Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Petr Dvořák

e-Voice Reader

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Pavel Cejnar (KTIML)
Studijní program: Informatika, Obecná informatika

2007

Rád bych poděkoval všem, kteří mi byli v průběhu psaní bakalářské práce nápomocni a bez nichž by její sepsání nebylo vůbec možné. Především zde děkuji panu Jiřímu Hanikovi za ochotu při řešení problémů s překladem a portováním TTS enginu Epos pro potřeby programu e-Voice Reader. Poděkování patří také Lukáši Machovi, který mi zapůjčil zařízení PDA. Samozřejmě děkuji i vedoucímu práce, Mgr. Pavlu Cejnarovi.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 30.05.2007 Petr Dvořák

Obsah

1 Úvod **6**

2 Epos **8**
2.1 Historie projektu Epos 8
2.2 Architektura Eposu 9
2.3 Portování Eposu pro cílovou platformu 10

3 e-Voice Reader **11**
3.1 Způsob propojení projektu s Eposem 11
3.2 Preprocessing textu pro TTS engine 12
3.3 Rozhraní pro podporu různých formátů 13
Společné požadavky na dokumenty 13
Objektový model 14
Prostý text (*.txt) 15
HTML Dokumenty (*.html) 15
Soubory elektronické pošty (*.eml) 16
Elektronické knihy založené na HTML 16
Specifikace formátu elektronických knih 17
Příklady důležitých souborů elektronických knih 20
3.4 Návrh GUI a ovládání 21
3.5 Implementace 25
Okna a dialogy 25
Komponenta HTMLView, načítání obrázků 26
Funkce pro práci s GUI 26
Třídy pro zpracování dokumentů 27
Propojení s TTS 29

4 Srovnání s podobnými produkty **31**
4.1 Microsoft Reader 31
4.2 MobiPocket Reader 33
4.3 Adobe Reader for PocketPC 2.0 34

5 Závěr **35**
5.1 Zhodnocení projektu 35
5.2 Budoucnost projektu 36

A Obsah přiloženého CD **37**

B Specifikace **38**

Literatura **42**

Název práce: e-Voice Reader
Autor: Petr Dvořák
Katedra (ústav): Katedra teoretické informatiky a matematické logiky
Vedoucí bakalářské práce: Mgr. Pavel Cejnar
e-mail vedoucího: cejnar@ktiml.mff.cuni.cz

Abstrakt: Cílem projektu je usnadnit proces komunikace mezi člověkem a mobilními technologiemi a umožnit tak, např. zrakově handicapovaným lidem, jejich bezproblémové používání. Systém ovládání vycházející z klávesnic, displejů a dotykových displejů nemusí být zcela přirozený. Nejstarší (a nespíš i nejefektivnější) formou vyjadřování je pro člověka ta verbální, tedy metoda mluveného slova. Tou však bohužel většina přístrojů nedisponuje v takové míře a kvalitě, aby se tento způsob mohl stát masově rozšířeným. Především jinak oblíbená přenosná "zařízení do kapsy", která svým aktuálně prudce vzrůstajícím potenciálem přímo vybízí k tomu, aby byla co nejvíce polidštěna a mohla se tak stát přirozenou součástí našeho života, téměř postrádají možnost sdělit uživateli informaci jinak, než přes displej, přestože hlasové ovládání se již stává standardní výbavou. Projekt e-Voice Reader je aplikace pro čtení několika častých formátů souboru s možností využití hlasové synézy a umožňuje například slabě zrakově handicapovanému uživateli PDA výkon některých běžných činností spojených s používáním zařízení PDA.

Klíčová slova: PocketPC, syntéza hlasu, čtení elektronických knih a dokumentů

Title: e-Voice Reader
Author: Petr Dvorak
Department: Department of Theoretical Computer Science and Computional Logics
Supervisor: Mgr. Pavel Cejnar
Supervisor's e-mail address: cejnar@ktiml.mff.cuni.cz

Abstract: The aim of this project is to make communication between human and mobile technologies easier and allow i.e. people with seeing difficulties to use it accordingly. Input and output methods based on keyboards, screens and touch screens might not be natural and usable in some cases. The most efficient form of communication is probably the verbal communication, in other words, the "method of spoken word". Unfortunately, most devices do not support this method in sufficient quality so that it could become widely spread. Mainly "smart devices" are very popular nowadays and they are a big challenge in making them more human-like so that they could become natural and standard part of our lives. They have almost no means of telling the information to user in other way than via (touch) screen. However, a voice control is becoming a standard. Project e-Voice Reader is an application that can perform a voice reading of some common file formats and it enables user with limited ability to see to use PDA and perform some common tasks with such a device.

Keywords: PocketPC, Text-To-Speech Synthesis, electronic books and documents

Kapitola 1

Úvod

Projekt e-Voice Reader byl vyvinut primárně jako software pro podporu uživatelů s částečným zrakovým handicapem, kteří chtějí používat zařízení typu PDA a nechtějí (nebo nepotřebují) si kupovat specializovaná zařízení pro nevidomé. Již od počátku však byl kladen důraz na to, aby nebyl zaměřen jen pro ně, ale aby jej mohli používat i ti, co chtějí pouze jednoduchou a přehlednou čtečku dokumentů. Projekt e-Voice Reader prezentuje aplikaci pro vizualizaci dokumentů nějakých předem daných formátů a umožňuje jejich předčítání hlasem, resp. separuje věcný obsah z dokumentu (ve formě prostého textu) a následně jej dokáže přečíst již existujícím hlasovým syntezátorem.

Nejdůležitějším cílem návrhu byla především jednoduchost používání tak, aby byla tato aplikace uživateli přístupná a její ovládání zvládnul i člověk s drobným zrakovým handicapem. Navigace v menu je proto dostupná i přes klávesnici a jednotlivé položky je možné nechat předčítat. V současnosti je dostatečná podpora jen pro ty nejběžnější formáty souboru (v základní verzi programu je možno předčítat "plain text", HTML dokumenty, e-mailové zprávy a speciální mixovaný formát zvukem obohacených elektronických knih založeny na HTML, viz dále).

Jedním z dodatečných cílů bylo zahrnout alespoň pro některé formáty podporu předčítání velmi velkých souborů, větších, než je v současné době na zařízeních typu PDA běžné. Při návrhu se důsledně zvažovalo, zdali se bude podpora dlouhých souborů vztahovat pro všechny formáty, nebo pouze pro některé. Nakonec zvítězila idea, že podporovány budou pouze dlouhé soubory v plain textu - bohužel, nakonec se ani u nich nedocílilo uspokojivého výsledku. Práce se velkými soubory je neúnosně pomalá, protože každý soubor je po otevření zaindexován (v lineárním čase) tak, aby bylo možné se v textu rychle navigovat. U ostatních typů souborů se od počátku od podpory velkých souborů ustoupilo, např. u souborů typu HTML by nutně muselo dojít k odstranění formátování, neboť by nebylo reálně možné udržovat v paměti všechny formátovací značky ze začátku dokumentu. Projekt má být podle filosofie autora přívětivý i k uživateli, který jej nebude využívat pro hlasovou syntézu, ale pouze pro čtení textů, proto bylo upřednostněno formátování.

Požadavek na odpovídající kvalitu hlasové syntézy si přirozeně vynutil volbu kvalitního TTS enginu. Aby bylo možné aplikaci v případě potřeby jednoduše nastavit pro syntézu textů v různých jazycích a aby bylo snadné doladit různé parametry syntézy a preprocessingu textu, musí být tento TTS engine zároveň dostatečně škálovatelný. Především požadavky na škálovatelnost splňuje TTS engine Epos, který byl nakonec použit, v plné míře. Bohužel se nakonec projevila některá omezení plynoucí ze zvolené architektury klient-server, na které Epos primárně pracuje.

Cílovou platformou, pro kterou byl projekt vyvinut, je PocketPC 2003 Second Edition. Tento operační systém je v dnešní době sice nahrazován systémem Windows Mobile 5.0, i tak jsou ale zařízení postavená na tomto OS velice rozšířená.

Aby byl zajištěn maximální výkon, projekt byl napsán v jazyce C++ jako nativní Win32 aplikace, používá tedy převážně funkce Win32 API. Pro přehrávání hudebních a zvukových souborů byla použita knihovna fmod, která je pro tento typ zařízení a pro tento účel de facto standardní. Projekt byl přeložen překladačem pro Smart Device Native Development, zahrnutým ve Visual Studiu 2005. Tento překladač je v současné době nejmodernější možný. Testování aplikace probíhalo na Microsoft Device Emulatoru V2, který je oproti své první verzi podstatně rychlejší, na silném PC skoro tak rychlý, jako pomalejší zařízení typu PDA.

Nedílnou součástí filosofie tohoto projektu je jeho otevřenost, zdrojové kódy budou tedy dostupné ke stažení zdarma na internetu tak, aby jej mohli případní zájemci přizpůsobit svým potřebám.

Kapitola 2

Epos

Předtím, než začneme hovořit o samotném projektu e-Voice Reader, považuji za nutné zmínit se v několika málo větách o projektu Epos, o jeho architektuře a také o průběhu tvorby portu pro cílovou platformu PocketPC 2003 SE. Projekt e-Voice Reader je v současné době postaven právě pro komunikaci s TTS enginem Epos, nevylučuje se ale jeho používání s jiným TTS enginem využívajícím TTSCP protokol (viz níže), a po případných úpravách ani s jakýmkoli jiným TTS enginem.

2.1 Historie projektu Epos

Projekt Epos je kompletní engine pro syntézu řeči z psaného textu. Primárně byl vyvíjený pro češtinu a slovenštinu. Slouží především pro výzkumné účely, nicméně v současnosti je, jak ostatně tvrdí sami autoři a jak se ukázalo i při psaní projektu e-Voice Reader, již příliš robustní (pro potřeby PocketPC každopádně), těžko upravitelný a jen složitě portovatelný na jiné platformy a operační systémy (výchozí OS na kterém Epos běží je UNIX/Linux).

Výhodou tohoto enginu je především to, že téměř všechna důležitá nastavení se ukládají do konfiguračních souborů (ve formátu INI) namísto toho, aby byla pevně zafixována ve zdrojovém kódu. Také jazykově závislé parametry syntézy řeči se zadávají formou pravidel uložených v textovém souboru, syntax těchto pravidel je dobře zdokumentován v [3]. Epos je tak snadno konfigurovatelný a může být - zadáním odpovídajících pravidel - potenciálně nastaven na syntézu řeči v libovolném jazyce (kromě češtiny a slovenštiny je k dispozici základní německá a francouzská sada pravidel a nastavení). Na druhou stranu se tak celý projekt stává poněkud těžkopádným - je nutné s sebou neustále uchovávat celou řadu souborů.

V současné době Epos poskytuje více algoritmů na generování řeči. Snad jenom jako informativní poznámku uvedu fakt, že název "Epos" není žádným akronymem, ale pochází vskutku od názvu literární formy. Více informací o projektu Epos naleznete v [3].

2.2 Architektura Eposu

Aby bylo možné zajistit co možná nejjednodušší propojení Eposu a aplikací, které potřebují využívat hlasovou syntézu, a aby bylo možné hlasovou syntézu provádět třeba i v síťovém prostředí, rozhodli se autoři Eposu pro architekturu typu klient-server. Jakákoli aplikace, která chce využívat Epos k předčítání textů, tedy automaticky předpokládá spuštěný Epos server. Klient se serverem komunikuje na portu 8778 prostřednictvím protokolu TTSCP (Text-to-Speech Control Protocol), který byl speciálně za tímto účelem vyvinut.

TTSCP je human- i machine-readable protokol v myšlence podobný protokolu FTP. Navržen byl pro práci nad protokolem TCP, nicméně je teoreticky schopný fungovat nad libovolným spolehlivým connection-oriented protokolem.

