 2011



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

HLASEM OVLÁDANÁ KALKULAČKA

VOICE CONTROLLED CALCULATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — OTA PAVELEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. FRANTIŠEK GRÉZL, Ph.D.
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce se zabývá implementací kalkulačky. Tuto kalkulačku lze ovládat jak hlasem, tak běžným způsobem. Rozpoznávání řeči je realizováno pomocí knihovny `BSCORE`. K rozpoznávači je vytvořena rozpoznávací síť, která rozpoznává pouze potřebná slova. Po rozpoznání zadaného výrazu se rozpoznaný výraz zobrazí uživateli, aby jej mohl uživatel (zejména v případě chybného rozpoznání) upravit. Výpočet výrazu probíhá na žádost uživatele. Cílem hlasového ovládání je učinit používání kalkulačky efektivnější a přístupnější pro handicapované uživatele.

Abstract

This thesis describes implementation of calculator, which can be controlled by both voice and normal way. Speech recognizing is realized with `BSCORE` library and has recognition network restricted to words needed by calculator. When the recognition is done, recognized sentence is transformed to expression and shown to user, so it can be corrected (especially in case of erroneous recognition). Expression is evaluated on user's request. The purpose of voice control is to make usage of calculator more effective and accesible for handicapped users.

Klíčová slova

kalkulačka, rozpoznávání řeči, hlasové ovládání

Keywords

calculator, speech recognition, voice control

Citace

Ota Pavelek: Hlasem ovládaná kalkulačka, bakalářská práce, Brno, FIT VUT v Brně, 2011

Hlasem ovládaná kalkulačka

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Františka Grézla, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ota Pavelek
17. května 2011

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce panu Ing. Františku Grézlovi, Ph.D. za odborné vedení, rady a čas, které mi při tvorbě práce poskytl. Zároveň bych chtěl poděkovat všem, kteří mě při práci podporovali.

Obsah

1 Úvod **4**

2 Rozpoznávače řeči **5**
2.1 Borcení časové osy . . . 5
2.2 Rozpoznávače založené na skrytých Markovových modelech . . . 5
2.2.1 Celoslovní skryté Markovovy modely . . . 6
2.2.2 Skryté Markovovy modely modelující fonémy . . . 6
2.3 Použitý rozpoznávač . . . 6

3 Návrh základní verze kalkulačky **7**
3.1 Požadavky na základní verzi kalkulačky . . . 7
3.2 Návrh základní verze kalkulačky . . . 8
3.2.1 Návrh slovníku . . . 8
3.2.2 Objektový návrh aplikace . . . 8
3.2.3 Návrh grafického uživatelského rozhraní . . . 9

4 Implementace základní verze kalkulačky **10**
4.1 Implementační nástroje . . . 10
4.1.1 Framework využitý pro tvorbu aplikace . . . 10
4.1.2 Knihovny pro nahrávání zvuku . . . 10
4.1.3 Rozpoznávač . . . 11
4.2 Tvorba rozpoznávací sítě . . . 11
4.2.1 Výslovnostní slovník . . . 11
4.2.2 Jazykový model . . . 11
4.3 Implementace tříd základní verze aplikace . . . 11
4.3.1 Třída `Recorder` . . . 12
4.3.2 Třída `SOfflineSpeechRecognizerI` . . . 12
4.3.3 Třída `Ui_MainWindow` . . . 13
4.3.4 Třída `MainWindow` . . . 13
4.4 Zhodnocení základní verze kalkulačky . . . 16

5 Návrh rozšířené verze kalkulačky **18**
5.1 Požadavky na rozšířenou verzi kalkulačky . . . 18
5.2 Návrh rozšířené verze kalkulačky . . . 18
5.3 Návrh slovníku . . . 19
5.4 Objektový návrh aplikace . . . 19
5.5 Návrh grafického uživatelského rozhraní . . . 19

6 Implementace rozšířené verze kalkulačky **21**
6.1 Implementační nástroje . . . 21
6.2 Tvorba rozpoznávací sítě . . . 21
6.2.1 Výslovnostní slovník . . . 21
6.2.2 Jazykový model . . . 23
6.3 Implementace tříd rozšířené verze aplikace . . . 23
6.3.1 Třída `Recorder` . . . 23
6.3.2 Třída `Recognizer` . . . 25
6.3.3 Třída `ExpressionManager` . . . 25
6.3.4 Třída `Ui_MainWindow` . . . 27
6.3.5 Třída `HelpWindow` . . . 27
6.3.6 Třída `Configuration` . . . 27
6.3.7 Třída `MainWindow` . . . 28
6.4 Zhodnocení rozšířené verze kalkulačky . . . 29

7 Testování **30**
7.1 Průběh testovaní . . . 30
7.2 Zhodnocení přesnosti rozpoznávače . . . 30
7.3 Uživatelská odezva . . . 30

8 Závěr **31**

A Návod k použití **33**
A.1 Zadání výrazu . . . 33
A.2 Příkazy pro ovládání kalkulačky . . . 33
A.3 Oprava výrazu . . . 33
A.4 Nápověda . . . 34
A.5 Přiřazení hodnoty do proměnné . . . 34
A.6 Tlačítka grafického uživatelského rozhraní . . . 34
A.7 Klávesové zkratky . . . 34

B Podrobné výsledky testování **35**

Seznam obrázků

3.1 Znázornění fungování základní verze kalkulačky. 7
3.2 Objektový návrh základní verze kalkulačky. 8

4.1 Grafické zobrazení interakce tříd v základní verzi kalkulačky. 13
4.2 Grafické uživatelské rozhraní základní verze aplikace. 14
4.3 Přehled metod a členských proměnných třídy `MainWindow`. 15
4.4 Znázornění převodu rozpoznaných slov na výraz zobrazený uživateli. 15

5.1 Objektový návrh rozšířené verze kalkulačky. 20

6.1 Třída `Recorder`. 23
6.2 Třída `Recognizer`. 25
6.3 Znázornění zpracování výrazu před výpočtem. 26
6.4 Grafické uživatelské rozhraní rozšířené verze aplikace. 28
6.5 Třída `HelpWindow`. 28
6.6 Třída `Configuration`. 29

Kapitola 1

Úvod

Hlasové ovládání se v poslední době dostává do popředí v oblasti uživatelských rozhraní [2]. Má obecně za cíl zefektivnit komunikaci (zejména nezkušených) uživatelů s aplikací. Místo pomalé manipulace s klávesnicí a myší komunikuje uživatel s aplikací přirozeným jazykem. Tento systém se dnes používá například v rezervačních systémech, callcentrech a mobilních telefonech [3]. Použitelnost systému se odvíjí od přesnosti použitého rozpoznávače. Přesnost rozpoznávačů mimo jiné závisí na tom, má-li rozpoznávač poznávat hlasy více lidí nebo jen jednoho člověka. Objevuje se též snaha o použití rozpoznávání hlasu pro autentizaci – například projekt MOBIO[1]. Hlasové ovládání může nacházet využité také u handicapovaných uživatelů, kteří nemohou plně ovládat klávesnici nebo myš.

Cílem této bakalářské práce bylo implementovat kalkulačku, která by byla šla kompletně ovládat hlasem. Pro rozpoznávání řeči byl použit rozpoznávač z balíku `BSCORE`. V první fázi práce měla být implementována kalkulačka, která by prováděla pouze základní operace s čísly. Těmito základními operacemi bylo myšleno sčítání, odčítání, násobení a dělení. Základní verze kalkulačky uměla také zaměňovat prioritu operátorů použitím závorek. Základní verze však neuměla pracovat s desetinnými čísly. V další fázi práce měla být tato kalkulačka rozšířena o další operace a funkce. Byly přidány operace faktoriál a mocnina s libovolným exponentem. Kromě těchto operací byly přidány i goniometrické funkce, cyklometrické funkce a logaritmus se základem 10. Goniometrické funkce mohou svoje parametry příjmat buď ve stupních, radiánech nebo gradiánech. Rozšířená verze již uměla pracovat s desetinnými čísly. Po implementaci těchto dalších funkcí měla být kalkulačka otestována na více uživatelích. Kromě ovládání hlasem by měla mít kalkulačka i běžné grafické uživatelské rozhraní. Implementace kalkulačky měla být provedena v programovacím jazyce `C++`.

Práce je členěna do několika kapitol. V kapitole 2 jsou popsány různé druhy rozpoznávačů. Podrobněji je v ní popsán rozpoznávač použitý pro tuto bakalářskou práci. Kapitoly 3 a 4 jsou věnovány základní verzi kalkulačky. Obsahují popis návrhu a implementace této kalkulačky. Kapitoly 5 a 6 se zabývají rozšířenou verzí kalkulačky. Kromě popisu návrhu a implementace této rozšířené verze se věnují zejména odlišnostem od základní verze. Testování uživateli je shrnuto v kapitole 7. Kromě údajů o přesnosti rozpoznávače obsahuje kapitola i shrnutí názorů uživatelů na možnost ovládat kalkulačku hlasem. V poslední části 8 jsou shrnuty dosažené výsledky a zhodnocení z pohledu dalšího vývoje.

[1] Více informací můžete nalézt na http://www.mobioproject.org.

Kapitola 2

Rozpoznávače řeči

Tato kapitola je věnována existujícím principům pro tvorbu rozpoznáváčů řeči. Podrobněji je zde popsaný rozpoznávač z balíku `BSCORE`, který byl použit pro tvorbu této bakalářské práce. Významným zdrojem informací pro tuto kapitolu je studijní opora [10], ve které jsou principy detailně popsány. Při rozpoznávání se bere v úvahu fakt, že uživatel neřekne nikdy dvě slova stejně. V nahrávce řeči se navíc mohou vyskytnout šumy okolí. Existuje několik různých přístupů pro realizaci počítačového rozpoznávání řeči. Pro rozpoznávání řeči jednoho konkrétního řečníka je vhodné tzv. **borcení časové osy** (*Dynamic Time Warping*). Pro rozpoznávání řeči různých řečníků jsou vhodné **skryté Markovovy modely** (*Hidden Markov Models*). V dnešních rozpoznávačích se používá skrytých Markovových modelů.

2.1 Borcení časové osy

Borcení časové osy je založeno na porovnávání dvou promluv. Rozpoznávač má k dispozici sadu vzorů slov. Vůči této sadě vzorů je porovnávána nahraná promluva. Při porovnávání je počítána vzdálenost příznaků řečového vstupu od příznaků prototypu slova. Pro úspěšné použití této metody je potřeba v řečovém vstupu správně provádět rozlišování na jednotlivá slova. Pro více informací viz například [1, 6, 7]. Výhodou borcení časové osy je to, že vzory může namluvit uživatel. Jedná se tedy o poměrně spolehlivou metodu, která je výpočetně nenáročná. Borcení časové osy nepotřebuje žádná trénovací data.

2.2 Rozpoznávače založené na skrytých Markovových modelech

Rozpoznávač založený na skrytých Markovových modelech může být určen pro rozpoznávání izolovaných slov, spojených slov nebo pro rozpoznávání plynulé řeči. Jedná se o statistický model, který se musí natrénovat. Stavy skrytého Markovova modelu reprezentují úseky slova. Každý úsek je modelován statistickým modelem – směsicí Gaussových rozdělení. U Gaussových rozdělení se předpokládá statistická nezávislost koeficientů v příznakovém vektoru. Používají se proto pouze Gaussova rozdělení s diagonální kovariační maticí. Skrytý Markovův model může reprezentovat celé slovo, pro každé slovo je pak trénován celý model.

Hledání nejpravděpodobnější cesty modelem probíhá tak, že se o nejlepší cestě rozhoduje průběžně při průchodu modelem. Počítání pravděpodobností všech jednotlivých cest před průchodem modelem by bylo zbytečně výpočetně náročné. Průchod modelem je implementován algoritmem *token passing*. Tento algoritmus prochází modelem a počítá prav-

děpodobnosti cest v modelu. Jako slovo je rozpoznán model, který generuje vstupní vektor s největší pravděpodobností.