TTSCP rozlišuje dva typy připojení - datové (data connection) a řídící (control connection). Každé připojení je po svém vytvoření řídící a má od serveru jednoznačně přiřazený handle (řetězec alfanumerických znaků), který potom mohou ostatní připojení používat k odkazování se na dané připojení.

Řídící připojení může volat následující TTSCP příkazy (seznam není úplný, více viz [3]):

appl N - zpracuj N znaků z aktuálního streamu (stream, viz příkaz str)

intr C - přeruš provádění příkazu `appl` na připojení C

data C - připojení se změní na datové připojení řízené řídícím připojením C

delh D - ukonči datové připojení dané parametrem D

done - ukonči relaci

down - ukonči server

strm - připrav datový stream, parametrem jsou dvojtečkou oddělené "moduly" zpracovávající vstupní text. Příkazy (například příkaz `appl`) zpracovávají data skrze uvedené moduly zleva doprava. Modul nejvíce vlevo je zdroj dat (nejčastěji datové připojení), modul nejvíce vpravo bývá nejčastěji zařízení zvukové karty (*#localsound*) nebo jiné datové připojení. V případě programu e-Voice Reader jsou moduly poskládány tak, že postupně provedou processing dat od prostého textu na zvuk formátu wave (v průběhu průchodu dat moduly dojde např. ke zpracování čísel, úpravě předpon a spodoby znělosti, dořešení výjimek, ...).

Příklad použití příkazu strm:

```
strm $przmbd8h12pb:raw:rules:diphs:synth:#localsound
```

V projektu e-Voice Reader se některé výhody architektury klient-server ztrácejí, jelikož server běží na localhostu, tedy na stejném zařízení jako klient. Toto zařízení je

navíc relativně málo výkonné a využívá poměrně primitivní operační systém. Dochází tedy ke snížení výkonu aplikace kvůli režii nutné pro komunikaci serveru a klienta. Důsledkem právě uvedených skutečností je poněkud zpomalená odezva aplikace při syntéze velmi dlouhých vět.

2.3 Portování Eposu pro cílovou platformu

Pro platformu WinCE byl port Eposu již velmi hrubě připravený, nicméně nebyl udržován po několik verzí a bylo tedy nutné jej důkladně zrevidovat tak, aby ho bylo možné přeložit optimálně právě ve Visual Studiu 2005. To se nakonec ukázalo být velmi obtížné.

V projektu musely být (v tomto pořadí) jednak přidány aktualizované a nové zdrojové soubory tak, aby port pro PocketPC obsahoval tytéž soubory jako udržované porty pro jiné platformy (vycházel jsem z projektu pro stolní Windows), dále byly opraveny některé více či méně závažné chyby (konflikty typu *const char∗* vs. *char∗*) a odstraněna drobná varování (*signed* vs. *unsigned*), následně odstraněny některé zastaralé a dnes již zavržené konstrukce jazyka C a v neposlední řadě i přepnuta direktiva /FORCE tak, aby se při linkování ignorovaly duplicitní definice funkcí a unresolved symboly (bez tohoto kroku se mi projekt nepodařilo přeložit a nebylo časově a ani technicky možné zkoumat a předělávat strukturu hlavičkových souborů tak, aby se projekt slinkoval).

Hlavním problémem při portování Eposu byl však fakt, že původně byl Epos navržen pro prostředí příkazové řádky (jednalo se o Win32 konzolovou aplikaci, běžící jako služba na pozadí). Toto prostředí však není zařízením typu PDA vůbec vlastní. Celý projekt proto bylo nutno přepsat na (nativní) Win32 aplikaci. Krom toho se vyskytl ten problém, že knihovna *stdlib.h* dostupná pro PocketPC neobsahuje implementaci pro některé základní funkce nezbytné pro správnou činnost serveru. Bylo proto nutné si je "doimplementovat" vytvořením souboru stdlib.cpp, a to tím způsobem, že dané funkce (např. funkce open, write, ...) pouze vrátily nějakou pevnou číselnou hodnotu. Je jasné, že tyto funkce dělají (resp. měly by dělat) mnohem více, nicméně odpovídající implementace *stdlib.cpp* není pro PocketPC v současnosti dostupná, vlastní implementace všech potřebných funkcí je v rozumném čase nereálná a v konečném důsledku se ukázalo, že není s trochou šikovnosti ani potřeba ji realizovat.

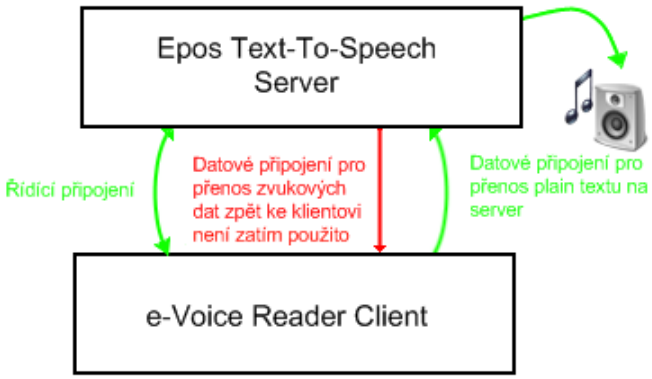
Nakonec zde stojí za to zmínit, že zařízení typu PDA stále ještě nedisponují výkonem dostatečným pro práci s kvalitně zabarvenými hlasy, jejichž datové soubory mívají i více než 15MB. Bylo proto potřeba se omezit na jednodušší, více minimalistické a v důsledku samozřejmě i méně kvalitní hlasy. Základním hlasem eposu je hlas "theimer". Ten sice není zdaleka optimální, pro většinu aplikací je však zcela dostačující, je hlavně rychlý a je mu vcelku dobře rozumět.

Kapitola 3

e-Voice Reader

3.1 Způsob propojení projektu s Eposem

V ranných fázích tvorby projektu se zvažovalo, jestli se bude využívat (doporučené) architektury klient-server, nebo zda se bude projekt Epos přepisovat tak, aby jej bylo možné využít jen jako (například) knihovnu, která se následně pouze přilinkuje k projektu e-Voice Reader. Přestože druhý přístup by byl z hlediska výsledného výkonu aplikace znatelně výhodnější, byl zvolen postup první, tedy byla zvolena architektura typu klient-server a následná komunikace klienta se serverem pomocí protokolu TTSCP. Hlavním faktorem při rozhodování byla časová náročnost nastudování zdrojových kódů Eposu a také složitost jejich případné modifikace. Projekt e-Voice Reader tedy tvoří klienta pro Epos server a s tímto serverem komunikuje prostřednictvím protokolu TTSCP. Program se k serveru připojuje třemi konexemi, jedna je řídící a dvě jsou datové (stačila by jen jedna datová konexe, druhou zdůvodňuji dále v kapitole Implementace a v současné verzi se nepoužívá). Schéma, podle kterého probíhá připojení klienta e-Voice Reader k serveru Epos je zachyceno na následujícím obrázku (obrázek 1). Je zde také znázorněno, že zvuk na zvukové zařízení vlastně posílá server.



Obr. 1 Schéma propojení klienta se serverem

3.2 Preprocessing textu pro TTS engine

Každý dokument musí projít předtím, než je přečten, jistým preprocessingem. Ten probíhá jak na klientovi, tak na serveru.

Nejdůležitější část preprocessingu textu probíhající na klientovi je rozdělení souboru na jednotlivé věty a zaindexování jejich začátků. Při uvažování o tom, jak posílat jednotlivé části dokumentu TTS serveru, se totiž došlo k závěru, že není možné předložit TTS enginu celý dokument najednou. Hlasová syntéza delších úseků je totiž časově náročná a i pro relativně krátký dokument by mohla trvat celé minuty. Naopak - kratší úseky, než je jedna věta, např. slova, nejsou vhodné už jen z toho důvodu, že většina TTS enginů používá pravidla syntézy řeči aplikovaná na celé věty. Volbou úseků odpovídajících větám tedy docílíme kvalitnějšího zvukového výstupu.

V projektu e-Voice Reader jsou tedy TTS enginu posílány jednotlivé věty. Aby je bylo možné detekovat co možná nejpřesněji, probíhá odstranění známých zkratek a jejich nahrazení plným slovním vyjádřením (zkratka "tj." se nahradí slovy "to jest", "Bc." se nahradí za "bakalář", ...) - tím se zamezí předčasnému rozdělení věty kvůli prezenci tečky za zkratkou a explicitně se určí jak se má zkratka přečíst. U aplikace je k tomuto účelu přiložen jednoduchý slovníček překladů zkratek na jejich nezkrácenou formu, který se v programu načte a potom se s ním v případě potřeby pracuje. Dále se v rámci hledání konců vět ignoruje tečka následující bezprostředně za číslovkou, aby nedocházelo k rozdělení věty po řadové číslovce. Tento krok se provádí na úkor možnosti přečtení dvou vět v případě, že číslovka byla opravdu na konci nějaké věty (takových vět je ale málo). Dále je ignorována tečka nacházející se uprostřed webové adresy začínající na "http://" nebo "www." - po nalezení těchto řetězců se považuje za URL následující text až po první bílý znak. Poznamenejme, že rozdělení textu na jednotlivé věty není nutné jen kvůli využívání hlasové syntézy. Zároveň slouží při synchronizaci audio obsahu s textem elektronických knih (o tom dále).

Operace provedené na klientovi jsou doplněny processingem textu na serveru. Dojde zde například k převedení čísel na jejich odpovídající slovní vyjádření. Epos umí čísla zpracovat velice kvalitně a navíc se snaží (ve výchozím nastavení, konfigurovatelnost pravidel Eposu byla již několikrát zmíněna výše), detekovat římské číslice. Zároveň zde dojde v relativně ranné fázi k odstranění spodoby znělosti, dále k převodu speciálních znaků na jejich přepis (znak "/" se přepíše na "lomeno"), a k slovníkovému nahrazení některých (například i anglických) slov, jejich výslovností. Další pravidla aplikovaná na text poslaný serveru se dají dohledat v human-readable souborech $*.dic$, umístěných v adresáři *$EPOS$/cfg/lng/czech*. Později se na serveru zároveň aplikují pravidla prozodie. Epos umí měnit frekvenci, délku trvání a intenzitu patřičných částí věty (např. poslední slabiky ve větě, ve slově, ...) tak, jak je to obvyklé v příslušném jazyce.

Při uvažování o tom, jak bude nejlépe přistupovat k souboru, aniž by jej bylo nutné uchovávat celý načtený do nějaké speciální struktury, si musíme uvědomit několik věcí. Tou hlavní je specifikum zařízení typu PDA, která nemají (alespoň ne v standardních provedeních) pevný disk založený na magnetických kotoučích a samostatnou rychlou RAM paměť, ale jen flash paměť, která slouží jak pro úložiště dat a nainstalovaných programů, tak pro paměť používanou pro běžící procesy. Velikost obou dvou úseků paměti se navíc dynamicky mění podle aktuální potřeby - v případě, že máme nainstalováno mnoho aplikací, zbude nám méně místa pro běžící programy a naopak. Důležité je, že při uvažování o časové složitosti jednotlivých IO operací nemusíme rozlišovat mezi "přístupem na disk" a "přístupem do paměti", neboť jsou obě tyto operace realizovány nad stejným typem hardware a poznamenejme ještě, že poměrně rychle. Nevyplácí se nám proto alokovat na "paměti" zbytečně velký kus místa a přesouvat do něj části souboru, protože si tím z hlediska rychlosti a v důsledku i paměťové náročnosti příliš nepomůžeme. Pro přistupování k souboru byl proto použit následující přístup. Pro čtení souboru se používají standardní STL streamy, které tvoří abstrakci nad souborem, takže se souborem se pracuje tak, jako by se jednalo o proud znaků. Po otevření souboru proběhne zaindexování začátků jednotlivých vět v celém dokumentu (v lineárním čase), abychom se po dokumentu mohli rychle pohybovat. Při pozdějším čtení souboru se ve streamu vyhledá dle indexu patřičná věta, načte se a odešle na Epos server. Koncept je sám o sobě relativně dobrý, např. je možno hned po načtení dokumentu zvolit přechod na konkrétní větu a rychle přejít na záložku. Počáteční zaindexování vět je však více časově náročné, než se zpočátku předpokládalo, a proto není reálně možné s velkými soubory rozumně pracovat. Řešením časové náročnosti by bylo neprovádět počáteční indexaci. Zde ale narážíme na problém detekce začátku vět. V současné době parsování probíhá odpředu dozadu. Není proto možné se bez indexu posunout na předchozí větu. Bylo zváženo, že by parsing mohl probíhat obousměrně, tedy že by se začátky vět mohly detekovat i odzadu. Průchod textu odpředu a odzadu však - vzhledem k aplikovaným pravidlům - nemusí být v korespondenci. Například ve spojení "17. listopadu" se při přečtení tečky věta při dopředném průchodu neukončí (tečka je za číslovkou), při zpětném ano. Toto by bylo možné řešit "okénkem", tento způsob by však byl velice pracný a v důsledku by se navigace po čteném textu zpomalila.

3.3 Rozhraní pro podporu různých formátů

V současné době jsou podporovány nejběžnější formáty souborů, které by mohl uživatel chtít zpočátku číst. Mezi činnosti běžně prováděné na PocketPC patří čtení e-mailů, prohlížení webových stránek, čtení elektronických knih a prohlížení textových dokumentů. Odpovídající formáty tedy musely být v programu podporovány. Podpora většiny formátů je řešena jen částečně. Cílem návrhu nebylo implementovat plnou podporu pro malý počet standardů (reálně pro jeden) tak, jako to dělá většina současných čteček, ale umožnit předčítat co nejvíce různých typů souboru a ukázat tak možnost existence jedné čtečky pro více různých formátů souboru. Plná implementace všech výše uvedených formátů je v rozumném čase nerealizovatelná.

Společné požadavky na dokumenty

Společným požadavkem na dokumenty, které umí program zpracovávat je některé z českých kódování (Latin2 nebo Windows-1250). Zároveň se vyžaduje ukončení řádky standardně dle Windows CR+LF, protože použití STL streamů v implementaci pro PocketPC toto nutně vyžaduje. V případě, že by řádky byly ukončeny jinak, dojde k chybnému zaindexování začátků vět. Více o tomto problému najdete v sekci Implementace.

Objektový model

Programátorské rozhraní pro parsování dokumentů bylo navrženo s ohledem na budoucí rozšiřitelnost aplikace.

Základním stavebním kamenem je třída TParser, která sama umí kompletně pracovat s dokumenty v prostém textu. Obsahuje zároveň všechny (dle potřeby virtuální) metody nutné pro zpracování libovolného typu souboru a pro navigaci v dokumentu. Zahrnuje tedy například podporu indexace obsahu dokumentu po větách (kvůli rychlé navigaci v dokumentu), datové struktury a metody pro ukládání záložek a poznámek, virtuální metodu pro získání jedné věty z dokumentu, metody pro pohyb po indexech, ... Tyto zmíněné funkcionality musí mít totiž všechny parsery bez ohledu na typ dokumentů.

Od třídy TParser je poděděna třída THTMLParser, která umí (navíc oproti výše zmíněnému) například pracovat s HTML entitami a značkami - toho se používá v metodě při získávání věty ze streamu, zde se pochopitelně oproti stejné metodě třídy TParser musí navíc odfiltrovat HTML značky a nahradit HTML entity.

Od třídy TParser může být potenciálně poděděna libovolná další třída pro teoreticky libovolný typ souboru. Mohlo by tak být například řešené i zpracování textových a HTML e-mailů: jednoduše by se podědily dvě třídy pro parsing mailů typu text/plain a pro parsing mailů typu text/html. Protože však bylo uváženo, že třída pro práci s e-maily bude v případě budoucího rozšíření programu vyžadovat více, než jen parsování textu a indexaci vět (bude muset například umět komunikovat s POP3 serverem nebo spravovat přílohy, případně maily odesílat přes SMTP server) byla tato třída postavena mimo dědičnou hierarchii třídy TParser. Protože již máme parsery pro prostý text i HTML, bylo v současné verzi zvoleno "zapouzdření" tříd odvozených od TParser do třídy TEmailParser (tato třída tedy obsahuje ukazatel na TParser a dle potřeby se alokuje buď parser HTML mailů nebo mailů v prostém textu). V tomto přístupu se vyskytl ten problém, že nelze přímo manipulovat s protected členy třídy TParser (TEmailParser není od TParser poděděna), bylo tedy nutno obalit metody třídy TParser metodami ve třídě TEmailParser.

Pro práci s elektronickými knihami byla napsána samostatná třída (TNavigationParser). Ta v sobě sdružuje seznam kapitol (dokáže jej vytáhnout z HTML souboru

vhodného formátu), seznam hudby doprovázející čtení (čte se z CSV souboru) a informace o knize. O formátu elektronických knih je pojednáno dále. Pro zpracování samotného souboru čtené kapitoly se - vzhledem k formátu (viz níže) - používá třída THTMLParser.

Shrneme si teď typy souborů, které program podporuje a definujeme si formát elektronických knih programu e-Voice Reader.

Prostý text (*.txt)

Základním formátem souboru, který vyžaduje jen minimální úroveň předzpracování, je prostý text. Jelikož se jedná o typ souboru, který de facto postrádá jakoukoli zásadní informaci o formátování, bylo možno pro něj zvážit podporu velmi dlouhých souborů. Vzhledem k výše uvedeným faktům však tento cíl nebyl naplněn v dostatečně uspokojivé podobě. Uživatel je na příliš velký soubor upozorněn a má možnost zrušit jeho otevření.

HTML Dokumenty (*.html)

Vzhledem k tomu, že HTML je velice populární formát pro tvorbu různých strukturovaných dokumentů, např. článků, a že formátování elektronické pošty pomocí HTML je také poměrně časté, jeho podpora byla jedním z důležitých požadavků, které byly na aplikaci již od počátku kladeny. Nebylo však možné a především výhodné (kvůli rychlosti aplikace) integrovat do aplikace celý robustní HTML parser, neboť to jediné, co potřebujeme s HTML dokumenty v podstatě provádět je odfiltrovat text od HTML značek a celý dokument nějak zobrazit.

K účelu zobrazování HTML dokumentu byl použit ovládací prvek dostupný v základní výbavě PocketPC SDK (je přímo ve Windows CE 3.0 API) - HTMLView (htmlview.lib) [2, HTML Control API] - který na stolních Windows nemá patřičný ekvivalent. Tato komponenta v sobě integruje takřka celou funkčnost Pocket Internet Exploreru, podporuje dokonce částečně i kaskádové styly (CSS). Vizualizace HTML dokumentů je tedy jednak na velice vysoké úrovni, a navíc bylo díky chování komponenty možné využívat pohyb po kotvách ve stránce. To zásadním způsobem usnadnilo synchronizaci hlasu s předčítanou částí dokumentu.

Co se parsování samotného týče, program obsahuje jen velmi základní prostředky pro chápání sémantiky dokumentu. U HTML značky je možno dotázat se na její jméno. To je totiž potřeba znát kvůli tomu, aby byla ukončena logická věta například na konci nadpisů, odstavců, apod., a to i v případě, že není ukončena tečkou, vykřičníkem nebo otazníkem. Toto chování (ukončování věty po uzavření některých tagů) se totiž ukázalo jako více logické a bylo dosahováno lepších výsledků, než při důsledném ukončení vět tečkou, otazníkem nebo vykřičníkem. Jelikož se zatím nevyskytla potřeba pracovat s atributy jednotlivých tagů, tuto funkčnost bude nutné v případě potřeby do parseru doimplementovat. Program rozpoznává HTML entity,

uchovává si ve speciálním souboru tabulku překladů entit na odpovídající znaky a číselné entity interpretuje přímo na odpovídající znaky.

Program předpokládá validní dokumenty, u nevalidních je správné zpracování nejisté.

Soubory elektronické pošty (*.eml)

Podpora elektronické pošty je na úrovni dostatečné pro čtení většiny česky psaných textových e-mailů (případně formátovaných pomocí HTML), které jsou uloženy na zařízení ve formátu souboru elektronické pošty (*.eml). Program tedy neobsahuje žádnou podporu pro stahování zpráv např. přes POP3 server. Aby mohl uživatel zprávy číst, musí být tyto staženy pomocí poštovního klienta, případně synchronizovány pomocí aplikace ActiveSync. Složku, kam se e-mailové zprávy stahují je možno nastavit. Poštovní klient tak může být použit jen jako prostředek pro stahování zpráv, aplikace e-Voice Reader může posloužit jako jednoduchá čtečka a vizualizér zprávy. V současné verzi programu se nepracuje s e-maily obsahujícími přílohy (tedy s typem "multipart/..."), musí se tedy jednat o jednodušší zprávu v prostém textu ("text/plain") nebo o zprávu formátovanou HTML ("text/html"). Také s e-maily kódovanými v quoted-printable si program v současné době neporadí (ale rozpozná jej a uživatele upozorní na nepodporovaný formát). Dekódování tohoto formátu bude však potřeba pravděpodobně doimplementovat, protože tento formát je v českých podmínkách relativně častý. Nejspíš nejrychlejší přístup by byl založený na uložení dekódovaného souboru do dočasného souboru a následná práce s tímto dočasným souborem.

Elektronické knihy založené na HTML

Aby mohl být e-Voice Reader použit i pro čtení elektronických knih, bylo nutné zvolit formát, ve kterém elektronické knihy budou reprezentovány. Po dlouhém zvažování bylo rozhodnuto, že program e-Voice Reader bude pracovat s vlastním mixovaným formátem. Existující formáty pro elektronické knihy (např. formát DAISY) měly totiž obrácenou logiku, než by bylo pro použití v programu vhodné (a žádané). Ani jeden z nich totiž neuměl synchronizovat audio s textem. Všechny synchronizovaly text s audiem. Čtení takovýchto knih potom bylo spíše poslechem přednesu uloženého ve formátu mp3 a text byl záležitostí do jisté míry redundantní. Jeho zobrazování se řídilo podle audia. Tento přístup kupříkladu zabraňuje tomu, aby čtenář četl knihu svým tempem, tedy tempem, které mu vyhovuje.

Formát elektronických knih programu e-Voice Reader je navržen tak, aby měl jednoduchou sémantiku dostatečnou pro reprezentaci knih, a bylo dbáno na to, aby byl vhodný jak pro prohlížení čtečkami, tak pro publikaci na webu. Zároveň bylo uváženo, že bude vhodné zvolit nějaký human-readable formát, aby bylo možné dokument v případě potřeby upravit. Bylo proto rozhodnuto založit elektronické knihy právě na HTML.

Na následujících pár stránkách provedeme specifikaci struktury souborů elektronických knih programu e-Voice Reader.

Specifikace formátu elektronických knih

Každá kniha je umístěna v samostatném adresáři ve složce Books v kořenovém adresáři programu e-Voice Reader (toto nastavení je možné změnit v config.txt, viz dále). Název složky může, ale nemusí být ve spojitosti s titulem knihy. Pro použití knihy v programu e-Voice Reader se doporučuje ponechat název složky stejný s titulem knihy, ten se totiž použije na první úrovni navigace. Obecně to ale nutné není. Ve složce knihy je soubor index.html, který obsahuje základní informace o knize a uspořádaný seznam jednotlivých kapitol, jedná se tedy o jakýsi rozcestník. Tento soubor je povinně souborem ve formátu XHTML 1.0 Transitional a tělo dokumentu je omezené pouze na elementy `<h1>`, `<h2>`, `<ol>`, `<li>` a `<a>`.