2.2.1 Celoslovní skryté Markovovy modely

Pro rozpoznávání izolovaných slov pomocí skrytých Markovových modelů potřebujeme slovník, který obsahuje N slov. Ke každému slovu máme natrénovaný model M. Při trénování modelu slova potřebujeme dostatečný počet výskytů tohoto slova v trénovacích nahrávkách. Cílem je nalezení modelu, který bude nejvěrohodněji modelovat vstupní sekvenci.

Rozpoznávání spojených slov využívá první a poslední stav skrytého Markovova modelu. Tyto stavy jsou při rozpoznávání izolovaných slov nevyužity. Při rozpoznávání spojených slov jsou tyto dosud nevyužité stavy využity k napojení na jiné modely.

2.2.2 Skryté Markovovy modely modelující fonémy

Skryté Markovovy modely modelující fonémy jsou vhodné pro rozpoznávače s velkými slovníky. Pro modelování se tedy nepoužívají celá slova, ale používají se jednotlivé *fonémy* nebo *kontextově závislé fonémy*. Výhodou je, že slova obsažená ve slovníku nemusí být obsažena v trénovacích nahrávkách a přesto pro ně můžeme natrénovat model. Kontextově závislé fonémy jsou fonémy, které berou v úvahu fonémy v jejich okolí. Nejčastěji používané jsou *trifony*. Trifon bere v úvahu typ fonémů vyskytující před a za ním. Modely fonémů se nejdříve spoji do slov a tato slova se následně spojí do rozpoznávací sítě, která se prochází algoritmem *token passing*.

2.3 Použitý rozpoznávač

Použitý rozpoznávač je odvozen z rozpoznávače pro plynulou řeč s velkým slovníkem. Ten modeluje elementární řečové jednotky – kontextově závislé fonémy – pomocí skrytých Markovových modelů. Z těchto jednotek se pak skládají slova podle tzv. *výslovnostního slovníku*. Napojení slov na sebe se řídí tzv. *jazykovým modelem* udávajícím pravděpodobnosti N-tic slov. Vše je skompilováno do jedné rozpoznávací sítě, kterou pak dekódovací algoritmus hledá nejpravděpodobnější cestu [9]. Vstupem rozpoznávače je nahrávka ve formátu `.wav` $8kHz$ MONO PCM, jeho výstupem je sekvence rozpoznaných slov.

Pro potřeby kalkulačky byl slovník omezen pouze na slova, která byla pro kalkulačku nutná. Také byla vytvořena nová rozpoznávací síť, která umožňuje libovolnou posloupnost slov. To výrazně urychlí proces rozpoznávaní a zaručí rychlou odezvu i při zpracování dlouhé nahrávky. Správnost posloupnosti rozpoznaných slov je kontrolována při výpočtu výrazu. Rozpoznávač je zapouzdřen do rozhraní `BSAPI`, které je volně dostupné pro nekomerční použití[1].

[1] Dokumentace dostupná na http://www.phonexia.com/docs/bsapi/.

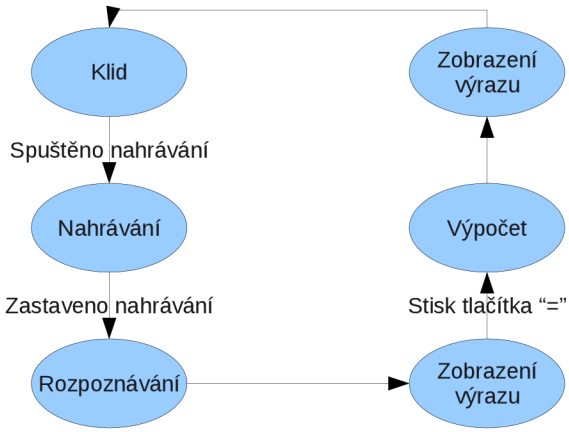
Kapitola 3

Návrh základní verze kalkulačky

V této kapitole je čtenáři představen popis základní verze kalkulačky. Nejdříve jsou zde popsány požadavky, které byly na tuto základní verzi aplikace kladeny. Poté je zde popsán návrh pro tvorbu slovníku, objektový návrh pro tvorbu aplikace, která by odpovídala stanoveným požadavkům, a návrh grafického uživatelského rozhraní. Implementace základní verze kalkulačky je popsána v samostatné kapitole 4.

3.1 Požadavky na základní verzi kalkulačky

Základním požadavkem bylo možnost zadávat výraz do kalkulačky za pomoci hlasu. Hlasový vstup by měl být pro uživatele co nejvíce intuitivní a pohodlný. Kromě ovládání hlasem měla mít kalkulačka i grafické uživatelské rozhraní. Jako implementační jazyk byl v zadání určen jazyk `C++`. Základní verze kalkulačky měla umět provádět základní operace sčítání, odčítání, násobení a dělení. Prioritu těchto operátorů mělo jít zaměňovat pomocí závorek. Tyto závorky by mělo jít do sebe libovolně zanořovat. Základní verze neumožňuje zadávání desetinných čísel. Popis činnosti kalkulačky je zobrazen na obrázku 3.1.



Obrázek 3.1: Znázornění fungování základní verze kalkulačky.

3.2 Návrh základní verze kalkulačky

Před samotnou implementací základní verze kalkulačky je potřeba provést návrh slovníku, který bude použit rozpoznávačem, objektový návrh aplikace a návrh grafického uživatelského rozhraní, pomocí kterého bude uživatel řídit nahrávání řeči a zadávat příkazy k výpočtu.

3.2.1 Návrh slovníku

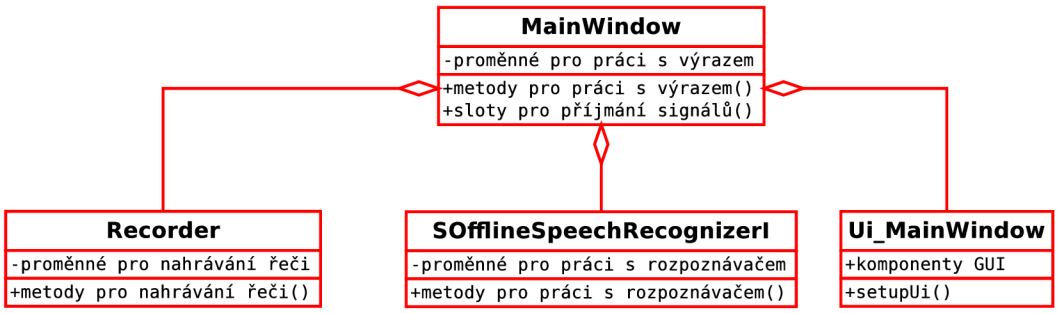
Vytvořit ve slovníku slovo pro každé možné číslo by bylo velmi neefektivní a zdlouhavé. Ve slovníku jsou proto vytvořena pouze slova k číslovkám. Převod posloupnosti těchto číslovek na číslo je realizováno na úrovni zdrojového kódu a je popsáno v kapitole 4. Kromě slov pro číslovky je potřeba vytvořit ještě slova pro implementované operátory (sčítání, odčítání, násobění a dělení) a slova pro levou a pravou závorku.

3.2.2 Objektový návrh aplikace

Aplikace má být implementována v programovacím jazyce `C++`, návrh je tedy objektový. Při návrhu je nutno zohlednit fakt, že bude potřeba nahrávat hlasový vstup od uživatele. Aby však aplikace tzv. *nezamrzala*, tak bude nahrávání hlasového vstupu realizováno v samostaném vlákně. Začátek a konec nahrávání hlasového vstupu bude probíhat na základě požadavku od uživatele. Uživatel bude tento požadavek zadávat prostřednictvím grafického uživatelského rozhraní, jehož návrhu je věnována sekce 3.2.3. Po ukončení nahrávání hlasového vstupu bude nahrávka předána rozpoznávači. Výstup rozpoznávače v podobě rozpoznaných slov bude potřeba zpracovat a zobrazit uživateli. Výpočet výrazu bude probíhat na základě příkazu od uživatele. Objektový návrh aplikace je zobrazen na obrázku 3.2.

Úkoly jednotlivých tříd:

- `Recorder` – nahrávání hlasového vstupu z mikrofonu, samostatné vlákno
- `SOfflineSpeechRecorderI` – rozpoznávač řeči z `BSAPI`
- `Ui_MainWindow` – uživatelské rozhraní
- `MainWindow` – řízení grafického uživatelského rozhraní, aplikace a operací s výrazem



Obrázek 3.2: Objektový návrh základní verze kalkulačky.

3.2.3 Návrh grafického uživatelského rozhraní

V grafickém uživatelském rozhraní pro tuto základní verzi kalkulačky bude potřeba existence tlačítek, pomocí kterých budou realizovány příkazy pro spuštění a ukončení nahrávání a pro výpočet výrazu. Další tlačítka budou standardní tlačítka jako u jiných kalkulaček, pomocí kterých půjde tato kalkulačka ovládat.

Kapitola 4

Implementace základní verze kalkulačky

Tato kapitola navazuje na kapitolu 3 o návrhu základní verze kalkulačky. Je zde popsána tvorba slovníku, implementace aplikace podle vytvořeného objektového návrhu a tvorba grafického uživatelského rozhraní. Na konci této kapitoly je uvedeno zhodnocení této verze kalkulačky.

4.1 Implementační nástroje

4.1.1 Framework využitý pro tvorbu aplikace

Jak již bylo zmíněno výše, jako implementační jazyk bylo zadáno `C++`. Na základě potřeby grafického uživatelského rozhraní byl vybrán Qt framework[1] od firmy Nokia. Tento framework je volně dostupný pro nekomerční použití. Navíc je objektově orientovaný a psaný v jazyce `C++`, což vyhovuje i požadavku na implementační jazyk. Kromě toho, že poskytuje mnoho tříd pro práci s grafickými prvky, tak poskytuje i mnoho jiných tříd. Zejména se jedná o třídy pro práci s vektory a jinými datovým strukturami, řetězci a také s procesy a vlákny. Tento framework je multiplatformní, funguje tedy na více operačních systémech, čímž je vyhověno požadavku na přenositelnost. Jedním z prvků tohoto frameworku jsou `signály` a `sloty`. Signály jsou vyslány funkci `emit` z nějaké třídy. Slot je metoda, jejíž kód je proveden jako reakce na přijatý signál. Signály a třídy, které je vysílají musí být propojeny se sloty a třídami, které tyto signály přijmají. Toto propojení je realizováno pomocí funkce `connect(odesilatel, signál, příjemce, slot)`. Příkladem použití signálu a slotu budiž situace, kdy uživatel klikne na tlačítko. Instance třídy, která realizuje zobrazení tlačítka vyšle signál značící stisk tohoto tlačítka. V instanci třídy, které je signál určen, je jako reakce na příjem signálu vykonán kód v patřičném slotu. Tento framework byl vybrán jako základ pro tuto aplikaci.

4.1.2 Knihovny pro nahrávání zvuku

Pro nahrávání hlasového vstupu z mikrofonu byly vyzkoušeny dvě knihovny. U obou knihoven byla provedena zkušební implementace programu pro nahrávání hlasového vstupu z mikrofonu a uložení tohoto hlasového vstupu na disk v požadovaném formátu. První zkušební

[1] Pro více informací viz `http://qt.nokia.com`.

implementace nahrávání byla provedena pomocí knihovny `RtAudio`[2]. Druhá zkušební implementace nahrávání byla provedena s využitím knihovny `FMOD`[3]. Použití v této aplikaci by umožňovaly licenční podmínky obou knihoven. Požadované parametry nahrávaného řečového vstupu (vzorkovací frekvence $8kHz$, MONO - 1 kanál, PCM - vzorky 16b čísla se znaménkem) se nastavují při inicializaci nahrávání a toto nastavení umožňovaly obě knihovny stejně pohodlně. Knihovna `FMOD` však poskytovala lepší rozhraní pro nahrávání, proto byla nakonec vybrána pro použití v této aplikaci. V ukázkových příkladech k této knihovně byla navíc i napsána funkce pro uložení souboru na disk v požadovaném formátu `.wav`.

4.1.3 Rozpoznávač

Použitý rozpoznávač je určen ze zadání této práce. Jedná se o rozpoznávač pro plynulou řeč z knihovny `BSCORE`[4]. Jak již bylo řečeno výše, tento rozpoznávač rozpoznává řeč uloženou v souboru ve formátu `wav` $8kHz$ MONO PCM. Výstupem je sekvence slov, které byly rozpoznány na základě rozpoznávací sítě vytvořené z výslovnostního slovníku. Tato rozpoznávací síť se ovšem musí nejdříve vyrobit z výslovnostního slovníku a jazykového modelu.

4.2 Tvorba rozpoznávací sítě

Aby mohla být vytvořena rozpoznávací síť, musí se prvně vyrobit výslovnostní slovník a jazykový model. Vytvořený výslovnostní slovník a jazykový model jsou skompilovány do jedné rozpoznávací sítě pomocí skriptu, který byl dodán s rozpoznávačem. Skript vytvoří rozpoznávací síť pomocí nástrojů knihovny `OpenFst`. Tuto rozpoznávací síť načítá rozpoznávač při své inicializaci.

4.2.1 Výslovnostní slovník

Výslovnostní slovník určuje, které fonémy budou tvořit konkrétní slovo. V tomto slovníku jsou vytvořeny výslovnosti pro číslovky, operátory a pro závorky. Přehled výslovností je uveden v tabulce 4.1.

4.2.2 Jazykový model

Jazykový model určuje pravděpodobnosti N-tic slov. Jazykový model, který byl vytvořen pro potřeby této verze kalkulačky, obsahuje pravděpodobnosti pouze pro jednotlivá slova (N = 1). Rozložení pravděpodobnosti slov určených pro kalkulačku je rovnoměrné.

4.3 Implementace tříd základní verze aplikace

Zpracování výrazu a ovládání grafického uživatelského rozhraní je implementováno ve třídě `MainWindow`. Nahrávání hlasového vstupu z mikrofonu je implementováno ve třídě `Recorder`. Třída `SOfflineSpeechRecognizerI` zajišťuje rozpoznávání řeči. Prvky grafického uživatelského rozhraní jsou uchovávány ve třídě `Ui_MainWindow`. Grafické zobrazení interakce tříd je zobrazeno na obrázku 4.1.

[2]Knihovna je dostupná na stránkách http://www.music.mcgill.ca/~gary/rtaudio/.
[3]Stránky této knihovny jsou k dispozici na www.fmod.org.
[4]Pro více informací viz http://www.phonexia.com/docs/bsapi/.

Slovo	Výslovnost	Slovo	Výslovnost
nula	[n u l a]	devatenáct	[d e v a t e n á s t]
jedna	[j e d n a]	dvacet	[d v a c e t]
dvě	[d v j e; d v a]	třicet	[t ř i c e t]
tři	[t ř i]	čtyřicet	[č t y ř i c e t]
čtyři	[č t y ř i]	padesát	[p a d e s á t]
pět	[p j e t]	šedesát	[š e d e s á t]
šest	[š e s t]	sedmdesát	[s e d m d e s á t]
sedm	[s e d m]	osmdesát	[o s m d e s á t]
osm	[o s m]	devadesát	[d e v a d e s á t]
devět	[d e v j e t]	sto	[s t o; s ť e; s t a; s e t]
deset	[d e s e t]	tisíc	[ť i s í c; ť i s í c e]
jedenáct	[j e d e n á s t]	milion	[m i l i j o n; m i l i j o n y; m i l i j o n ů]
dvanáct	[d v a n á s t]	plus	[p l u s]
třináct	[t ř i n á s t]	mínus	[m í n u s]
čtrnáct	[č t r n á s t]	krát	[k r á t]
patnáct	[p a t n á s t]	děleno	[ď e l e n o]
šestnáct	[š e s t n á s t]	pravá	[p r a v á]
sedmnáct	[s e d m n á s t]	levá	[l e v á]
osmnáct	[o s m n á s t]		

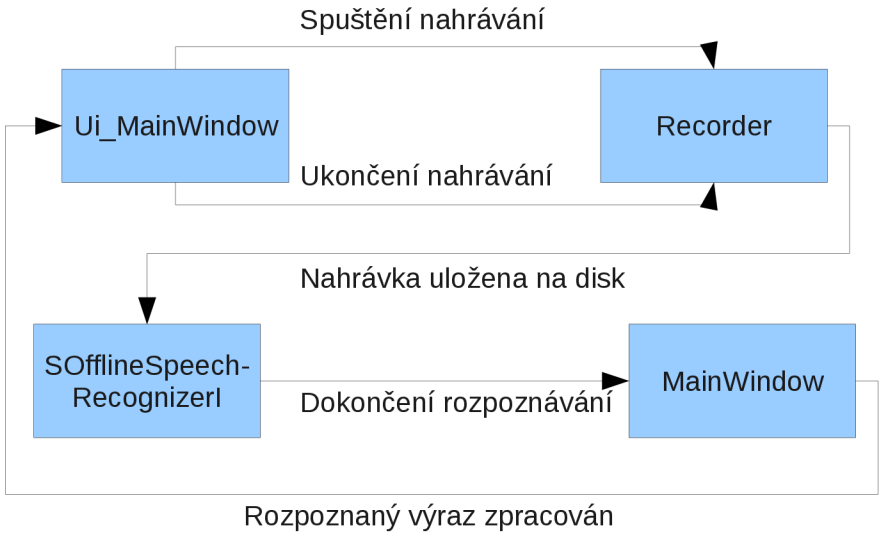
Tabulka 4.1: Přehled použitých slov a jejich výslovností v základní verzi kalkulačky.

4.3.1 Třída `Recorder`

Třída `Recorder` se veřejně dědí od třídy `QThread` z Qt frameworku. Tato dědičnost umožňuje nahrávání zvuku bez toho, aniž by byla samotná aplikace blokována. Pro spuštění nového vlákna je potřeba ve třídě `Recorder` reimplementovat metodu `run` třídy `QThread`. V této metodě `run` je implementována inicializace nahrávacího zařízení, ukládání zvukové nahrávky do souboru na disku a uvolnění zdrojů po dokončení nahrávání. Před samotným nahráváním je zapsána hlavička zvukového `.wav` souboru. Ukládání vzorků zvukové nahrávky do souboru na disku je realizováno průběžně při získávání hlasového vstupu z mikrofonu. Po dokončení nahrávání je hlavička souboru přepsána. Přepsání hlavičky souboru je nutné, protože hlavička obsahuje položku, která obsahuje počet vzorků v souboru [5]. Zapsání `.wav` hlavičky je realizováno metodou `WriteWavHeader`. Po uložení souboru s nahranou řečí a uvolnění všech zdrojů je do instance třídy `MainWindow` zaslán signál, který značí dokončení nahrávání.

4.3.2 Třída `SOfflineSpeechRecognizerI`

Rozpoznávání řeči je implementováno ve třídě `SOfflineSpeechRecognizerI`. Tato třída je jednou ze tříd poskytovaných rozhraním `BSAPI`. Rozpoznání řeči umožňuje její metoda `ProcessFile`. Tato metoda přijímá dva parametry. Prvním parametrem je název souboru s uloženou řečovou nahrávkou z mikrofonu, který byl vytvořen třídou `Recorder`. Druhým parametrem je název souboru, do kterého se uloží posloupnost rozpoznaných slov. Z této třídy nebylo pro potřeby kalkulačky potřeba reimplementovat žádnou metodu. Proces rozpoznávání není aplikací spouštěn v samostatném vlákně.



Obrázek 4.1: Grafické zobrazení interakce tříd v základní verzi kalkulačky.

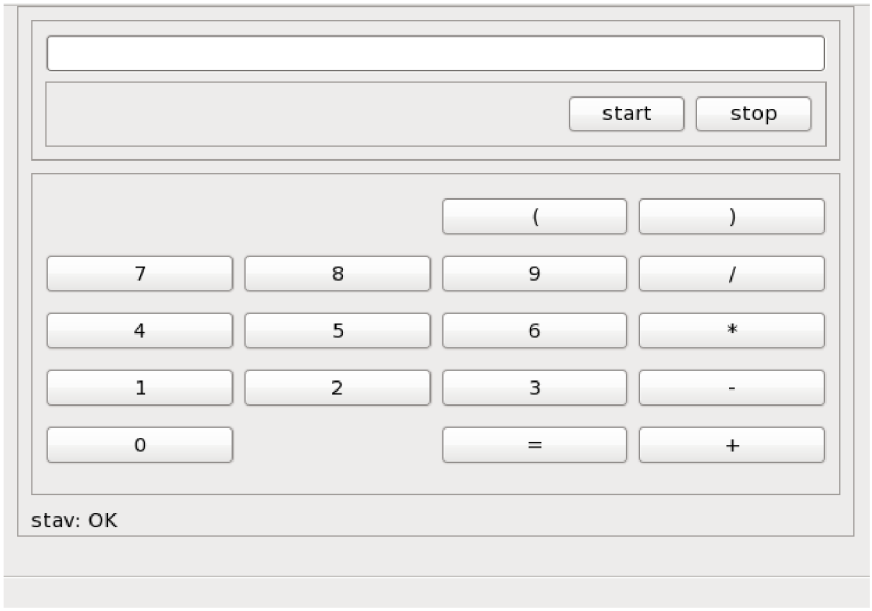
4.3.3 Třída `Ui_MainWindow`

Prvky grafického uživatelského rozhraní jsou uloženy ve třídě `Ui_MainWindow`. Tato třída byla vytvořena pomocí programu `Qt Designer`. Stará se o alokaci a počáteční nastavení jednotlivých prvků. Grafické uživatelské rozhraní je ukázáno na obrázku 4.2. Toto uživatelské rozhraní obsahuje řádek, ve kterém se zobrazuje rozpoznaný výraz. Řádek pro zobrazení výrazu je editovatelný, uživatel tedy může zobrazený výraz libovolně měnit za pomocí klávesnice. Tlačítka pro zapnutí nahrávání a ukončení nahrávání jsou umístěna pod editovatelným řádkem. Jsou označena *start* a *stop*. Největší část grafického uživatelského rozhraní je tvořena tlačítky pro ovládání kalkulačky pomocí myši. Vzhledem k tomu, že tato kalkulačka neobsahuje slovo, které by fungovalo jako příkaz pro výpočet zadaného výrazu, je jediným možným způsobem vypočítání výrazu stisk tlačítka „=“. O stavu kalkulačky je uživatel informován prostřednictvím textu, který je zobrazen ve levé spodní části grafické rozhraní. Pokud je v kalkulačce vše v pořádku, je zde napsáno *„stav: OK“*. Pokud však došlo k chybě při běhu kalkulačky (může dojít k chybě při nahrávání, při rozpoznávání nebo při zpracování výrazu), tak je zde napsáno *„stav: CHYBA“*. Žádný další stav kalkulačky není uživateli zobrazován.

4.3.4 Třída `MainWindow`

Třída `MainWindow` se dědí od třídy `QMainWindow`. Stará se o ovládání grafického uživatelského rozhraní, o zpracování rozpoznaného výrazu a o výpočet výrazu. Stará se tedy o řízení běhu aplikace. Metody a členské proměnné tvořící tuto třídu jsou zobrazeny na obrázku 4.3.