Struktura souboru index.html

Hlavička souboru index.html (element `<head>`) obsahuje vždy povinně potomka `<title>`, který obsahuje název titulu, a několik povinných potomků `<meta>` obsahujících dodatečné metainformace o knize. Syntaxe elementu `<meta>` je tradiční jako v HTML, tedy: `<meta name="jmeno" content="obsah" />`.

Povinné `<meta>` elementy jsou uvedeny v následujícím seznamu (tučně je vždy uveden atribut `name`, následuje ho popis obsahu atributu `content`). Seznam vychází z doporučení DCMI (Dublin Core Metadata Initiative) pro publikování elektronických knih.

dc:title - obsahuje název publikace

dc:creator - obsahuje jméno (nebo jména) a autorů, kteří mají nárok na intelektuální vlastnictví dokumentu

dc:date - obsahuje datum vytvoření publikace ve formátu "yyyy-mm-dd"

dc:description - popis dokumentu, typicky obsahuje abstrakt nebo výtah dokumentu

dc:subject - téma, o kterém dokument pojednává, typicky klíčová slova a klíčové fráze

dc:language - jazyk, ve kterém je dokument publikován

dc:source - odkaz na zdroje, ze kterých dokument čerpá (tento ¡meta¿ element se může opakovat několikrát)

dc:rights - prohlášení o autorských právech

dc:format - je vždy "text-html"

Mohou být zároveň doplněny jiné `<meta>` elementy sloužící například k doplnění informací o HTML dokumentu jako takovém, volitelně je možné použít element `<link>` pro připojení externího stylu.

Tělo souboru index.html (element `<body>`) obsahuje povinně právě jeden nadpis první úrovně (`<h1>`), ve kterém je uveden název publikace. Dále je zde právě jeden nadpis druhé úrovně (`<h2>`), obsahující jméno autora. Následuje uspořádaný seznam s odkazy do příslušných kapitol knihy (`<ol>`, `<li>`, `<a>`), odkazy povinně obsahují relativní cestu k jednotlivým kapitolám vůči adresáři knihy. Kapitoly knihy jsou uloženy povinně v podadresáři adresáře příslušné knihy pojmenovaném nejlépe "chapter" nebo "kapitola".

Kapitoly knihy

Jednotlivé kapitoly jsou povinně uloženy v podadresáři adresáře knihy, který se dle doporučení jmenuje buď "chapter" nebo "kapitola" tak, aby byla viditelná sémantika struktury knihy například při publikaci na webu. Názvy souborů samotných kapitol jsou opět libovolné, doporučuje se jim dát stejný nebo podobný název, jako má příslušná kapitola, opět kvůli sémantice dokumentů.

Soubory kapitol jsou validní XHTML 1.0 Transitional soubory, v těle povinně redukované na elementy nadpisů (`<h1>` - `<h6>`), odstavců (`<p>`), seznamů (`<ul>`, `<ol>` a `<li>`), formátovacích značek částí textu (`<span>`) a obrázků (`<img>`). Takovýto formát dokumentu má jasnou a čistou sémantiku, může být obohacen kaskádovým stylem a zajišťuje možnost jednoduchého předčítání dokumentu. Absence prostředku pro zachycení tabulek je cílená, tabulky totiž nejsou přirozenou součástí knih a není v podstatě možné je rozumně předčítat. Většina jevů se dá pro potřebu knihy sumarizovat do seznamu (např. pořadí běžeckého závodu) místo do tabulek. Tabulky navíc odpradávna vybízí k tomu, aby sloužily k účelům, ke kterým nemají, tedy k vizuálnímu formátování dokumentu.

Doplnění knihy o audio obsah

Jednotlivé kapitoly mohou být volitelně podbarveny zvukem či hudbou ve formátu mp3, wav, mod, it a xm. Příslušné skladby jsou povinně uloženy v podadresáři knihy, který se doporučeně jmenuje "audio". V případě, že chceme podbarvit knihu hudbou, umístíme v kořenovém adresáři knihy soubor audio.txt. Přítomnost či absence tohoto souboru rozhoduje o tom, je-li kniha podbarvena zvukem. Soubor "audio.txt" je klasický soubor formátu CSV (Character Separated Values), ve kterém jsou hodnoty oddělené středníkem. Řádky tohoto souboru mají následující strukturu:

```
STR[kapitola];INT[věta];STR[audio];BOOL[TTS]
```

Jednotlivé sloupce označují po řadě relativní cestu k souboru s kapitolou knihy (cesta je vůči adresáři knihy), pořadí věty v té kapitole, ke které se daný zvuk pojí

(zde pozor! záleží na tom, jak jsou věty indexovány, proto se audio obsah nedoporučuje u knih obsahující konfliktní významy interpunkčních znamének), relativní cestu vůči audio souboru a hodnotu 0 nebo 1 dle toho, zda se má pro tuto větu použít TTS. Poslední sloupec je nepovinný - implicitně se doplňuje na 1 (true).

S trochou píle je tedy možné kompletně spojit čtený text s přehráváním audia a například měnit náladu knihy zvolením vhodného zvukového doprovodu pro jistou pasáž knihy, nebo dokonce přehrávat jednotlivé věty ve formátu mp3, případně takto přihrávat jen citace...

Příklady důležitých souborů elektronických knih

Příklad souboru index.html

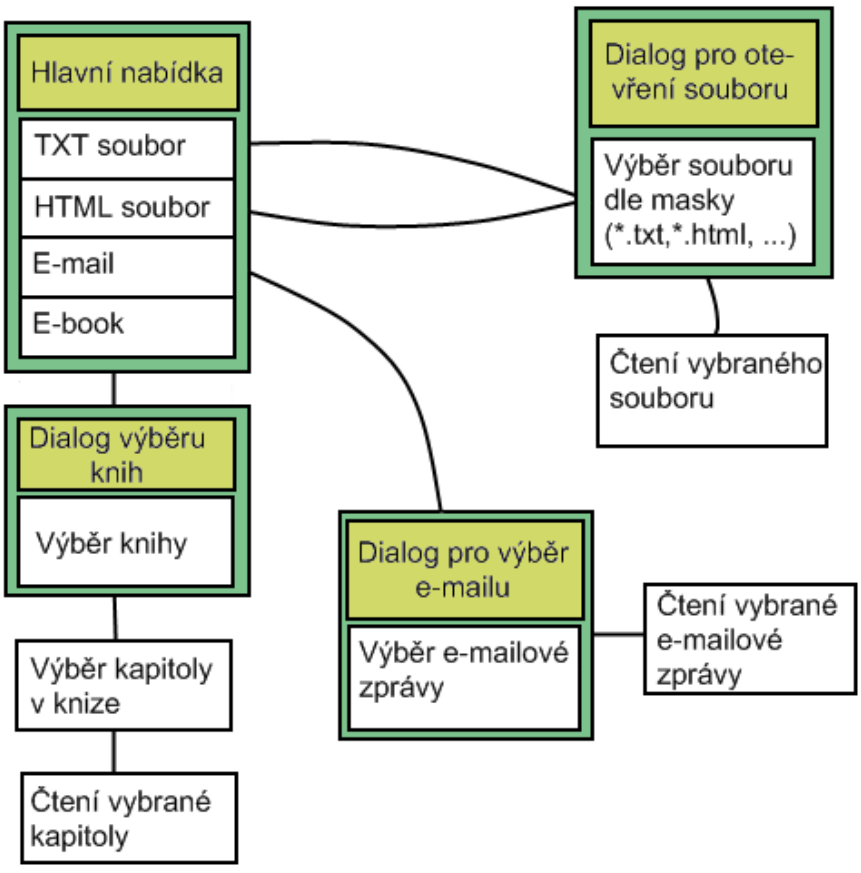
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs">
  <head>
    <title>Titulek dokumentu</title>
    <meta name="dc:title" content="Titulek dokumentu" />
    <meta name="dc:creator" content="Autor dokumentu" />
    <meta name="dc:date" content="2004-12-20"/>
    <meta name="dc:description" content="Místo pro popis dokumentu" />
    <meta name="dc:subject" content="klíčová slova, klíčové fráze" />
    <meta name="dc:language" content="cs" />
    <meta name="dc:source" content="Odkaz na první zdroj" />
    <meta name="dc:source" content="Odkaz na druhý zdroj" />
    <meta name="dc:rights" content="Dokument je volně šiřitelný" />
    <meta name="dc:format" content="text/html" />
    <meta http-equiv="Content-Type" content="text/html;charset=iso-8859-2"/>
    <link type="text/css" rel="stylesheet" href="style/base.css" />
  </head>
  <body>
    <h1>Titulek dokumentu</h1>
    <h2>Autor dokumentu</h2>
    Seznam kapitol
    <ol>
      <li><a href="chapter/chapter1.html">Kapitola 1</a></li>
      <li><a href="chapter/chapter2.html">Kapitola 2</a></li>
      <li><a href="chapter/chapter3.html">Kapitola 3</a></li>
    </ol>
  </body>
</html>
```

Příklad souboru sync.csv:

```
chapters/chapter1.html;0;audio0.mp3
chapters/chapter1.html;4;audio1.mod;0
chapters/chapter2.html;0;audio0.mp3
chapters/chapter3.html;9;audio2.mod;0
```

3.4 Návrh GUI a ovládání

Aplikace byla navržena především s důrazem na jednoduchost. Proto bylo uživatelské rozhraní napsáno v souladu se současnými trendy pro návrh GUI. Celá aplikace je postavena tak, aby si na ni uživatel snadno a rychle zvykl a aby neměl pokud možno žádné problémy s jejím používáním. Všechny důležité činnosti související s používáním programu jsou namapovány na klávesové zkratky tak, aby bylo možné program ovládat jen s použitím klávesnice (připojené například přes Bluetooth). Aplikaci je ale samozřejmě možné pohodlně ovládat i stylusem a navigačním křížem, neboť bylo uváženo, že tento způsob ovládání bude využívat pravděpodobně většina jejich uživatelů. Po spuštění aplikace se zobrazí velice jednoduché okno, ve kterém uživatel vybere typ dokumentu. Toto okno tvoří hlavní rozcestník aplikace, jakýsi centrální bod. Z něj se uživatel snadno dostane do dialogu pro výběr konkrétního dokumentu zvoleného typu. Po výběru dokumentů se zobrazí dialog pro předčítání vybraného dokumentu. Navigace v programu byla tedy navržena i s důrazem na logiku: schéma obsahuje jeden centrální bod (hlavní obrazovku) a z něj vedoucí větve odpovídající jednotlivým typům dokumentu. Přehledně je schéma zachyceno na obrázku 2.



Obr. 2 Schéma navigace v programu

Vizualizace čteného dokumentu byla také poměrně důležitá, a i když se nejednalo o klíčovou záležitost při návrhu aplikace, výsledek je velice dobrý. Program se tak stává velice použitelnou čtečkou elektronických knih a dokumentů bez použití TTS.

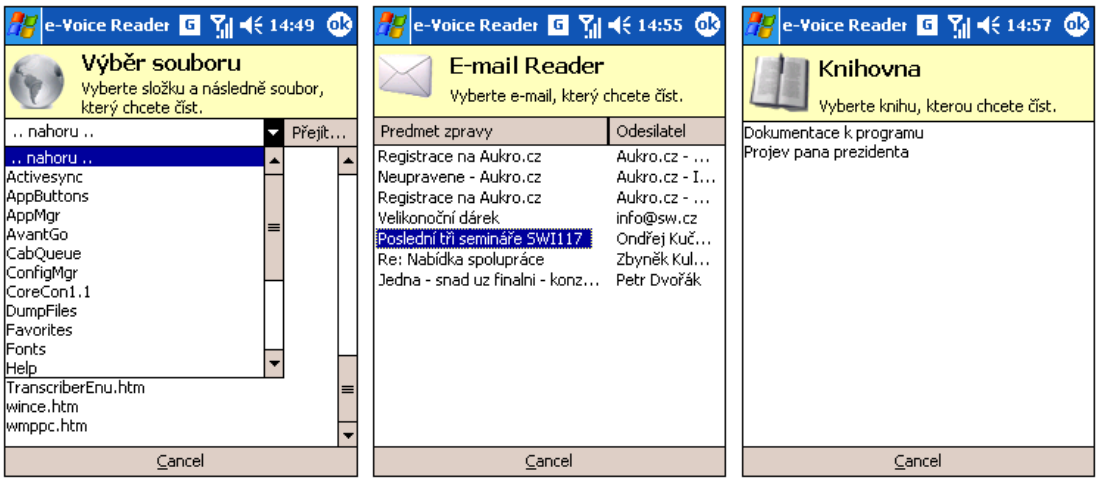
Při čtení se uživatel může po čteném textu pohybovat po celých větách (šipkami vpravo/vlevo). K tomuto kompromisu v navigaci bylo přistoupeno z několika důvodů. Především to bylo proto, že nebylo nutné pohybovat se po jednotlivých slovech, nebo dokonce po znacích. Věta čtená například od prostředního slova nemá totiž velký smysl (o čtení slova zprostředka a případných změnách významu z toho plynoucích ani nemluvě), TTS engine optimalizuje syntézu řeči pro celé věty, vizualizace dokumentu není pohybem po celých větách nějak výrazně omezena, bylo by nutno zahrnout dodatečné struktury pro indexaci odpovídajících částí dokumentů a v neposlední řadě - bylo by nutno přidat ovládací prvky do GUI aplikace. Poslední jmenovaný fakt by zajisté omezil jednoduchost a přehlednost GUI. Při pohybu po čteném textu je možno přejít na začátek (Home) a na konec (End) dokumentu, pohyb po čteném textu se dá urychlit skokem po třech větách (šipka nahoru/dolů). K přečtení aktuálně zvoleného ovládacího prvku slouží ve většině případů klávesová zkratka Alt+R. Ovládání všech dialogů pro čtení dokumentů bylo co nejvíce sjednoceno, aby si uživatel nemusel uvědomovat, že pracuje s konkrétním typem souboru a aby se ovládání programu rychleji naučil.

Zkusíme si nyní ve stručnosti popsat vzhled aplikace a ukázat detaily jednotlivých dialogů. Pokud jste již četli uživatelskou dokumentaci (najdete ji na přiloženém CD) nebo dokonce používali program, můžete zbytek kapitoly přeskočit. Slouží jen jako náhled na zpracování aplikace.

Po spuštění programu se zobrazí hlavní okno aplikace. To slouží k výběru typu čteného dokumentu. Je možné zvolit soubory v prostém textu, HTML dokumenty, e-mailové zprávy a elektronické knihy založené na HTML.

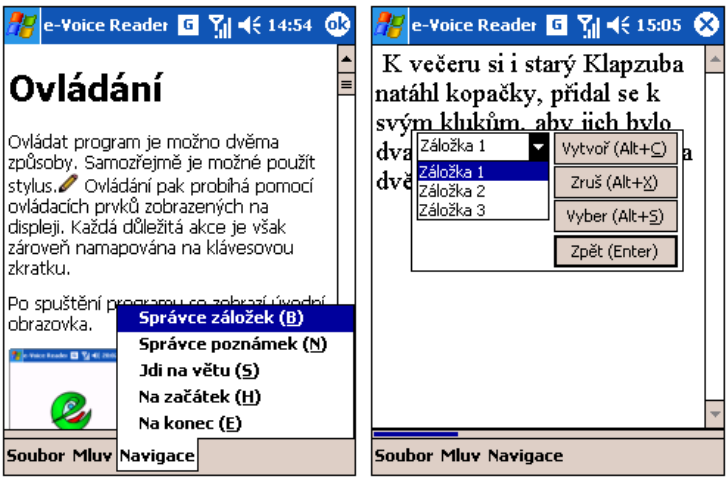
Obr. 3a Úvodní okno programu, výběr typu dokumentu

Po vybrání příslušného typu dokumentu se dostáváme k dialogu, pomocí kterého vybereme konkrétní dokument daného typu. Vzhled tohoto dialogu se liší podle zvoleného typu dokumentu. V programu e-Voice Reader jsou tak k dispozici tři typy dialogů pro výběr dokumentu - dialog pro výběr souboru, dialog pro výběr e-mailu a dialog pro volbu elektronické knihy.



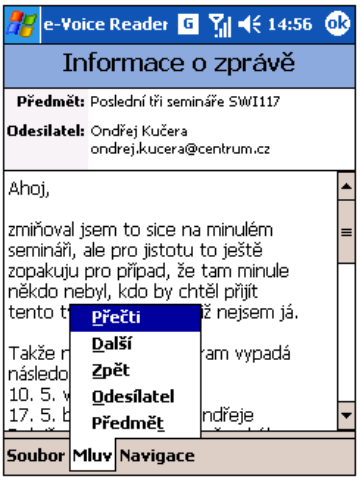
Obr. 3b Typy oken pro výběr dokumentu: soubor, e-mail a e-book

Po výběru souboru z prvního typu dialogu se otevře okno, ve kterém probíhá samotné zobrazování dokumentu a které umožňuje dokument předčítat po větách hlasem. Toto okno může být (dle typu otevřeného dokumentu) dvojího typu. Pakliže je dokument formátován pomocí HTML, zobrazí v okně formátovaný text. V případě plain textu (a potenciálně libovolného dokumentu, a to třeba včetně HTML) se v dialogu zobrazuje jediná, a to právě čtená, věta. Ke každému dokumentu je možno vytvořit libovolně mnoho záložek (ukládají se do externího souboru s příponou .bookmark) a libovolné množství poznámek (přípona .note).



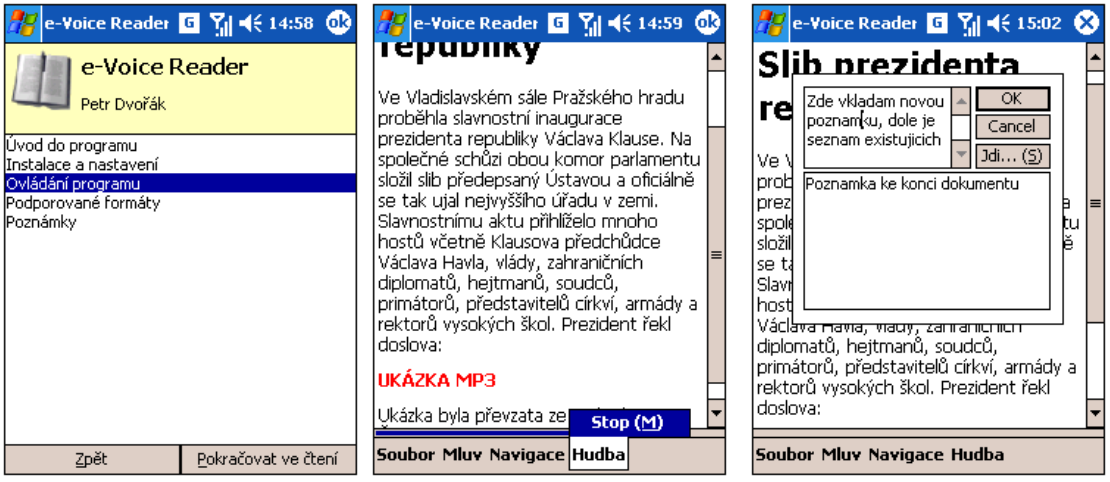
Obr. 3c Předčítání souboru, v prvním případě HTML, v druhém prostý text

Po výběru e-mailu se zobrazí okno, ve kterém je základní info z hlavičky e-mailu a text e-mailu. Ten je opět možno nechat předčítat po větách. Opět lze ke každému dokumentu vytvořit libovolně mnoho záložek a poznámek.



Obr. 3d Dialog pro čtení e-mailů

Při čtení elektronických knih se nejprve zobrazí výběr kapitol, odtud je možné buď pokračovat výběrem kapitoly, nebo pokračovat ve čtení knihy z místa, kde byla kniha naposledy opuštěna (informace o místě posledního opuštění knihy obsahuje soubor master.bookmark v kořenovém adresáři knihy). V každé kapitole můžeme navíc vytvářet záložky a poznámky.



Obr. 3e Dialogy pro výběr kapitol a pro čtení elektronických knih

3.5 Implementace

Při samotné implementaci programu se musely důkladně zvážit operace a funkcionalita, kterou musí program poskytovat a jednotlivá rozhodnutí bylo nutno přizpůsobit těmto požadavkům.

Uživatel musí mít možnost zvolit, jaký typ souboru a jaký soubor chce otevřít, musí být schopen do souboru vkládat záložky a poznámky, musí mít možnost se po dokumentu rozumně pohybovat, zároveň je nutno velké množství informací posílat na TTS server. V této kapitole, která je součástí programátorské dokumentace, si shrneme, jak jsou jednotlivé výše popsané funkce realizovány ve zdrojových kódech programu.

Okna a dialogy

Aplikace obsahuje jedno hlavní okno a z něj jsou potom vyvolávány modální dialogy. Všechny "call-back" funkce jsou k dohledání v souboru "main.cpp", stejně jako funkce pro registraci hlavního okna. Jsou pojmenovány rozumně tak, aby se dalo už z názvu usoudit, k čemu jednotlivé funkce slouží. Podobně bylo dbáno na pojmenovávání dialogů a dialogových prvků.

Po spuštění aplikace se zobrazí hlavní okno sloužící k výběru typu dokumentu. To obsahuje genericky vytvořené tlačítko a listbox. Zároveň je zde pomocí volně stažitelné knihovny VOImage (a také knihovny VOString - ta je nutná právě pro používání VOImage) od společnosti Virtual Office System a knihovny imgdecmp.dll zobrazený obrázek - logo programu. To se překresluje při zpracování zprávy `WM_PAINT` [1, str. 57, 64]. Aby bylo možné zachytávat odpovídající zprávy a aby je "nepohlcoval" listbox, je vynucován fokus na hlavní okno aplikace při zpracování zprávy `WM_KILLFOCUS` [1, str. 183]. Toto je nutné kvůli předčítání textu v reakci na konkrétní zprávu, kterou okno obdrží. Veškeré relevantní zprávy, například zprávy pro listování v listboxu a pro potvrzení tlačítkem jsou tak posílány jednotlivým ovládacím prvkům přímo hlavním oknem. Tento postup byl zvolen proto, že je podstatně jednodušší než například subclassing prvků, který by byl na druhou stranu rozhodně systémovějším řešením. Po stisku tlačítka "OK" nebo stisku Enter (při stisku Enter se tlačítku pošle `WM_CLICK`) [**?**, str. 183] se vyvolá odpovídající modální dialog. Call-back funkce hlavního okna se jmenuje `WindowProcMain`.

Každý z dialogů, na které se dostaneme z hlavního okna, obsahuje v horní části custom komponentu, která slouží jako obal pro komponentu HTMLView. Při psaní programu je nutné mít na paměti, že do dialogu vkládáme vlastně "okno v okně". Komponenta umožňuje rychle vytvořit pěkně vyhlížející hlavičku dialogu. Za zvážení by však stálo použití kreslené hlavičky pomocí základních API funkcí a knihovny VOImage, neboť použití komponenty HTMLView je zde "kanon na vrabce" (ale je to jednoduché). V každém dialogu druhé úrovně je dále ListBox (resp. ListView v dialogu pro výběr mailů) a několik tlačítek sloužících k navigaci. Každý z dialogů

má názorně pojmenovanou call-back funkci: `OpenDlgProc` pro výběr obecného souboru, `OpenEmlProc` pro výběr e-mailu a `OpenLibraryProc` pro výběr elektronické knihy. Každý dialog reaguje na příkazy (zpráva `WM_COMMAND`) [1, str. 383] IDOK a IDCANCEL, zároveň je reagováno na změnu fokusu prvků (kvůli čtečce). Z dialogů druhé úrovně se dostaneme většinou rovnou ke čtenému dokumentu. Výjimku tvoří dialog pro výběr kapitoly knihy, který se vyvolá z knihovny. Tento dialog je velice podobný dialogům pro výběr dokumentu, jeho call-back funkce se jmenuje `ChapterSelectDlgProc`.