Ovládáním grafického uživatelského rozhraní je myšleno propojení signálů a slotů, které mají na tyto signály reagovat. Toto propojení je realizováno v konstruktoru třídy. V kon-



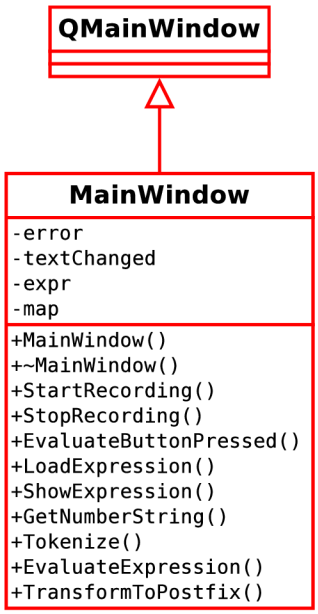
Obrázek 4.2: Grafické uživatelské rozhraní základní verze aplikace.

struktoru jsou také provedeny nezbytné inicializace instancí tříd `SOfflineSpeechRecognizerI` a `Recorder`. Sloty `StartRecording` a `StopRecording` slouží k řízení nahrávání řeči z mikrofonu. Po dokončení nahrávání předkládá třída `MainWindow` nahrávku rozpoznávači ke zpracování.

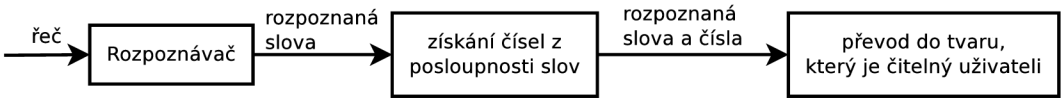
Posloupnost rozpoznaných slov je potřeba zobrazit v čitelné podobě uživateli. Rozpoznaná slova načítá ze souboru metoda `LoadExpression`. Příkladem posloupnosti načtených rozpoznaných slov budiž „STO DVACET OSM PLUS ŠEST“. Třída `MainWindow` obsahuje členskou proměnnou `map`. Tato proměnná je typu asociativní pole, jejímž klíčem je slovo rozpoznané rozpoznávačem. Pomocí klíče je z tohoto asociativního pole získána struktura obsahující řetězec určený pro zobrazení (například pro klíč „OSM“ je tento tvar „8“), typ slova (číslo, operátor nebo závorka) a v případě operátorů obsahuje informace o jeho prioritě. Zobrazení však nelze provést ihned jednoduchou substitucí rozpoznaných slov. V posloupnosti rozpoznaných slov je potřeba nalézt posloupnosti slov, které tvoří čísla. Pro grafické znázornění viz 4.4.

O zobrazení výrazu uživateli se stará metoda `ShowExpression`. Tato metoda zpracovává jednotlivá slova z rozpoznávače. U slov reprezentujících operátor nebo závorku přidá jejich textový tvar do řetězce, který bude zobrazen uživateli. U slov reprezentujících číslo je touto metodou volána metoda `GetNumberString` (této metodě se podrobněji věnuje následující odstavec). Poté, co zpracuje celou posloupnost rozpoznaných slov, je výsledný řetězec zobrazen uživateli.

Metoda `GetNumberString` hledá posloupnost slov, které tvoří číslo. Algoritmus v metodě `GetNumberString`, který hledá takovéto posloupnosti, je řízen tabulkou (příklad takové tabulky viz 4.2). Tabulka je realizována jako globální matice typu `bool`. V této tabulce je na řádku stávající číslovka, ve sloupci následující číslovka. Pokud výskyt následující číslovky za současnou číslovkou dává smysl, je stávající číslovka považována za součást čísla – například „STO DVACET OSM“ je považováno za jedno číslo, zatímco „DESET STO“ už jsou čísla dvě.



Obrázek 4.3: Přehled metod a členských proměnných třídy `MainWindow`.



Obrázek 4.4: Znázornění převodu rozpoznaných slov na výraz zobrazený uživateli.

Při uživatelově požadavku na výpočet výrazu je výraz nejdříve rozdělen na jednotlivé tokeny. Tento úkol je realizován metodou `Tokenize`, která provede lexikální analýzu zadaného výrazu. Návratová hodnota této funkce je vektor, ve kterém jsou uloženy informace o jednotlivých tokenech.

Po získání výrazu ve formě tokenů dojde k převodu z infixové notace do reverzní polské notace[5]. Tento převod zajišťuje metoda `TransformToPostfix` pomocí algoritmu uvedeného v [4]. Tento algoritmus se nazývá `Shunting-yard` a byl vytvořen Edsgerem Dijkstrou. Algoritmus je slovně popsán v tabulce 4.3.

Výraz v reverzní polské notaci lze vypočítat pomocí algoritmu, který využívá zásobníku. V programu byl použit algoritmus uvedený opět v [4]. Algoritmus pro výpočet výrazu je implementován v metodě `EvaluateExpression`. Algoritmus je slovně popsán v tabulce 4.4. Pokud při výpočtu nedošlo k chybě, tak je výsledek zobrazen uživateli. Pokud došlo k chybě, je uživateli zobrazen původní výraz a uživatel je o chybě informován prostřednictvím stavového řádku.

Pro detekci chyby slouží proměnná `error` typu bool. Proměnná `textChanged` typu bool slouží k tomu, aby mohla být detekována změna výrazu. Změnu výrazu je potřeba detekovat z toho důvodu, že při opětovném stisknutí tlačítka pro výpočet výrazu, je zobrazen původní vypočítávaný výraz.

[5]Také nazývána jako postfixová notace.

	0	**1 – 9**	**10**	**11 – 19**	**20, 30, . . . , 90**	**100**
8	F	F	F	F	F	F
20	F	T	F	F	F	F
100	F	T	T	T	T	F

Tabulka 4.2: Příklad tabulky, která řídí hledání posloupnosti čísel pro číslo 128.

1. Zpracovávej vstupní řetězec zleva doprava
2. Je-li zpracovávanou položkou operand
   přidej tento operand na konec výstupního řetězce
3. Je-li zpracovávanou položkou levá závorka
   vlož tuto závorku na vrchol zásobníku
4. Je-li zpracovávanou položkou operátor
   vlož ho na vrchol zásobníku v případě, že:
   - zásobník je prázdný
   - na vrcholu zásobníku je levá závorka
   - na vrcholu zásobníku je operátor s nižší prioritou
5. Je-li na vrcholu zásobníku operátor s vyšší nebo shodnou prioritou
   odstraň tento operátor ze zásobníku
   přidej operátor odstraněný ze zásobníku na konec výstupního řetězce
   opakuj krok 4, přestaň když se podaří vložit operátor na vrchol zásobníku
6. Je-li zpracovávanou položkou pravá závorka
   odebírej z vrcholu položky a dávej je na konec výstupního řetězce
   přestaň až narazíš na levou závorku
7. Je-li zpracovávanou položkou omezovač, pak postupně všechny odstraňuj prvky z vrcholu zásobníku a přidávej je na konec řetězce.

Tabulka 4.3: Algoritmus převodu výrazu z infixové notace do reverzní polské notace.

4.4 Zhodnocení základní verze kalkulačky

Z hlediska návrhu není v pořádku fakt, že třída `MainWindow` má na starosti dva odlišné úkoly - interakci s uživatelským rozhraním a správu (zobrazení a výpočet) výrazu. V základní verzi kalkulačky tento fakt není nějak zatěžující, ovšem při rozšíření možností kalkulačky a při rozšíření možností grafického uživatelského rozhraní by mohlo dojít k tomu, že by jedna třída `MainWindow` obsahovala příliš mnoho kódu, stala by se nepřehlednou a mohlo by dojít k potížím se složitostí programu při jeho rozšiřování.

Stěžejním problémem z uživatelského hlediska je fakt, že při rozpoznávání delšího výrazu kalkulačka tzv. *zamrzne*. Je to způsobeno tím, že rozpoznávač neběží v samostatném vlákně, ale v hlavním programu. Ačkoli se tento nedostatek může zdát pouze jako kosmetický, z uživatelského hlediska je velmi nepříjemný. V další verzi by bylo tedy jej vhodné odstranit.

Dalším velmi nepříjemným problémem z uživatelského hlediska jsou tlačítka pro zahájení nahrávání a ukončení nahrávání. Tyto tlačítka degradují celé ovládání hlasem. Tento fakt by připravil zejména tělesně handicapované uživatele o možnost ovládat tuto kalkulačku, proto by bylo dobré, aby bylo další verzi možno ovládat kompletně bez použití těchto tlačítek.

Nutnost mačkat tlačítko „=“ pro výpočet je také velmi neuspokojivá. Ze stejných dů-

1. Zpracovávej vstupní řetězec zleva doprava
2. Je-li zpracovávaným prvkem operand
   vlož ho do zásobníku
3. Je-li zpracovávaným prvkem operátor
   vyjmi ze zásobníku tolik operandů, kolikanární je operátor
   proveď danou operaci
   výsledek operace ulož na vrchol zásobníku
4. Je-li zpracovávaným prvkem omezovač
   výsledek je na vrcholu zásobníku

Tabulka 4.4: Algoritmus pro vyčíslení výrazu v reverzní polské notaci.

vodu jako v předchozím odstavci by proto byla uvítána možnost pro zadávání příkazu k výpočtu hlasem.

Bez použití myši nemá uživatel možnost opravit již zadaný výraz. Opět se jedná z hlediska uživatele o velmi nepříjemný problém. Hlasem ovládaná oprava výrazu by byla v další verzi vítanou možností.

Dalším problémem je fakt, že uživatel nemá přehled o tom, co se v kalkulačce děje. Pokud kalkulačka rozpoznává, uživatel o tom ví pouze proto, že mu kalkulačka tzv. *zamrzne* a nereaguje na jeho příkazy. Bylo by proto lepší uživatele informovat ve stavovém řádku o činnosti aplikace a to nejen v rovině „výraz vypočítán úspěšně/neúspěšně“, jak tomu je v této verzi.

Kapitola 5

Návrh rozšířené verze kalkulačky

Obsah této kapitoly je tvořen popisem požadavků na rozšířenou verzi. Po požadavcích následuje návrh rozšířené kalkulačky. V popisu návrhu jsou uvedeny změny oproti základní verzi.

5.1 Požadavky na rozšířenou verzi kalkulačky

Pokročilá verze kalkulačky má oproti základní verzi umožňovat více matematických funkcí a výrazů. V sekci 4.4 byly popsány nedostatky základní verze kalkulačky. Tyto nedostatky se snaží rozšířená verze kalkulačky odstranit. Jako rozšíření byly vybrány goniometrické funkce, cyklometrické funkce a logaritmus při základu 10. Parametry goniometrických funkcí, stejně jako návratovou hodnotu cyklometrických funkcí, by mělo jít zadávat ve stupních, radiánech nebo gradiánech. Nově přidanými operátory budou mocnina s libovolným exponentem a faktoriál. Dalším zlepšením oproti základní verzi bude možnost využívat proměnné `x`, `y` a `z`. Do těchto proměnných bude mít uživatel možnost uložit libovolnou hodnotu a v dalších výrazech s ní pak pracovat. Hodnoty těchto proměnných by měly být po vypnutí kalkulačky uchovány. Přibude možnost pracovat s desetinnými čísly. Ovládání hlasem v rozšířené verzi by mělo být zcela nezávislé na používání klávesnice nebo myši. Oprava výrazu hlasem bude probíhat po jednotlivých částech výrazu, nebude se opravovat celý výraz najednou. Aby se kalkulačka přiblížila co nejvíce jiným, běžně používaným kalkulačkám, tak by měla umět eliminovat posloupnost znamének plus a mínus. Pokud není parametr funkce uzávorkován, tak se jako parametr berou všechna čísla oddělená operátory krát, děleno, mocnina a faktoriál.

5.2 Návrh rozšířené verze kalkulačky

V rozšířené verzi je potřeba nový slovník, nejdříve je potřeba provést jeho návrh. Jak bylo zmíněno v sekci o zhodnocení základní verze kalkulačky 4.4, objektový návrh není zcela v pořádku. Objektový návrh bude proto přepracován a rozšířen vzhledem k novým požadavkům. Grafické uživatelské rozhraní bude také potřeba znova navrhnout, aby odpovídalo novým požadavkům.