Dialogy pro čtení dokumentů měly požadavek na co nejvíce unifikované ovládání, reaguje se tedy na velice podobné zprávy. Klávesy jsou většinou namapovány na konkrétní akce tak, že se při zpracování zprávy `WM_KEYUP` [1, str. 182] (tato se ukázala jako vhodnější než `WM_KEYDOWN`) pošle zpráva `WM_COMMAND` příslušnému prvku v menu (to se do dialogu vloží rychle zavoláním funkce `CreateMyCommandBar(HWND, int)`, která je na začátku souboru "main.cpp"). Tento přístup je velice jednoduchý na implementaci. Pomocí klávesových zkratek je například možné "spustit z menu" dialog pro přechod na konkrétní větu, pro správu záložek a pro správu poznámek. Pokud odpovídající prvek v menu není (např. stisk klávesy dolů může místo položky v menu obhospodařit navigační kříž), provede se přímo dané operaci odpovídající kód. Call-back funkce jednotlivých dialogů se jmenují názorně: `EmlReadDlgProc`, `PlainTXTReadDlgProc`, `HTMLReadDlgProc` a `HTMLBookReadDlgProc`.

Komponenta HTMLView, načítání obrázků

Zmiňme se jen krátce o použití komponenty HTMLView. Ta byla v programu použita nejen pro hlavičky dialogů, ale i pro vizualizaci HTML souborů a mailů ve formátu HTML. Tato komponenta se musí inicializovat - to se provádí ve funkci InitInstance(). Tato komponenta je potom naplněna obsahem HTML souboru (posloupnost zpráv `DTM_CLEAR`, `DTM_ADDTEXTW` a `DTM_ENDOFSOURCE`). Co se týče zobrazování obrázků pomocí této komponenty (např. v HTML souboru), bylo manuálně realizováno při zpracování zprávy `WM_NOTIFY`, konkrétněji `NM_INLINE_IMAGE`. Komponenta totiž vyžaduje absolutní cesty k souborům obrázků, proto bylo ručně dopsáno ošetření relativní cesty, v současné verzi pouze směrem nahoru. Nefungují tedy relativní cesty obsahující "..", místo očekávaného obrázku se objeví tradiční ikonka pro chybně načtený obrázek. Dle doporučení na MSDN byly i k účelu načtení obrázků použity knihovny VOImage [2, Developing in C++ with the HTML Viewer Control]. Zde právě bylo nutné uvědomit si, že komponentu HTMLView neumisťujeme do generického okna (vytvořeného například funkcí CreateWindow), ale do dialogu, bylo proto nutné volat před návratem z call-back procedury `SetWindowLong(hDlg, DWL_MSGRESULT, TRUE)` [2, SetWindowLong] a následně vrátit hodnotu TRUE k notifikaci zapouzdřeného okna HTMLView o korektním zpracování načtení obrázku (nastavením návratové hodnoty zprávy). Tento krok zmiňuji především proto, že je velice těžko k dohledání.

Funkce pro práci s GUI

Pro manipulaci s GUI byla napsána sada funkcí deklarovaných v "simplemanip.h". Jedná se vlastně o funkce tvořící obal nad WinAPI a propojující WinAPI s STL. Tyto funkce především umí naplnit ListBox, ListView a ComboBox z vektoru řetězců, resp. z vektoru obsahujících třídu TTranslation (třída pro uchování dvou řetězců - zde se používá trochu v nesouladu s názvem pro naplnění ListView předmětem a odesílatelem zprávy v dialogu pro výběr e-mailu). Dále je v tomto hlavičkovém souboru funkce pro zobrazení resp. skrytí SIP, pro test toho, jestli je zvolená cesta adresář nebo pro nastavení základních vlastností dialogu (aplikace standardního vzhledu pomocí `SHInitDialog`).

Třídy pro zpracování dokumentů

K zpracovávání dokumentů dochází prostřednictvím tříd deklarovaných v souboru "parsers.h". Nejdůležitější je bezpochyby třída TParser. Od ní je odvozena třída THTMLParser. Třída TEmailParser zapouzdřuje třídu TParser, resp. obsahuje ukazatel na TParser a dle potřeby se alokuje buď TParser (text/plain) nebo THTMLParser (text/html). Třída obsahuje dodatečné informace o zprávě. Bokem stojí třída TNavigationParser pro práci s informacemi o knihách a s jednotlivými kapitolami.

V souboru "main.cpp" jsou za účelem parsingu deklarovány tři globální proměnné. Volba globálních proměnných v někom může vyvolávat hrůzu a zajisté se nejedná o nejelegantnější řešení. Na druhou stranu se jedná o jednoduchý a poměrně rychlý způsob, jak sdílet data mezi více dialogy. Dialogy jsou navíc vždy modální, ke konfliktům by tedy nemělo při běžné práci s programem docházet. Proměnná CurrentParser je typu ukazatel na TParser a slouží pro práci s textovými a HTML soubory. EmailParser je typu ukazatel na TEmailParser a slouží k práci s e-mailovou zprávou. Proměnná Chapters je typu ukazatel na TNavigationParser se používá pro práci s otevřenou knihou. Všechny tyto proměnné se dle potřeby alokují v dialozích vyvolaných z hlavní obrazovky před vyvoláním dialogu se samotným dokumentem (resp. výběrem kapitol knihy) a dealokují se po návratu z tohoto vyvolaného modálního dialogu.

Popíšeme si v rychlosti to, jak je navržena třída TParser. Třída obsahuje struktury, které jsou společné pro všechny typy parseru, aby bylo možné od ní jednoduše odvozovat další třídy. Musí tedy splňovat většinu požadavků na funkcionalitu nutnou pro práci s dokumenty, jako je mimo jiné navigace v dokumentu. Jsou zde tedy i struktury pro ukládání záložek a poznámek, dále slovníček překladů zkratek a seznam indexů začátků vět. Tyto struktury jsou jednoduše STL vektory. Třída TParser obsahuje konstruktor, umožňující její inicializaci ze souboru (wstring) a ze slovníčku pojmů, tedy např. zkratek (vektor obsahující TTranslation). Tomuto konstruktoru je ekvivalentní volání virtuálních metod AcceptDictionary a Load, ty je nutno volat při použití výchozího konstruktoru (před indexací vět). Dále třída obsahuje důležitou virtuální metodu GetSentence, která slouží k získání wstringu, ob-

sahujícího jednu větu načtenou z aktuální pozice v STL streamu (tedy z aktuálního indexu). Ve volání této metody je použita některá ze tříd poděděná od abstraktní třídy TSentence, která se umí inicializovat ze streamu seznamem slov jedné věty a vrátit obsah věty v podobě wstringu. Dále je zde virtuální metoda GetSegment sloužící k vrácení nezpracovaného obsahu souboru od i-tého do i+1. indexu (resp. do konce souboru). Toho se využívá k posílání obsahu HTML dokumentu po částech odpovídajících větám komponentě HTMLView. Přitom před každou takto odeslanou část se vloží kotva (aby bylo možné synchronizovat čtený text a se zvukem). Třída TParser zároveň obsahuje metody pro posun ukazatele na daný index. Tyto metody jsou volány při zpracování příslušné zprávy v proceduře dialogu v případě, že se uživatel aplikace nějakým způsobem naviguje po čteném textu. Dále jsou zde metody pro uložení a načtení poznámek a záložek do externího souboru. Název těchto souborů je v rámci třídy TParser a od ní nějakým způsobem odvozených tříd libovolný, ale v programu e-Voice Reader se používá název shodný s názvem čteného souboru a příponami "note" resp. "bookmark". Informace o místě posledního opuštění knihy obsahuje soubor master.bookmark v kořenovém adresáři knihy. Změnu chování ukládání záložek a poznámek je potřeba provést v dialozích pro výběr souboru, e-mailu a kapitoly knihy (načítání i ukládání záložek totiž probíhá před otevřením a po uzavření souboru nebo kapitoly). Nakonec zde nemůžeme nezmínit, že indexace dokumentu probíhá při volání virtuální metody Reindex(). Tato metoda v lineárním čase projde otevřený dokument a do seznamu indexů uloží polohu začátků vět v rámci dokumentu.

Pro podporu elektronických knih se vyskytla potřeba silnějšího parsingu HTML, než byla schopna zajistit třída THTMLParser. Ta totiž dokáže maximálně rozpoznat název elementů a odfiltrovat text, ale neumí zjistit obsah atributů elementu. Ten je však potřeba znát při parsování rozcestníku HTML knih (zajímá nás atribut `href` elementu `<a>`). Třída TNavigationParser byla navržena právě za tímto účelem. Umí rozparsovat soubor typu HTML a vybrat z něho nadpis 1. a 2. úrovně a seznam odkazů, které ze souboru vedou. Slouží tak dobře právě pro parsování a zpracování HTML rozcestníku elektronických knih. Zároveň umožňuje se na získané informace jednoduše dotazovat. Můžeme se tak v programu dotázat na autora, titul a kapitoly dané knihy. Toho se využívá v dialogu pro výběr kapitol. Dialog pro výběr knihy informaci z tohoto parseru nevyužívá. Prvotní navigace využívá pouze názvu adresáře knihy. To je totiž rychlejší, než prohledávat všechny adresáře v adresáři knih a v nich případně parsovat soubor index.html. Třída TNavigationParser zároveň obsahuje strukturu pro ukládání audio informace ke knize. Konkrétně se jedná o vektor obsahující položky typu AudioInfo - jednoduchá a průhledná třída obsahující soubor kapitoly, číslo věty a soubor audia. Pro samotné přehrávání audia bylo použito API fmod, je tedy potřeba mít knihovnu fmodce.lib a také to znamená, že do adresáře Windows se při instalaci nakopíruje soubor fmodce.dll.

Je také potřeba zmínit, že všechny dokumenty, se kterými se v programu pracuje, musejí mít v aktuální implementaci řádky ukončené posloupností znaku CR+LF. Dokumenty s jinak ukončenými řádky totiž nedokáží STL streamy správně zpracovávat.

Posloupnost volání metod seekg a tellg třídy fstream se totiž v souborech s řádkami ukončenými jinak chová velice nežádoucím způsobem: bude-li IFSTREAM instance třídy wifstream, volání `IFSTREAM.seekg(IFSTREAM.tellg(), ios_base::beg)` posune ukazatel do streamu pro následující volání metody `get`, přestože by se logicky posunout neměl a je nutno podotknout, že v dokumentaci těchto metod (na MSDN) o tom zmínka není. Pro podporu i jinak ukončených řádků by se nejspíš musely streamy předělat na binární streamy (to by znamenalo četné drobné zásahy v celé aplikaci).

Propojení s TTS

Abstrakci nad sockety Eposu (a tak i nad samotnou syntézou řeči) zajišťuje třída Talker deklarovaná v "talkingclass.h". Sockety si tedy při používání hlasové syntézy není nutné explicitně uvědomovat. Třída Talker tvoří nad základními příkazy a často prováděnými posloupnostmi příkazů protokolu TTSCP jednoduchý objektový obal, je tedy možné jednoduše volat metody této třídy. Celá aplikace obsahuje právě jednoho "mluvčího" (globální proměnná "Jimmy" v "main.cpp"), který obhospodařuje v podstatě celou komunikaci klienta se serverem, a tak fakticky řídí předčítání textu uživateli.

Třída Talker umožňuje například poslat serveru žádost o syntézu řeči z textu a následné přehrání výstupu na zvukovém zařízení. Dále tato třída umožňuje nastavit požadované kódování textu pro komunikaci s Epos serverem, nastavit stream pro syntézu řeči z prostého textu (viz předchozí kapitola), připojit se k serveru a nebo se od něj odpojit. Dodatečně byla do rozhraní této obalující třídy přidána možnost připravit si do zásoby buffer obsahující zvuk ve formátu waveform (proto to druhé datové připojení, slouží k přenosu zvukových dat od serveru ke klientovi) a možnost přehrát zvuk uložený v takovémto bufferu na klientovi voláním WinAPI funkcí pro přehrávání zvuků - waveOut... Tyto poslední dvě jmenované možnosti implementované uvnitř třídy mají za úkol právě umožnit kompletní zpracování přehrávání generované řeči na klientovi a dát tak základ pro budoucí rozšiřování aplikace a vylepšování její funkčnosti. Portování aplikace pro WinCE a (pravděpodobně především) absence kompletní knihovny stdlib.h mají totiž jeden velice nepříjemný důsledek. Server zašle klientovi oznámení o ukončení syntézy řeči (odpověď "200 - OK") ještě v době, kdy se zvuk vzniklý touto syntézou přehrává (na serveru). Přesněji téměř okamžitě po odeslání zvuku na zvukové zařízení, a nikoliv až po dokončení přehrávání, jak je tomu u serveru, přeloženém pro stolní Windows. Při přehrávání zvuku na serveru (v našem případě na *localhostu*) tedy není možné zjistit, kdy bylo dokončeno reálné přehrávání zvuku. Kvůli tomuto nedostatku je tedy v současné verzi projektu e-Voice Readeru nemožné nechat přehrát celý dokument naráz, neboť syntéza probíhá po větách a další věta by se odeslala na zvukové zařízení ještě v době, kdy předchozí věta nebyla dopovězena. Je tedy nutné si explicitně požádat (pomocí menu nebo ovládacího kříže) o přečtení další věty.

Musíme zde také poznamenat, že přenos zvukových dat ze serveru ke klientovi trvá poměrně dlouho. Možnost přenést data ke klientovi a přehrát je prostřednictvím klienta proto nebyla v současné verzi v projektu použita, přestože tento přístup by umožňoval (zachytáváním zprávy WOM_DONE [1, str. 1093] a příslušnou reakcí na ni) spustit přehrávání zvuku věty až po dokončení přehrávání předešlé věty a předčítat tak celý dokument naráz.

Pakliže bychom chtěli pro syntézu řeči používat jiný prostředek, než je klient-server komunikace s Eposem (například kdybychom chtěli použít jiný TTS, třeba ani nefungující na architektuře klient-server), můžeme to udělat jednoduše změnou implementace metod třídy Talker. Jakékoli další změny by neměly být potřeba.

Kapitola 4

Srovnání s podobnými produkty

Testování produktů probíhalo na Device Emulatoru a na zařízení Compaq iPaq 3870 (PocketPC 2002). V závěru popisu každého programu se objeví shrnutí kladů a záporů, které byly při používání jednotlivých aplikací zpozorovány.

4.1 Microsoft Reader

V zařízeních do kapsy s operačním systémem PocketPC 2003 SE je program Microsoft Reader předinstalován. Aby byl schopný pracovat s česky psanými texty (tedy aby uměl zobrazovat správně háčky a čárky), je nutné spustit speciální patch.

Tento program používá svůj vlastní formát (resp. formát společnosti Microsoft) pro zakódování souborů knih (*.lit), založený na firemním systému pro balíčkování on-line nápovědy s prvky DES šifrování. Soubor v tomto formátu se mi podařilo pomocí Active Sync překopírovat do stolního počítače a tedy i blíže prozkoumat. Na základě tohoto zkoumání (formát nebyl ani omylem human-readable) a s uvážením filosofie Microsoftu předpokládám, že se jedná o uzavřený formát, dokumenty tedy není možné editovat a většinu knih bude nutno kupovat. Na webu je k dohledání spousta programů pro konverzi LIT souborů do jiného, přívětivějšího formátu. Za všechny uvedu program "Open Convert .LIT Tool", který dokáže některé knihy v tomto formátu dekódovat do (dle slov autora programu) původního XML/HTML formátu. O případné (ne)legálnosti konvertování LIT souborů se však na internetu vedou spory.

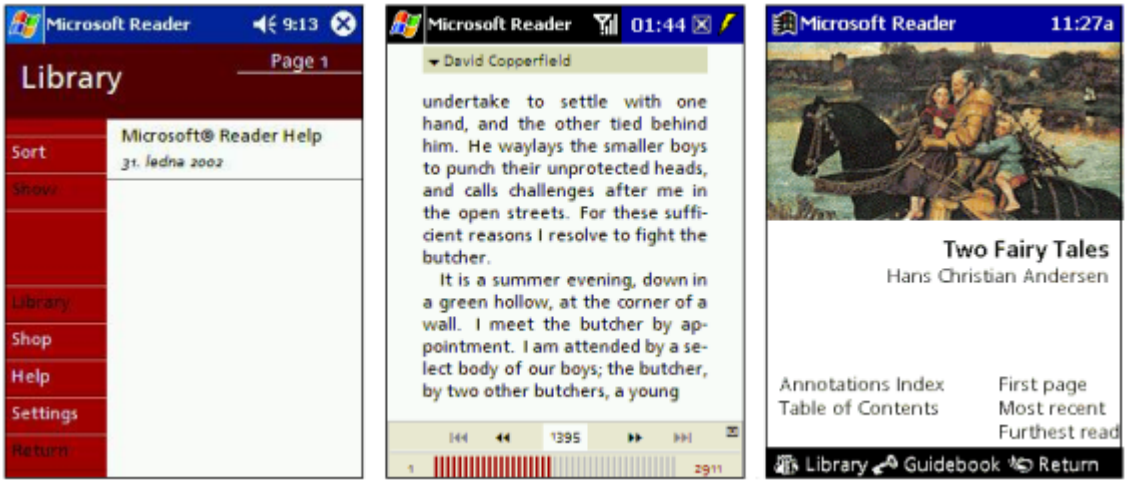
Program Microsoft Reader umí předčítat knihy, které jsou speciálně upraveny, použitím Microsoft SAPI. Program podle manuálu obsahuje zároveň podporu pro Audio Book (Audible 1, 2, 3), pro stahování takto podporovaných knih je ale (opět dle instrukcí v nápovědě k programu) nutná registrace na www.audible.com a instalace speciálního software.

Navigace v knize a ovládání celkově je na velice dobré úrovni, program působí velice přehledně. V knihovně je možné vybrat si knihu, v sekci "Table of Contents" je

možné vybrat příslušnou kapitolu, po otevření kapitoly se listuje po celých stránkách, k dispozici je navíc fulltextové vyhledávání. Samozřejmostí je možnost vkládání záložek do dokumentu.

Plusy: je v základním vybavení PocketPC, jednoduchý a příjemný na ovládání, dobrá navigace v knihách, podpora prodeje knih ze strany Microsoftu (přímo na stránkách firmy je možno kupovat nové tituly)

Mínusy: používá uzavřený formát, je nedostatek knih zdarma, knihy musí být připraveny pro podporu SAPI, pro Audio Books je nutná registrace na Audible.com



Obr. 4.1: Microsoft Reader for PocketPC

4.2 MobiPocket Reader

Jednoznačně nejlepším software pro čtení elektronických knih na PDA je MobiPocket Reader. Aby podporoval česky psané texty, je nutné nainstalovat do zařízení český font (a ten potom v programu nastavit). MobiPocket Reader umí pracovat s plain textem, s HTML dokumenty, knihami Palm Doc (*.pdb) a MobiPocket knihami (*.prc), podpora formátů je tedy velice široká. Program kromě standardních funkcí, jako je třeba vkládání záložek a poznámek nabízí také možnost ohodnotit knihu, umí dokument automaticky plynule scrollovat (nastavitelnou rychlostí), zvýrazňovat časti dokumentu, přepsat část textu (resp. přeškrtnout text původní a nad něj umístit opravu), dále je možno např. nastavit barvy používané v programu.

Ovládání aplikace je vynikající, prostředí je přehledné a barevně jednoduché, program je velice rychlý a nedělá mu problém přeskočit okamžitě ze začátku doprostřed knihy (s tím měl například Adobe Reader značný problém). Program bohužel nepodporuje syntézu řeči. Podle stránek produktu (www.mobipocket.com) je však již zahájena implementace podpory TTS.

Na internetu je k dohledání spousta domácích i světových knih zdarma ke stažení, včetně klasických děl. Program tedy netrpí podobným handicapem, jako MS Reader.

V programu jsou nicméně drobné chyby, aby se ale projevily, musí se uživatel "trochu snažit".

Plusy: přehledné prostředí a jednoduché ovládání, zvýraznění a přepis textu, rychlost provádění operací, široká nabídka knih zdarma, hodnocení knih, plynulé automatické scrollování dokumentu

Mínusy: nutnost instalovat český font, občasné chyby typu "záložka na špatném" místě při změně velikosti fontu

Obr. 4.1: MobiPocket Reader

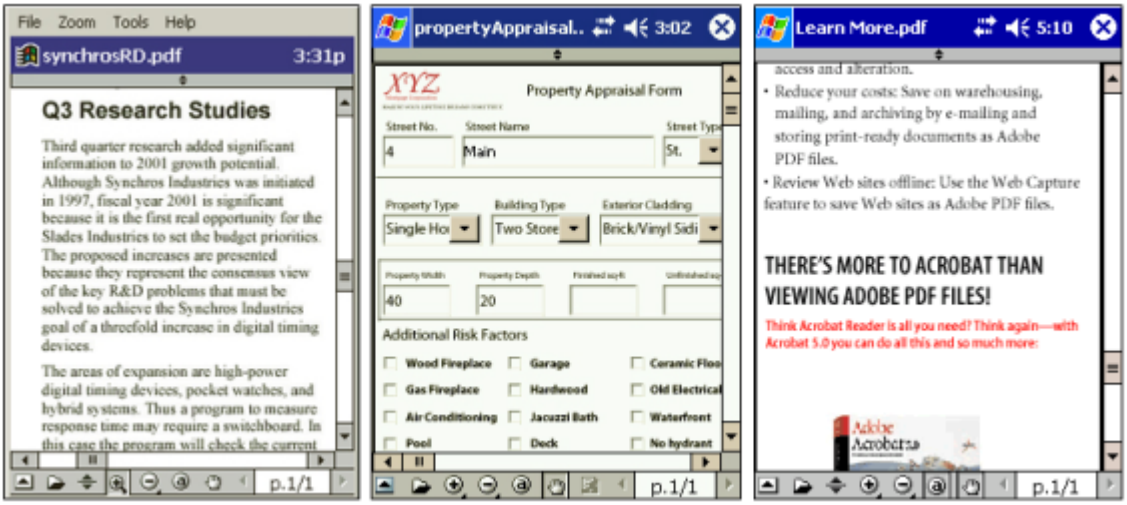
4.3 Adobe Reader for PocketPC 2.0

Program Adobe Reader je tradiční software pro čtení dokumentů ve formátu PDF, a to jak pro stolní PC, tak pro kapesní počítače. Podporován je pouze formát PDF, není tedy jednoduché obsah takového dokumentu modifikovat, a to ani po nahrání na stolní PC, navíc tento fakt velice snižuje prostor použitelnosti programu jen na čtení tohoto formátu. Verze Adobe Readeru pro PocketPC navíc (z mého pohledu překvapivě) nenabízí funkci "Read Out Loud", tedy přečtení obsahu dokumentu pomocí Microsoft SAPI, a to přestože SAPI pro PocketPC existuje a je na relativně dobré úrovni.

Adobe Reader for PocketPC 2.0 nabízí možnost prohlížet dokument jak v originálních rozměrech (potom je typicky nutné scrollovat horizontálně), tak v režimu "Reflow", ve kterém je dokument vyrenderován pro zobrazení na malé obrazovce kapesního počítače. Navigace po dokumentu je na dobré úrovni, je možné zvolit konkrétní stránku nebo záložku, samozřejmě je možné se posouvat i po řádcích. Program nabízí spoustu standardních, ale i vcelku nadstandardních (z pohledu aplikace pro PocketPC) funkcí, jako například fulltextové vyhledávání, funkce zoom-in a zoom-out, tisk dokumentu přímo z PocketPC, možnost použít dokument při prezentaci (připojením PocketPC k projektoru), funkci slide-show a mnohé další. Z mého pohledu byl však Adobe Reader pro PocketPC v provádění některých operací až příliš těžkopádný, troufám si tvrdit, že byl téměř neúnosně pomalý, často na 4-6 vteřin zamrzal.

Plusy: nabízí velice široké spektrum funkcí (slide-show, prezentace, zoom), z pohledu aplikace pro PocketPC nadstandardních, standard pro čtení dokumentu PDF, podporuje režim Reflow i originální zobrazení

Mínusy: není zahrnuta podpora SAPI, při provádění některých operací je poměrně těžkopádný, vyžaduje tedy silnější zařízení, podpora je pouze pro soubory typu PDF a tedy není možnost dokument rozumně editovat



Obr. 4.2: Adobe Reader for PocketPC 2.0

Kapitola 5

Závěr

5.1 Zhodnocení projektu

V projektu e-Voice Reader se podařilo splnit všechny body specifikace, vznikla plnohodnotná aplikace pro čtení elektronických dokumentů.