5.3 Návrh slovníku

V rozšířené verzi je potřeba přidat slova, která rozpoznávač umí rozpoznat. Jedná se o slova, kterými se ovládá chod kalkulačky – slova pro zadání příkazu k výpočtu nebo k opravení výrazu, přepínání režimu počítání goniometrických funkcí - stupně/radiány/gradiány. Také je potřeba přidat slova reprezentující nové operátory, nové funkce a desetinnou tečku. Kalkulačku nebude možné vypnout pomocí hlasu. Vypnutí kalkulačky hlasem není zahrnuto z toho důvodu, že při chybném rozpoznání slova by se mohla kalkulačka vypnout, aniž by si to uživatel přál.

Ačkoli má použitý rozpoznávač vysokou úspěšnost v rozpoznávání slov, tak není vždy neomylný. Problémy mohou nastat především u vzájemně si velmi podobných slov. Typickým příkladem těchto podobných slov jsou krátká slova, která mohou být obsažena v jiných, delších slovech. Konkrétně by se mohlo jednat například o slova *„jedna“* (číslovka) a *„na“* (ve významu mocnina). Jednoduchým řešením tohoto problému by bylo přiřadit takto problematickým slovům jinou výslovnost, která by byla úplně odlišná od výslovnosti původní. To se ovšem dostává do konfliktu s požadavkem na přirozené a intuitivní ovládání kalkulačky. Při řešení tohoto problému je nutno pohlížet na tyto obě skutečnosti a najít vhodný kompromis mezi účinností rozpoznávače a uživatelským pohodlím.

5.4 Objektový návrh aplikace

Objektový návrh je přizpůsobem novým požadavkům na aplikaci a snaží se odstranit nedostatky základní verze. Oproti základní verzi je odděleno zpracování výrazu a ovládání grafického uživatelského rozhraní, obojí je realizováno ve vlastní třídě. Rozpoznávání výrazů je spouštěno v samostatném vlákně. Nově je vytvořena třída, která je určena pro ukládání proměnných, pozice a velikosti zobrazeného okna. Objektový návrh je znázorněn na obrázku 5.1.

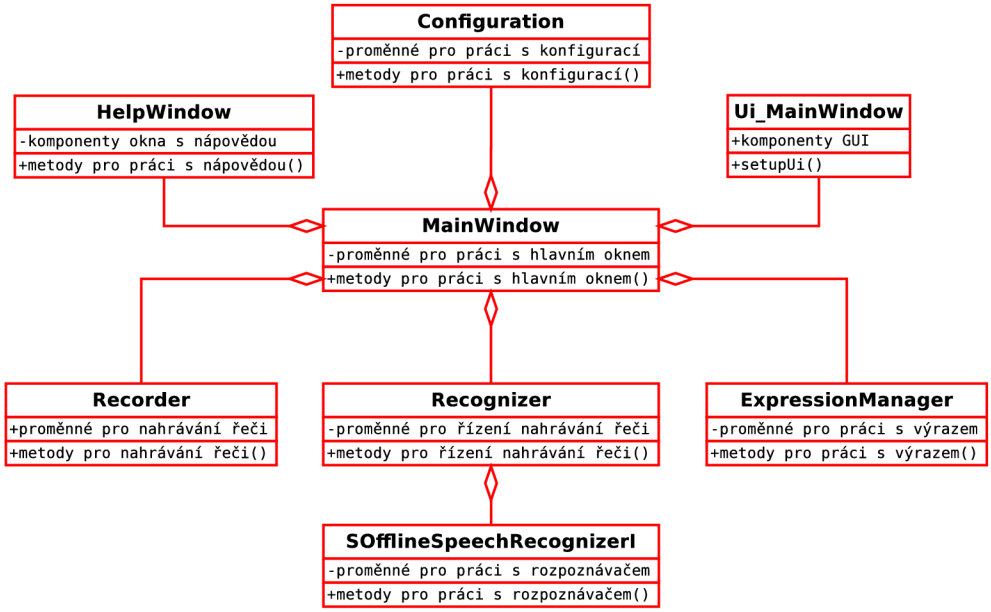
Aby šla aplikace ovládat pouze hlasem, pro nahrávání bude potřeba implementovat VAD – Voice Activity Detection. Když bude uživatel mluvit, zapne se nahrávání. Až přestane mluvit, nahrávání se vypne a nahrávka se předá rozpoznávači.

Úkoly jednotlivých tříd:

- `Recorder` – nahrávání hlasového vstupu z mikrofonu, samostatné vlákno
- `Recognizer` – ovládání rozpoznávače, samostatné vlákno
- `SOfflineSpeechRecognizerI` – rozpoznávač řeči z `BSAPI`
- `ExpressionManager` – operace s výrazem
- `Ui_MainWindow` – uživatelské rozhraní
- `HelpWindow` – dialogové okno s nápovědou
- `MainWindow` – řízení grafického uživatelského rozhraní a chodu aplikace

5.5 Návrh grafického uživatelského rozhraní

Vzhledem ke skutečnosti, že tato verze kalkulačky bude plně ovládaná hlasem, již nebude potřeba tlačítek pro zahájení a ukončení nahrávání. Bude však potřeba přidat tlačítka na



Obrázek 5.1: Objektový návrh rozšířené verze kalkulačky.

přepínání mezi stupni/radiány/gradiány, tlačítka pro proměnné, funkce, nové operátory a desetinnou tečku. Nově je přidáno menu, pomocí kterého lze zobrazit nemodální dialogové okno s nápovědou.

Kapitola 6

Implementace rozšířené verze kalkulačky

V této kapitole je popsána implementace rozšířené verze kalkulačky podle návrhu vytvořeném v kapitole 5. Je zde popsána implementace jednotlivých tříd. Na konci kapitoly je uvedeno zhodnocení této verze.

6.1 Implementační nástroje

Ačkoli knihovna `FMOD` pro nahrávání zvuku funguje dobře a jednoduše se s ní pracuje, má i své nevýhody. Touto nevýhodou je fakt, že vstupní data nezaznamenává v pravidelných intervalech. Data jsou zaznamenávána do dvou bufferů, jejichž délka se pokaždé mění. Nahrávání zvuku bylo proto implementováno pomocí knihovny `RtAudio`. Knihovna `RtAudio` volá v pravidelných intervalech funkci pro uložení zvuku. Velikost bufferu je tedy pokaždé stejná, čímž se dosáhne pohodlnější implementace detekce řeči.

Problém s třídou `RtAudio` spočívá v tom, že neumí ukládat soubor v požadovaném `.wav` formátu. Pro ukládání souboru v požadovaném `.wav` formátu je použita knihovna `libsndfile`[1].

Ostatní použité implementační nástroje zůstaly stejné jako v základní verzi kalkulačky – viz sekce 4.1.

6.2 Tvorba rozpoznávací sítě

Pro rozpoznávač se musí vytvořit nová rozpoznávací síť, pomocí které bude možné rozpoznat další potřebná slova.

6.2.1 Výslovnostní slovník

Při tvorbě slov je potřeba vzít v úvahy problém s podobnou výslovností některých slov, který je podrobněji popsaný v kapitole o návrhu (viz sekci 5.3). Za účelem plnohodnotného ovládání kalkulačky hlasem je použito celkem 64 slov. Jako nejproblémovější slova se ukázala být slova *„na“* (ve významu mocnina), *„oprava“* (slovo, které spustí opravu výrazu) a slova *„sinus“* a *„kosinus“*. Slovo *„na“* rozpoznávač často rozpoznal ve slovech *„jedenáct“*, *„dvanáct“* a podobně. Slovo *„oprava“* bylo často mylně interpretováno jako slovo *„pravá“*

[1] Přesněji řečeno její wrapper pro `C++`.

(ve významu pravá závorka). Slova *„sinus“* a *„kosinus“* se rozpoznávači pletla navzájem. Stejný problém způsobovaly slova *„radiány“* a *„gradiány“* (přepínají režim pro parametry goniometrických funkcí). Pokud uživatel dobře artikuloval, byl problém s těmito slovy minimální. Pro větší uživatelské pohodlí je tento problém ovšem vyřešen použitím jiných slov se stejným významem, které jsou dostatečně intuitivní. Slovo *„na“* je nahrazeno slovem *„exponent“*. Slovo *„oprava“* je nahrazeno slovem *„korekce“*. Slova *„sinus“* a *„kosinus“* nemají vhodné ekvivalenty, kterými by šly nahradit, proto zůstaly nezměněny. Problém s podobností slov *„radiány“* a *„gradiány“* by bylo možné řešit záměnou slova *„radiány“* za některé z jeho synonym. Pro záměnu se nabízí slovo *„grad“* nebo *„gon“*. Ani jedno z těchto slov však není dostatečně rozšířené. Slova *„radiány“* a *„gradiány“* zůstávají tedy nezměněny. Seznam všech použitých slov a jejich výslovností je podrobně zobrazen v tabulce 6.1.

Slovo	**Výslovnost**	**Slovo**	**Výslovnost**
nula	[n u l a]	sto	[s t o; s ť e; s t a; s e t]
jedna	[j e d n a]	tisíc	[ť i s í c; ť i s í c e]
dva	[d v j e; d v a]	milion	[m i l i j o n; m i l i j o n y; m i l i j o n ů]
tři	[t ř i]	celá	[c e l á]
čtyři	[č t y ř i]	plus	[p l u s]
pět	[p j e t]	mínus	[m í n u s]
šest	[š e s t]	krát	[k r á t]
sedm	[s e d m]	děleno	[ď e l e n o]
osm	[o s m]	faktoriál	[f a k t o r i á l]
devět	[d e v j e t]	exponent	[e k s p o n e n t]
deset	[d e s e t]	pravá	[p r a v á]
jedenáct	[j e d e n á s t]	levá	[l e v á]
dvanáct	[d v a n á s t]	logaritmus	[l o g a r i t m u s]
třináct	[t ř i n á s t]	sinus	[s i n u s]
čtrnáct	[č t r n á s t]	kosinus	[k o s i n u s]
patnáct	[p a t n á s t]	tangens	[t a n g e n s]
šestnáct	[š e s t n á s t]	arkus	[a r k u s]
sedmnáct	[s e d m n á s t]	stupně	[s t u p ň e]
osmnáct	[o s m n á s t]	radiány	[r a d i á n y]
devatenáct	[d e v a t e n á s t]	gradiány	[g r a d i á n y]
dvacet	[d v a c e t]	rovno	[r o v n o]
třicet	[t ř i c e t]	x	[i k s]
čtyřicet	[č t y ř i c e t]	y	[y p s i l o n]
padesát	[p a d e s á t]	z	[z e t]
šedesát	[š e d e s á t]	vypočti	[v y p o č ť i]
sedmdesát	[s e d m d e s á t]	smaž	[s m a š]
osmdesát	[o s m d e s á t]	korekce	[k o r e k c e]
devadesát	[d e v a d e s á t]	dobré	[d o b r é]

Tabulka 6.1: Podrobný seznam všech použitých slov a jejich výslovností.

6.2.2 Jazykový model

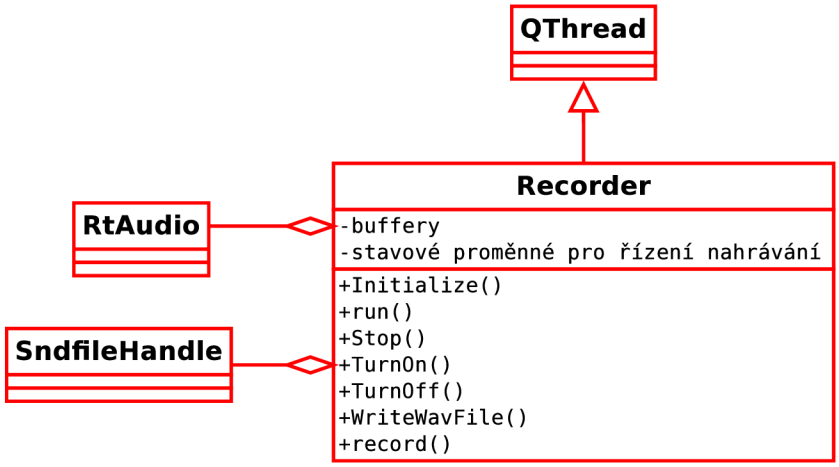
Jak již bylo napsáno v sekci 4.2.2, jazykový model určuje pravděpodobnost N-tic slov. Stejně jako jazykový model v základní verzi obsahuje jazykový model pro rozšířenou verzi pouze jednotlivá slova. Rozložení pravděpodobnosti výskytu jednotlivých slov je i zde rovnoměrné.

6.3 Implementace tříd rozšířené verze aplikace

Detekce řeči a její nahrávání je implementováno ve třídě `Recorder`. Tato třída pracuje v samostatném vlákně, aplikace tedy není nijak blokována. Pokud by třída `Recorder` nepracovala v samostatném vlákně, aplikace by nebyla použitelná. Třída `Recognizer` realizuje činnost rozpoznávače v samostatném vlákně. I v tomto případě se tak děje kvůli neblokování aplikace, avšak pouze z estetických důvodů. Práce s výrazem – tzn. zpracování výstupu rozpoznávače a výpočet výrazu – je implementována ve třídě `ExpressionManager`. Prvky grafického uživatelského rozhraní jsou umístěny ve třídě `Ui_MainWindow`. Pro zobrazení nápovědy uživateli je implementována třída `HelpWindow`. Nápověda je realizována jako nemodální dialogové okno. Řízení běhu aplikace je implementováno ve třídě `MainWindow`. Aplikace je tvořena navíc třídami `LineEdit` a `PushButton`, které nejsou v obrázku 5.1 kvůli udržení přehlednosti uvedeny. Tyto třídy reimplementují třídy pro zobrazení prvků grafického uživatelského rozhraní z Qt frameworku.

6.3.1 Třída `Recorder`

Třída `Recorder` se dědí od třídy `QThread`, což ji umožňuje spouštět v samostatném vlákně. Ve třídě `Recorder` je narozdíl od předchozí verze implementována detekce řeči. Nahrávání je realizováno za pomoci jiné knihovny. Třída je znázorněna na obrázku 6.1.



Obrázek 6.1: Třída `Recorder`.

Nahrávání pomocí knihovny `RtAudio` probíhá tak, že je v pravidelných intervalech volána funkce `record`, která zpracovává zvuk přijatý mikrofonem. Instance třídy `Recorder` zahajuje svou činnost po spuštění programu. Nahrávat se začne po detekci řeči. Detekce řeči probíhá na základě výpočtu energie vstupního signálu. Vzorec pro výpočet energie

diskrétního signálu podle [8]:

$$E = \sum_{n=0}^{N-1} |x_n|^2 \tag{6.1}$$

Pokud je energie vstupního signálu vyšší než určitá prahová hodnota, tak je nahrávání zapnuto. Ve chvíli, kdy energie vstupního signálu klesne pod prahovou hodnotu, je nahrávání vypnuto.

Empirickými pokusy byla stanovena délka signálu, jehož energie se počítá, a prahová hodnota. Délka signálu, z něhož se počítá velikost energie, byla stanovena na 250ms. Druhé mocniny vzorků, z nichž se podle vzorce 6.1 počítá druhá mocnina, jsou ukládány do posuvného okna. Vzorkovací frekvence je $8kHz$, velikost posuvného okna je tedy 2000 vzorků.

Aby se rozpoznávači ulehčil proces rozpoznávání, tak je k nahrávce předávané rozpoznávači přidána i část vstupu z mikrofonu před detekcí řeči a část vstupu z mikrofonu po detekci ticha. Délka části vstupu z mikrofonu, která je uchována před detekcí řeči, má délku 480ms. Délka části vstupu z mikrofonu, která je uchovávána po detekci ticha, má délku 320ms. Při nahrávání zvuku po detekci ticha není počítání energie signálu odstaveno, je tedy možné znova detekovat řeč. Opětovná možnost detekovat řeč byla implementována kvůli tomu, že uživatelé se při zadávání výrazu občas zamysleli a učinili tak krátkou proluku. Během této proluky bylo detekováno ticho, spuštěn rozpoznávač, a uživatel tak musel chvíli čekat než mohl zadat další část výrazu. V takovém případě se však část vstupu z mikrofonu přidávaná před nahrávku nemodifikuje. Všechny empiricky zjištěné hodnoty byly získány na základě testování uživateli.

Uložení nahrávky na disk v požadovaném formátu `.wav` $8kHz$ MONO PCM, je realizováno v metodě `WriteWavFile`. Samotné uložení probíhá pomocí instance třídy `SndfileHandle` z knihovny `libsndfile`. Zajímavostí je, že třída `SndfileHandle` neposkytuje metodu pro zavření souboru. Zavření souboru, který byl vytvořen instancí třídy `SndfileHandle`, se děje voláním destruktoru této třídy. Po uložení souboru s nahrávkou na disk je metodou `WriteWavFile` vyslán signál, který značí dokončení nahrávání.

K inicializaci rozpoznávače slouží metoda `Initialize`. Tato metoda nastavuje vstupní buffery a posuvné okno pro počítání energie vstupního signálu.

Metoda `run` je reimplementovaná metoda třídy `QThread`. Pomocí této metody je zajišťeno vytvoření nového vlákna pro nahrávání zvuku.

Metody `TurnOn` a `TurnOff` slouží k zapnutí, resp. vypnutí, zpracovávání zvukového vstupu. Zvuk se nezpracovává v době, kdy je rozpoznávač v činnosti.

Pro ukončení přijímání jakéhokoli vstupu z mikrofonu je určena metoda `Stop`. Tato metoda uvolňuje zdroje používané třídou `RtAudio`.

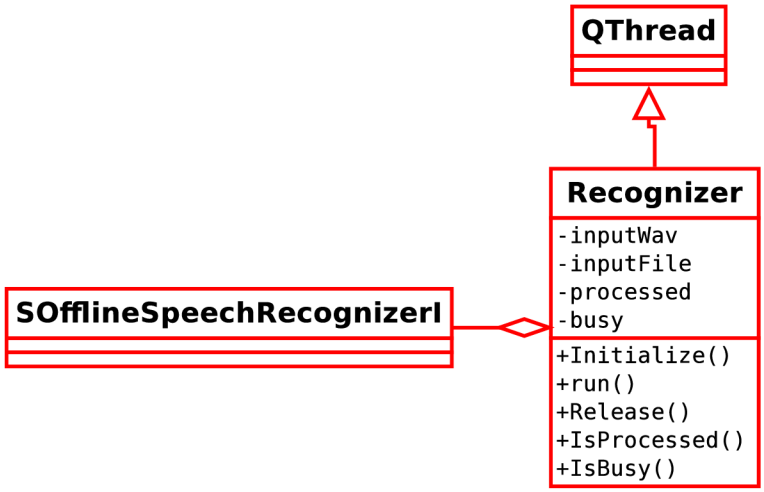
Mezi soukromými proměnnými třídy `Recorder` se nacházejí tři vektory. Vektor `preData` je určen k uchování vzorků, které se při ukládání nahrávky připojí před nahrávku. Vektor `data` obsahuje vzorky určené k uložení. Posledním použitým vektorem je vektor `slidingWindow`, pomocí kterého je realizováno posuvné okno. Tento vektor obsahuje druhé mocniny vzorků, z nichž se počítá energie signálu. Pro pamatování pozice při práci s posuvným oknem slouží proměnná `windowPos`.

Činnost třídy je řízena stavovými proměnnými. Proměnná `speaking` určuje, zda-li uživatel právě mluví. Proměnná `ready` určuje, zda-li je možno zpracovávat nahrávku z mikrofonu. Poslední stavová proměnná se nazývá `recordPrependedData` a určuje, zda-li se mají ukládat data do bufferu `preData`.

Proměnné `parameters`, `sampleRate` a `bufferFrames` jsou uchovávají parametry nahrávání. Jsou nastaveny v metodě `Initalize`. Nemohou však být lokální, protože je jich potřeba pro volání funkce `record` v metodě `run`.

6.3.2 Třída `Recognizer`

Tato třída se stará o obsluhu rozpoznávače a realizuje činnost rozpoznávače v samostatném vlákně. Třída `Recognizer` se veřejně dědí od třídy `QThread`. Je znázorněna na obrázku 6.2.



Obrázek 6.2: Třída `Recognizer`.

Metoda `Initialize` slouží k inicializaci rozpoznávače a nastavuje stavové proměnné třídy `Recognizer`. Pro uvolnění zdrojů slouží metoda `Release`.

V metodě `run` je implementován kód běžící v samostatném vlákně. Metoda nastavuje stavové proměnné a volá metodu třídy `SOfflineSpeechRecognizerI` pro rozpoznání řeči. Po dokončování řeči je vyslán signál značící konec rozpoznávání.

Stavové proměnné jsou v této třídě dvě. V proměnné `processed` je uložena informace o tom, zda-li při rozpoznání nedošlo k chybě. Proměnná `busy` indikuje zda-li rozpoznávač pracuje. Přístupové metody k těmto proměnným se nazývají `IsProcessed` a `IsBusy`.

V členských proměnných `inputWav` a `outputFile` jsou uloženy názvy souborů. Proměnná `inputWav` obsahuje název souboru s nahranou řečí, proměnná `outputFile` obsahuje název výstupního souboru, do kterého se uloží rozpoznaná slova. Názvy těchto souborů se předávají konstruktoru.

6.3.3 Třída `ExpressionManager`

Tato třída se stará jak o zobrazení výrazu uživateli, tak o výpočet výrazu. Zejména při opravě výrazu spolu mohou obě činnosti úzce souviset, nachází se proto v jedné třídě. Zobrazení i výpočet výrazu probíhají principiálně stejně jako v základní verzi.

Po dokončení rozpoznání je zavolána metoda `LoadExpression`, která načte posloupnost rozpoznaných slov. Ke zpracování posloupnosti načtených slov jsou určeny metody `ShowExpression` a `CorrectExpression`. Hlavní rozdíl mezi těmito metodami spočívá v tom, že metoda `ShowExpression` se volá pro zobrazení výrazu, zatímco metoda `CorrectExpression` se volá při opravě výrazu.

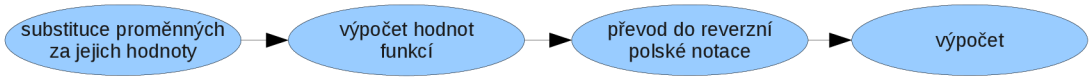
Metoda `ShowExpression` získá voláním metody `GetExpressionString` řetězec, který má zobrazit, a následně jej zobrazí. Metoda `ShowExpression` také realizuje reakci na příkazy ke smazání výrazu a vrácení se o krok nazpět.

V metodě `GetExpressionString` jsou pomocí mapovací proměnné `map` stejně jako v základní verzi získány řetězce jednotlivých rozpoznaných slov. Pro více informací o tomto postupu viz sekci 4.3.4. V této metodě jsou detekovány příkazy pro chod kalkulačky (příkazy pro výpočet, krok zpět, smazání nebo opravu výrazu, přepínání mezi stupněmi/ radiány/ gradiány). Pokud metoda narazí v posloupnosti rozpoznaných slov na číslo, je volána metoda `GetNumberString`.

Úkolem metody `GetNumberString` je najít v posloupnosti rozpoznaných slov celá nebo desetinná čísla. Pro získání celého čísla z posloupnosti rozpoznaných slov je z metody `GetNumberString` volána metoda `GetNumber`. Jako desetinné číslo jsou následně považována dvě čísla oddělená desetinnou tečkou.