Faktorem, který se ukázal být pro aplikaci nejvíce limitující, byl výpočetní výkon zařízení PDA a do značné míry i volba architektury typu klient-server pro podporu TTS. Syntéza řeči nebyla realizována dostatečně rychle na to, aby bylo možné ji využívat při čtení dlouhých textů. Tohoto problému se však lze snadno zbavit změnou implementace metod třídy Talker (např. tak lze zvolit jiný TTS engine nefungující na architektuře klient-server, nebo změnit postup komunikace se serverem...). V některých ohledech bylo problematické i používání kontejnerů a funkcí STL. Ostatní volby, například volba WinAPI, knihovny fmod pro přehrávání audia a použití komponenty HTMLView pro vizualizaci dokumentů, se ukázaly být správnými kroky - daly základ pro kvalitní čtečku elektronických dokumentů.

I ve srovnání s ostatními aplikacemi podobného typu si e-Voice Reader vede dobře. Aplikace je jednoduchá a přehledná, při čtení dokumentů bez použití TTS je i dostatečně rychlá. Také formát elektronických knih byl navržen vhodně, protože knihy je možné jednoduše vytvářet a případně editovat (často dochází k chybám při skenování knih pomocí OCR a většina formátů neumožní čtenáři si dokument opravit). Knihy je také možno jednoduše a rychle publikovat na webu. Je zde tedy vytvořen prostor pro možné budoucí šíření tohoto formátu. Formát knih zároveň umožňuje kompletní spojení textu s audio doprovodem. Sice na opačné bázi, než to dělá většina současných formátů (nesynchronizuje se text s audiem, ale audio s textem), to však byl od počátku záměr, protože výsledek se mnohem víc podobá čtení, než poslechu audia.

5.2 Budoucnost projektu

Budeme-li se zabývat budoucností projektu, lze například uvažovat zdokonalenou podporu pro již implementované formáty souborů (hlavně e-maily v quoted-printable), případně rozšíření podpory o nějaký další formát. Podobně nejspíš musí být do budoucna doděláno parsování relativních cest k obrázkům zobrazovaných pomocí HTMLView (viz Implementace). Zřejmě by bylo přínosem přepsat a odlehčit Epos tak, aby se dal k projektu jen přilinkovat, a aby tak nebyla nutnost mít na pozadí spuštěný Epos server a komunikovat pomocí protokolu TTSCP. Do budoucna by se dala zvážit i možnost tvorby aplikace pro automatické nebo poloautomatické spojování knih programu e-Voice Reader s audio obsahem. Vhodnou funkcí, o kterou by bylo možné program obohatit, by mohlo být fulltextové vyhledávání v dokumentech.

Dodatek A

Obsah přiloženého CD

Na přiloženém CD naleznete:

Instalační balíček programu - složka `Install`

Zdrojové kódy aplikace - složka `Source`

Programátorskou dokumentaci - složka `Development`

Uživatelskou dokumentaci - složka `Help`

Text této práce ve formátu PDF - složka `Bakalarka`

LaTeX zdroják a obrázky - složka `LaTeXSource`

Postup instalace:

1. Pokud tak ještě není učiněno, nastavte české regionální nastavení.
2. Nakopírujte balíček instalace.cab na zařízení a spusťte jej.
3. Spusťte soubor epos.exe.
4. Nyní můžete program spustit kliknutím na ikonu v nabídce Programy.

Dodatek B

Specifikace

Popis

Program e-Voice Reader bude program pro čtení několika různých formátů souborů napojený na již hotový TTS (text-to-speech) engine (pracující např. s prostým textem). Tato koncepce umožní hlasem předčítat e-books, e-maily a textové soubory, tedy bude umět nosnou informaci obsaženou v souboru filtrovat např. od formátovacích značek, popisků, obrázků, hlaviček e-mailů, apod.

Formáty souborů, které program (zpočátku) podporuje, jsou tyto:

- prostý text
- elektronické knihy
- soubory typu *.eml - elektronická pošta
- HTML stránky

Koncepce bude zároveň taková, aby umožnila do programu jednoduše přidat podporu pro další typy souboru a tak rozšiřovat spektrum použitelnosti.

Provedení

Program e-Voice Reader bude napsán pro operační systém WinCE, tedy pro zařízení PocketPC.

Zjednodušeně můžeme e-Voice Reader označit za čtečku souborů napojenou na již existující TTS engine. Bude podporovat formáty uvedené výše. Orientace v textu bude probíhat pomocí výběru kapitol, text se bude číst a indexovat po větách.

Zdůrazňuji, že vývoj nového TTS enginu není součástí projektu. Plán je použít TTS engine EPOS (k dispozici na http://epos.ure.cas.cz/) a ten propojit s parserem

souborů. Epos je v současné době dostupný i pro platformu WinCE. Je navíc velice flexibilní - umožňuje modifikovat výšku hlasu, rychlost čtení i další prosodická pravidla (k tomu má speciální ini soubory), nicméně s možností tyto parametry nastavovat dynamicky během přehrávání se při implementaci nepočítá. Budou zvolena kvalitní pevná pravidla. Hlasitost čtení bude obhospodařena mimo Epos pomocí manipulace s ovládáním hlasitosti.

Dále program bude obsahovat podporu pro přehrávání zvuků, kvůli formátu pro elektronické knihy bude zahrnuta i podpora formátu mp3. Jelikož formát mp3 je ve verzi od PocketPC 2002 výše podporován (resp. MediaPlayer jej umí přehrát s výchozími kodeky), bude přehrávání tohoto formátu v režii dll knihoven, které jsou součástí operačního systému PocketPC, nebo knihoven volně dostupných například na webu.

Samotnou činnost programu bude možné ovládat pomocí GUI.

Všechny tyto součásti budou napsány v jazyce C++ (s využitím WinAPI), aby se docílilo maximálního možného výkonu. Využívat se budou standardní knihovny operačního systému a knihovny volně dostupné na webu (např. pro přehrávání zvuku, mp3). Komunikace mezi jednotlivými komponentami je velice důležitou součástí návrhu, protože rozhodujícím způsobem ovlivní výslednou použitelnost systému, jeho rozšiřitelnost i samotný výkon.

Ještě jednou zde vyjmenujme jednotlivé součásti programu:

- parser souborů - text, HTML, e-mail, ...

- přehrávač zvukových souborů napsaný v C++ s využitím knihoven OS podporující formát mp3

- grafický interface umožňující ovládání činnosti jednotlivých komponent zobrazuje nosnou informaci jako klasický reader

- použití existujícího TTS enginu pro předčítání textu

Popis součástí, idea návrhu

Parser

Parser je komponenta programu, která má za úkol zpracovat soubor daného typu, tzn. číst soubor ze vstupního proudu, zpracovat daný formát souboru a posílat TTS enginu jednotlivé bloky textu pro přečtení. Bloky budou tvořeny větami. Rovněž indexování souboru bude probíhat po větách.

Počáteční myšlenka konstrukce parseru ovlivňuje jeho další rozšiřitelnost. V projektu e-Voice Reader použijeme parser postavený na dědičnosti.

Hlavní třída, řekněme TParser, obsahuje datové prvky tak, aby mohla pojmout vstupní formát, a hlavní metodu v ní představuje virtuální metoda FillBuffer().

Třída od sebe má odvozeny potomky, pojmenujme je například TEMailParser, THTMLParser, atd., kteří budou provádět zpracování konkrétního typu vstupu (pomocí volání virtuální funkce FillBuffer()), a výsledek následně odesílat Eposu.

Takováto koncepce umožňuje přidávat nové typy souborů napsáním poděděné třídy a v ní jedné virtuální funkce FillBuffer() a konstruktoru (např. u e-mailu se v konstruktoru přečte odesílatel a předmět).

Výstup, který parser posílá, musí být přečten TTS enginem a následně přehrán. Tato operace přirozeně nějakou dobu trvá. Proto je vhodné zavést do parseru mechanismus, který bude rozhodovat, je-li možné pokračovat v parsingu, nebo má-li se čekat na dokončení nějaké operace, například zmíněné operace čtení. Proto bude zaveden pomocný buffer (pro předzpracování dat) a zámky, které umožní parser zastavit do doby, než skončí aktuálně prováděná operace využívající data vzniklá předchozím parsingem.

Přehrávač zvukových souborů

Přestože součástí OS WinCE je i MediaPlayer - program schopný přehrát většinu dnes známých formátů mediálních souborů - bude e-Voice Reader obsahovat jednodušší součást pro přehrávání zvukových médií s některými specifickými funkcemi. Mezi požadavky kladenými na přehravač patří například to, aby nám oznámil, že již skončil s přehráváním zvuku.

Grafické uživatelské rozhraní (GUI) - počáteční úvahy
Výsledná podoba GUI se bude v průběhu vývoje modifikovat dle potřeb aplikace, zde proto uvádím jen východiska návrhu.

Aby bylo umožněno ovládání přehrávání, bude součástí e-Voice Readeru i přívětivý uživatelský interface, nutně navíc zohledňující možnost, že uživatel aplikace je zrakově handicapovaný. Základem úspěchu je bezesporu jednoduchost. Abychom si mohli lépe rozmyslet, co vlastně na displej umístit, shrneme si body, které budeme od GUI očekávat.

Naše uživatelské rozhraní musí zejména:

- umět zobrazit aktuálně čtený blok textu

- zastavit a opětovně spustit čtení, nejlépe asi použitím hardwarových tlačítek

- přejít do menu pro nastavení a výběr souboru ke čtení, nejlépe tlačítkem "Menu" dole na displeji

- používat dostatečně kontrastní barvy a velké písmo, tlačítka,...

- pohybovat se ve čteném textu tam a zpět - nejlépe dořešit výběrem "kapitol" přes menu

Po spuštění programu bude možné vybrat typ souboru, následně specifikovat konkrétní soubor, ten poté přehrát. Během přehrávání se bude zobrazovat čtená věta, klientská plocha bude rovněž obsahovat tlačítka pro ovládání přehrávání, samozřejmě pak hlavní menu.

Literatura

[1] PETZOLD CH.: *Programování ve Windows*, Computer Press (1999)

[2] *Microsoft Development Network*, http://msdn.microsoft.com/

[3] *Dokumentace k TTS enginu Epos*, http://epos.ure.cas.cz/