O opravu výrazu se stará metoda `CorrectExpression`. Oprava výrazu probíhá po částech. Metoda `CorrectExpression` nahrazuje zvýrazněnou část výrazu novým výrazem. Novou část výrazu získá voláním metody `GetExpressionString`. V aktuálním výrazu je potřeba najít část řetězce, která se má nahradit. Z aktuálně zobrazeného výrazu se získá vektor tokenů voláním metody `Tokenize`. Pomocí vektoru tokenů se následně vypočítá počáteční pozice a délka opravované části zobrazeného výrazu. Ve chvíli, kdy je známa počáteční pozice a délka opravované části zobrazeného výrazu, je tato část smazána a na její místo je vložena nová část výrazu. Metoda `CorrectExpression` reaguje na příkazy smazání, posunutí opravy na další část výrazu a výpočet výrazu. Na příkaz smazání reaguje tak, že smaže momentálně opravovaný výraz a posune opravu na další část výrazu. Při příkazu k vypočtení výrazu je režim opravy ukončen a výraz je vypočítán.

Pro výpočet výrazu je určena metoda `EvaluateExpression`. Vstupem metody je posloupnost tokenů tvořící výraz. V této metodě jsou voláním metody `CreateInverseFunctions` v posloupnosti tokenů nalezeny inverzní goniometrické funkce. Poté jsou v posloupnosti tokenů nalezeny proměnné a jejich výskyt je nahrazen jejich číselnou hodnotou. Následně je posloupnost tokenů převedena z infixové notace do postfixové notace voláním metody `TransformToPostfix`. Výstupem metody `TransformToPostfix` posloupnost tokenů tvořená pouze čísly, operátory a závorkami. Výraz v reverzní polské notaci je vypočítán algoritmem 4.4. Postup při výpočtu je znázorněn na obrázku 6.3.



Obrázek 6.3: Znázornění zpracování výrazu před výpočtem.

Metoda zajišťující převod do reverzní polské notace `TransformToPostfix` před samotným převodem najde ve výrazu všechny funkce a vypočítá jejich hodnoty. Výpočet hodnot funkcí zajišťují metody `EvaluateLogarithm` a `EvaluateGoniometric`. Hodnoty funkcí se nahradí na místě jejich výskytu ve výrazu. Funkce dále zajišťuje splnění požadavku na nahrazování posloupností znamének plus a mínus jedním operátorem. Ve chvíli, kdy je výraz tvořen pouze čísly, operátory a závorkami, je proveden převod do reverzní polské notace. Al-

goritmus pro převod do reverzní polské notace zůstal stejný jako v základní verzi kalkulačky, viz 4.3.

V metodách pro výpočet funkcí `EvaluateLogarithm` a `EvaluateGoniometric` je nejdříve zjištěn argument funkcí. Tento argument je vypočítán voláním metody `EvaluateExpression`, čímž je umožněno rekurzivní zanořování při počítání argumentů – argumentem může být libovolný výraz včetně jiné funkce.

Argument může být buď uzávorkovaný nebo neuzávorkovaný. Uzávorkovaný argument si získají metody `EvaluateLogarithm` a `EvaluateGoniometric` samy. Pokud je argument neuzávorkovaný, volá se metoda `GetArgumentValue`. Metoda `GetArgumentValue` vrací posloupnost tokenů, které tvoří argument funkce. Jako argument funkce se považují čísla nebo funkce, které jsou odděleny operátory krát, děleno, mocnina nebo faktoriál. Například u funkce $sin\ 4+4$ je argument číslo 4, ale u funkce $sin\ 4*4$ je argument výraz $4*4$. Metoda `EvaluateGoniometric` navíc zohledňuje nastavení kalkulačky ve stupních, radiánech nebo gradiánech.

Goniometrické a cyklometrické funkce poskytované knihovnou `math.h` jako parametr přebírají (goniometrické funkce) nebo vrací (cyklometrické funkce) hodnotu v radiánech. Pro převod hodnoty z/do radiánů jsou implementovány metody `ConvertToRadians` a `ConvertFromRadians`.

Členské proměnné třídy `ExpressionManager` jsou tvořeny zejména stavovými proměnnými. Další členské proměnné jsou určeny pro uchování hodnoty proměnných, předchozího výrazu (pro uchování možnosti krok zpět) a pro opravu výrazu. Stavové proměnné jsou určeny k detekci příkazů k výpočtu, vrácení předchozího výrazu, smazání výrazu, pro řízení opravy výrazu a k detekci chyb.

6.3.4 Třída `Ui_MainWindow`

Tato třída slouží stejně jako v základní verzi k uchování prvků grafického uživatelského rozhraní. Oproti základní verzi byla upravena. Byly přidány tlačítka pro nové operátory a funkce kalkulačky. Tlačítka pro zahájení a ukončení nahrávání byla odstraněna. Informace o stavu kalkulačky byla přesunuta pod řádek pro zobrazení výrazu. Na horní stranu kalkulačky bylo přidáno menu, pomocí kterého lze aplikaci vypnout. Pomocí tohoto menu lze také vyvolat nápovědu. Grafické uživatelské rozhraní je zobrazeno na obrázku 6.4.

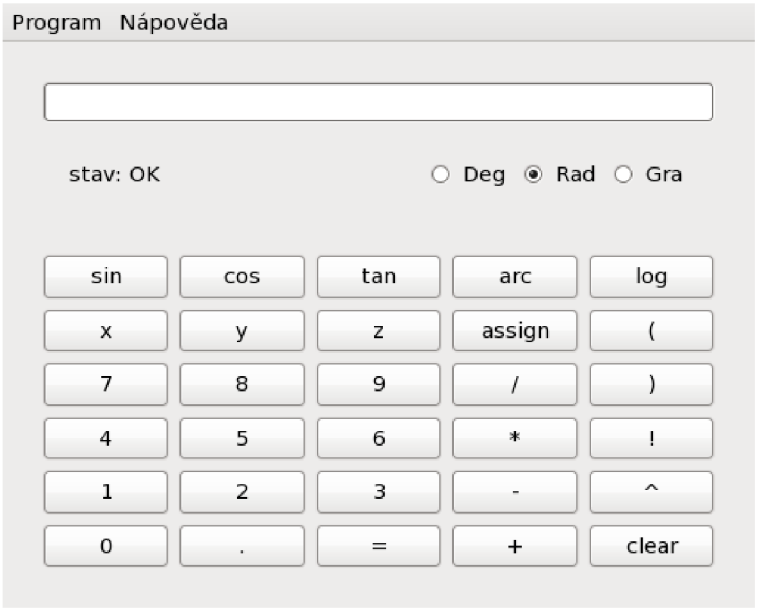
Pro potřeby grafického uživatelského rozhraní byly implementovány třídy `PushButton` a `LineEdit`. Tyto třídy se dědí od tříd z Qt frameworku. Reimplementace byla provedena kvůli potřebě specifické reakce na události. K funkcím byly přidány nové signály. Podrobněji je interakce s těmito třídami popsána v sekci 6.3.7.

6.3.5 Třída `HelpWindow`

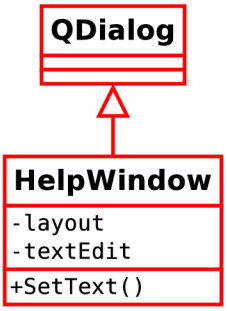
Pomocí této třídy je realizováno zobrazení nápovědy. Nápověda je zobrazena v nemodálním dialogovém okně, neblokuje tedy práci s aplikací. Text nápovědy je uložen v souboru ve formátu `html`. Grafické znázornění třídy `HelpWindow` je na obrázku 6.5.

6.3.6 Třída `Configuration`

Ukládání konfigurace programu je realizováno třídou `Configuration`. Hlavním úkolem této třídy je uchování proměnných `x`, `y` a `z`. Kromě toho třída také uchovává rozměry a pozici okna po posledním zavření. Konfigurace je načítána ze souboru ve formátu `xml`. Pokud se nepodaří otevřít konfigurační soubor, jsou nastaveny hodnoty na výchozí hodnotu. Výchozí



Obrázek 6.4: Grafické uživatelské rozhraní rozšířené verze aplikace.



Obrázek 6.5: Třída `HelpWindow`.

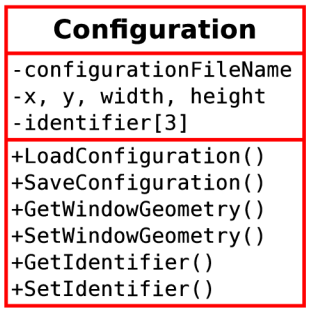
hodnota pro pozici okna je levý horní roh obrazovky. Pro všechny proměnné je výchozí hodnota 0. Třída je znázorněna na obrázku 6.6.

Metody pro načtení a uložení konfigurace se jmenují `LoadConfiguration` a `SaveConfiguration`. Členské proměnné slouží k uchování pozice hlavního okna, pozice hlavního okna a hodnot proměnných. K těmto členským proměnným jsou implementovány přístupové metody. Název souboru s konfigurací je také členská proměnná, ovšem není k ní potřeba přístupová metoda.

6.3.7 Třída `MainWindow`

Hlavním úkolem této třídy je řídit běh aplikace. V konstruktoru třídy jsou provedeny inicializace potřebné pro chod aplikace – je načtena konfigurace, zavolána metoda pro propojení signálů a slotů, je inicializován rozpoznávač a nahrávání hlasového vstupu z mikrofonu. Signály a sloty jsou propojeny v metodě `ConnectSignals`.

Po dokončení rozpoznávání výrazu je spuštěna metoda `ProcessRecord`, která volá



Obrázek 6.6: Třída `Configuration`.

příslušné metody instance třídy `ExpressionManager` pro zpracování výrazu. Rozlišuje případ, kdy je kalkulačka v režimu opravy.

Na příkaz k výpočtu reaguje metoda `EvaluateButtonPressed`. Tato metoda volá příslušné metody instance třídy `ExpressionManager`, které se starají o výpočet výrazu.

Zbylé metody ve třídě jsou implementovány jako reakce na události z uživatelského rozhraní. Jedná se o metody `ExpressionChanged`, `ButtonPressed`, `Quit` a `ShowHelp`. Metoda `ExpressionChanged` je volána v případě, že se změní zobrazený výraz. Díky ní je pak možný krok zpět. Pro ukončení programu klávesovou zkratkou nebo kliknutím v menu uživatelského rozhraní je implementována metoda `Quit`. Zobrazení nápovědy řídí metoda `ShowHelp`.

Metoda `ButtonPressed` slouží k obsluze tlačítek na grafickém uživatelském rozhraní. Aby mohla všechny tlačítka obsluhovat tato jediná metoda `ButtonPressed`, bylo nutné reimplementovat třídu `QPushButton` pro zobrazení tlačítek. Nová třída pro zobrazení tlačítek `PushButton` při kliknutí na tlačítko vysílá v signálu i název tlačítka. Díky tomu je možno v metodě `ButtonPressed` rozlišit, jaké tlačítko bylo stisknuto. Při obsluze stisknutí tlačítek je nutné brát zřetel, zda-li byla označena nějaká část výrazu – například při režimu opravy výrazu. Tuto informaci v sobě uchovává reimplementovaná třída pro zobrazení výrazu `LineEdit`.

6.4 Zhodnocení rozšířené verze kalkulačky

Vzhledem k tomu, že nahrávání i rozpoznávání je spouštěno v samostatných vláknech, tak kalkulačka tzv. *nezamrzá*. Ačkoli se jedná pouze o estetický prvek, tak byl uživateli jeho přínos hodnocen velmi pozitivně.

Odstranění tlačítek pro spuštění a ukončení nahrávání a jejich nahrazení za detekci řeči umožňuje ovládání pouze hlasem, bez toho aniž by musel uživatel použít myš nebo klávesnici. Pro kompletní ovládání hlasem byly implementovány řečové příkazy pro opravy výrazu, výpočet výrazu a smazání výrazu.

Oproti základní verzi má uživatel větší přehled o tom, co se v kalkulačce děje. Tato skutečnost je důležitá zejména při rozpoznávání výrazu. Při rozpoznávání výrazu je vypnuto nahrávání. Uživatel seznámený s ovládáním kalkulačky tedy ví, že když je na kalkulačce napsáno, že rozpoznává, tak nemá zadávat další výraz.

Kapitola 7

Testování

Kapitola se zabývá testováním uživateli. Je zde uvedena přesnost rozpoznávače při jeho testování uživateli a shrnutí dojmů uživatelů z používání hlasového ovládání.

7.1 Průběh testovaní

Aplikaci testovalo 14 uživatelů. Uživatelé měli za úkol zadat do kalkulačky několik výrazů, které pokrývaly všechna slova, která umí rozpoznávač rozpoznat. Všichni uživatelé do kalkulačky zadávali stejné výrazy. Celkový počet slov, které měl každý uživatel do kalkulačky zadat byl 109. Před samotným testováním byli uživatelé seznámeni s ovládáním této kalkulačky. Testování probíhalo v klidném prostředí s minimálním hlukem na pozadí.

7.2 Zhodnocení přesnosti rozpoznávače

Při testování přesnosti rozpoznávače byla za chybu počítána nepřítomnost vysloveného slova v rozpoznaném výrazu. Průměrná úspěšnost rozpoznání byla 85.7%. Nejvíce dělalo rozpoznávači problém slovo *„stupně“*. Ostatní chybně rozpoznaná slova byla u každého uživatele jiná. Podrobné výsledky testování jsou ukázány v příloze B. Zajímavostí je, že se testování účastnili i dva nachlazení uživatelé. Úspěšnost rozpoznávače při rozpoznávání řeči těchto dvou nachlazených uživatelů nebyla odlišná od úspěšnosti rozpoznávače při rozpoznávání řeči ostatních uživatelů.

7.3 Uživatelská odezva

Po otestování přesnosti rozpoznávače byl každý uživatel dotázán na to, co považuje za výhody a nevýhody hlasového rozhraní.

Jako výhodu uváděli uživatelé často možnost paralelní práce - jako příklad jeden z uživatelů uvedl možnost psát na papír matematický výraz a zároveň ho říkat nahlas do kalkulačky. Jako další výhodu uvedlo několik uživatelů možnost použití tohoto rozhraní handicapovanými uživateli (aniž by o této možnosti byli informováni předtím). Uživatelé, jejichž řeč kalkulačka rozpoznávala přesně, uváděli jako výhodu relativní přesnost rozpoznávání. Naopak uživatelé, jejichž řeč byla rozpoznávána s chybami, uváděli jako nevýhodu nepřesnost kalkulačky. Velká většina uživatelů následně uvedla jako nevýhodu dobu čekání na rozpoznání výrazu. Možnost ovládat kalkulačku hlasem se líbila většině uživatelů. Hlasové ovládání považoval za nepřínosné jeden uživatel.

Kapitola 8

Závěr

V rámci této bakalářské práce byla vytvořena kalkulačka, kterou lze zcela ovládat hlasem. Kalkulačka umí pracovat s desetinnými čísly v desítkové soustavě. V kalkulačce jsou implementovány operátory sčítání, odčítání, násobení, dělení, mocnina s libovolným exponentem a faktoriál. Prioritu operátorů lze zaměňovat využitím závorek. Kalkulačka umí počítat goniometrické funkce, cyklometrické funkce a logaritmus se základem deset. Argumentem funkce může být libovolný výraz. Argumenty goniometrických funkcí a návratové hodnoty cyklometrických funkcí jsou zadávány ve stupních, radiánech nebo gradiánech v závislosti na nastavení kalkulačky. Kalkulačka umožňuje používat tři proměnné.

Kromě ovládání hlasem je součástí kalkulačky také plnohodnotné grafické uživatelské rozhraní, které lze používat pomocí myši. Kalkulačku je také možné ovládat pouze pomocí klávesových zkratek zadaných z klávesnice.

Uživatelé, kteří kalkulačku zkoušeli, přijali možnost ovládat kalkulačku hlasem vesměs pozitivně.

Další vývoj kalkulačky je možné směrovat k přidání nových funkcí. Kromě přidání nových funkcí by také nebyla obtížná implementace možnosti používání jiných soustav než jen soustavy desítkové. Pro větší přehlednost by následně bylo vhodné rozšíření možnost přepínat kalkulačku mezi různými zobrazeními – například základní, pokročilé, vědecké nebo programátorské. Kalkulačka by pak poskytovala funkce v závislosti na použitém zobrazení.

Kromě přidání nových vlastností by bylo možné se zaměřit na zvýšení přesnosti rozpoznávání pomocí adaptace na konkrétního uživatele. Každý uživatel by namluvil testovací nahrávku, pomocí které by se realizovala adaptace. V kalkulačce by pak byly profily různých uživatelů.

Tato bakalářská práce nenavazuje na jiné projekty, které byly právě dokončeny.

Kalkulačka byla přijata do sborníku konference EEICT 2011 a byla prezentována v rámci soutěže Student EEICT 2011, kde se umístila na třetím místě v kategorii Grafika, multimédia, informační a počítačové systémy[1].

[1] Výsledky dostupné na http://www.feec.vutbr.cz/EEICT/2011/vysledky.htm.

Literatura

[1] Cassidy, S.: Speech Recognition. [online], Naposledy navštíveno 2. 5. 2011.
URL http://web.science.mq.edu.au/~cassidy/comp449/html/ch11s02.html

[2] Goodhead, P.: Bit-tech.net. [online], Naposledy navštíveno 3. 3. 2011.
URL http://www.bit-tech.net/news/gaming/2011/02/22/plymouth-based-developer-sets-sights-on-voi/1

[3] Hardy, H.; Strzalkowski, T.; WU, M.; aj.: Data-driven strategies for an automated dialogue system. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA: Association for Computational Linguistics, 2004.
URL http://acl.ldc.upenn.edu/P/P04/P04-1010.pdf

[4] Honzík, J. M.: *Algoritmy*. FIT VUT v Brně, 2007, 262 s., studijní opora.

[5] Kabal, P.: Audio File Format Specifications. [online], Naposledy navštíveno 4.5.2011.
URL
http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html

[6] Keogh, E.: Exact Indexing of Dynamic Time Warping. Naposledy navštíveno 2. 5. 2011.
URL http://www.vldb.org/conf/2002/S12P01.pdf

[7] Keogh, E.; Pazzani, M.: Derivative Dynamic Time Warping. In *In First SIAM International Conference on Data Mining (SDM'2001)*, 2001.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.6686&rep=rep1&type=pdf

[8] Scavone, G.: Discrete-Time Signal Metrics. [online], Naposledy navštíveno 9. 5. 2011.
URL http://www.music.mcgill.ca/~gary/306/week11/metrics.html

[9] Young, S.; Kerwhaw, D.; Odell, J.; aj.: *The HTK Book*. 1995.
URL http://nesl.ee.ucla.edu/projects/ibadge/docs/ASR/htk/htkbook.pdf

[10] Černocký, J.: *Zpracování řečových signálů*. FIT VUT v Brně, 2006, 128 s., studijní opora.

Dodatek A

Návod k použití

Pro úspěšné spuštění kalkulačky je potřeba mít platnou licenci pro rozpoznávač poskytovaný rozhraním `BSAPI`. Licence poskytuje firma Phonexia – `www.phonexia.com`.

A.1 Zadání výrazu

Zadání výrazu do kalkulačky probíhá tak, že uživatel řekne výraz složený ze slov uvedených v tabulce 6.1. Výraz je potřeba říkat se stejnou plynulostí jako při normální promluvě. Snaha vyslovit nějaké slovo pomaleji snižuje šanci na správné rozpoznání slova.

A.2 Příkazy pro ovládání kalkulačky

Kalkulačku lze ovládat slovy reprezentující příkazy pro:

- výpočet výrazu: *„vypočti“*
- smazání výrazu: *„smaž“*
- krok zpět: *„zpátky“*
- opravy výrazu: *„korekce“*
- posun na další část opravovaného výrazu *„dobré“*

A.3 Oprava výrazu

Při opravě není opravován celý výraz zaráz, ale je opravován po krátkých částech. Část výrazu, která je momentálně opravována je uživateli zvýrazněna.

Oprava výrazu je spuštěna vyslovením slova *„korekce“*. Poté se zvýrazní část výrazu, která je opravována. Od uživatele je očekáváno zadání výrazu. Výraz zadaný uživatelem poté nahradí zvýrazněnou část výrazu. Nově zadaný výraz uživatelem je poté označen jako opravovaná část.

Pro posunutí opravy na další část výrazu slouží příkaz *„dobré“*. Příkaz na smazání výrazu *„smaž“* v režimu opravy smaže zvýrazněnou část výrazu, nikoli celý výraz.

A.4 Nápověda

Nápovědu lze zobrazit buď stisknutím klávesy F1 nebo kliknutím na položku v menu Nápověda → Výslovnost. Nápověda obsahuje výslovnosti slov, tak jak byly vytvořeny ve výslovnostním slovníku.

A.5 Přiřazení hodnoty do proměnné

Hodnotu do proměnné lze přiřadit pomocí slova *„rovno“*. Tlačítko grafického uživatelského rozhraní, které plní stejný úkol jako vyslovení slova *„rovno“* má na sobě napsáno *assign*.

A.6 Tlačítka grafického uživatelského rozhraní

Jedná se o běžné tlačítka, která jsou obsažena v kalkulačkách. Jediná zvláštnost oproti běžným tlačítkům je tlačítko *clear*. Kliknutí myši na tlačítko *clear* smaže poslední znak výrazu, dvojklik smaže celý výraz.

A.7 Klávesové zkratky

Klávesové zkratky pro funkce jsou jejich první písmena – například klávesová zkratka pro sinus je klávesa „s“. Klávesová zkratka pro tlačítko *assign* je klávesa „=“. Klávesová zkratka pro tlačítko *clear* je klávesa backspace. Pro výpočet je jako klávesová zkratka použita klávesa enter.

Dodatek B

Podrobné výsledky testování

Hodnoty ve sloupci úspěšnost rozpoznání jsou zaokrouhleny na jedno desetinné místo. Celková úspěšnost rozpoznávače byla počítána vydělením počtu úspěšně rozpoznaných slov počtem celkově vyřčených slov. Celkově bylo uživateli řečeno 1526 slov. Z toho jich bylo úspěšně rozpoznáno 1308. Procentuální úspěšnost rozpoznávače tedy byla 85.7%.

Uživatel	**Úspěšnost rozpoznání v %**	**Výhoda**	**Nevýhoda**	**Hlasové ovládání**
uživatel1	92.7	pohodlné	rychlost rozpoznání	líbí
uživatel2	80.7	pohodlné	rychlost rozpoznání	líbí
uživatel3	89.0	vhodné pro handicapované	rychlost rozpoznání	líbí
uživatel4	79.8	vhodné pro handicapované	rychlost rozpoznání, nepřesné	líbí
uživatel5	91.7	přesné	rychlost rozpoznání	líbí
uživatel6	88.1	přesné	rychlost rozpoznání	líbí
uživatel7	87.2	pohodlné	rychlost rozpoznání	líbí
uživatel8	91.7	pohodlné, přesné	rychlost rozpoznání	líbí
uživatel9	86.2	pohodlné	nepřesnost	líbí
uživatel10	71.6	vhodné pro handicapované	nepřesnost	líbí
uživatel11	84.4	možnost paralelní práce	nepřesnost	líbí
uživatel12	85.3	vhodné pro handicapované	nepřesnost	líbí
uživatel13	89.0	přesné	rychlost rozpoznání	nelíbí
uživatel14	82.6	možnost paralelní práce	nepřesnost	líbí

Tabulka B.1: Podrobné výsledky testování uživateli.