FAKULTA ELEKTROTECHNIKY A INFORMATIKY STU V BRATISLAVE

Tomáš Šághy

DIFÓNOVÝ SYNTETIZÁTOR SLOVENČINY V JAZYKU PHP

Diplomová práca

Vedúci diplomovej práce: doc. Ing. Gregor Rozinaj, CSc.

Bratislava, máj 2010

Zadanie:

1. Naštudujte metódy syntézy reči.
2. Analyzujte možnosti jazyka PHP pre aplikácie syntézy reči.
3. Navrhnite a realizujte modul pre difónovú syntézu reči v PHP.
4. Vytvorte webovú aplikáciu pre demonštráciu PHP modulu pre difónovú syntézu
5. Navrhnutú aplikáciu otestujte a výsledky vyhodnoťte!

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Študijný program: Telekomunikácie

Autor: Tomáš Šághy
Diplomová práca: Difónový syntetizátor slovenčiny v jazyku PHP
Vedúci diplomovej práce: doc. Ing. Gregor Rozinaj, CSc.
Máj 2010

Diplomová práca sa zaoberá problematikou syntézy reči a jej cieľom je vytvorenie difónového syntetizátora slovenčiny v programovacom jazyku PHP. V práci sa rozoberá história a vývoj rečových syntetizátorov, prirodzený proces tvorby reči človekom, ako aj sluchové vnímanie reči. Podrobnejšie sa venuje metódam syntézy reči a ich deleniu. Väčšia časť práce opisuje návrh difónového rečového syntetizátora a jeho realizáciu v jazyku PHP. Popisuje aj problémy späté s použitím tohto jazyka. Záverom opisuje možnosti použitia v reálnych webových aplikáciách a demonštruje ich na programe čítajúcom články na informačnom portáli www.sme.sk.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY
Degree Course: Telecommunications

Author: Tomáš Šághy
Diploma Thesis: Diphone Synthesizer of Slovak language in PHP
Supervisor: doc. Ing. Gregor Rozinaj, CSc.
May 2010

This thesis deals with speech synthesis and its aim is to create diphone synthesizer of Slovak language using PHP programming language. The paper discusses the history and development of speech synthesizers, natural process of human speech and auditory perception of speech. In detail it deals with methods of speech synthesis and their categorization. Considerable part of this thesis describes the design of the diphone speech synthesizer and possibilities of its implementation in PHP. It also describes problems associated with using this language. Finally, it describes the uses in real web applications and demonstrates it by the program capable of reading articles on the information portal www.sme.sk.

Obsah

1. Úvod

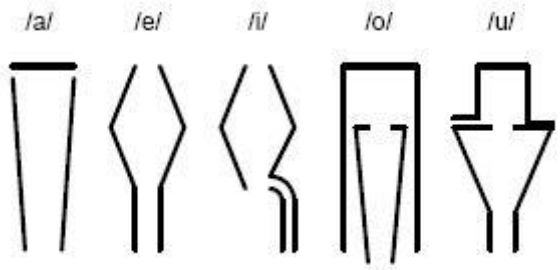
Komunikácia prostredníctvom reči je základným a najdôležitejším spôsobom prenosu informácií medzi ľuďmi od praveku až po súčasnosť. Je to proces, ktorý začína prípravou správy v mozgu rečníka a vytvorením akustického signálu v jeho vokálnom trakte a končí rozpoznaním signálu a porozumením významu prenášanej správy poslucháčom.

V prípade, že s človekom prostredníctvom reči komunikuje počítač (informačný systém), ide o totožný proces. V úlohe rečníka je na vytvorenie umelej reči použitý rečový syntetizátor, v úlohe poslucháča systém rozpoznávania reči.

Samotný syntetizátor je možné realizovať rôznymi algoritmickými spôsobmi a v dnešnej dobe existuje niekoľko syntetizátorov slovenčiny, ktoré sú realizované vo viacerých programovacích jazykoch. Táto práca sa zaoberá realizáciou difónového syntetizátora v PHP. Čo sa týka výberu programovacieho jazyka, ide v syntéze slovenčiny o unikátny projekt a aj syntetizátory iných jazykov dostupné na internete, ktoré sa prezentujú ako syntetizátory napísané v PHP, sú poväčšinou rozšíreniami ku knižniciam implementovaným v iných programovacích jazykoch. Tento syntetizátor bude teda ojedinelým otestovaním vhodnosti použitia jazyka PHP pre aplikáciu syntézy reči.

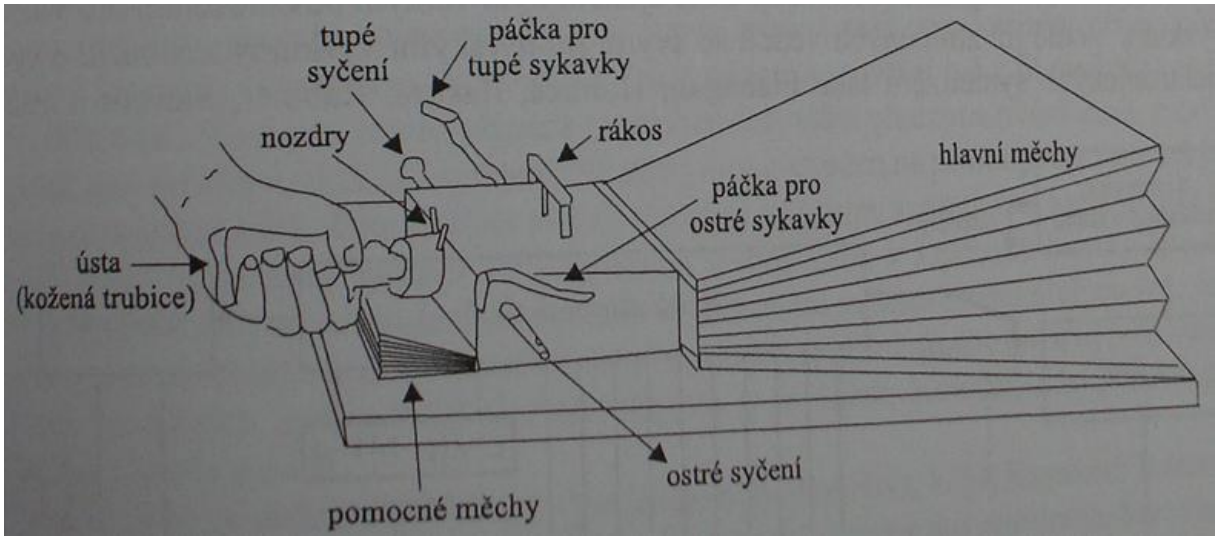
2. História syntézy reči

Prvé syntetizátory reči boli mechanické. Vôbec prvé pokusy o umelé vytváranie reči sa pripisujú ruskému profesorovi Christianovi Kratzensteinovi, ktorý v roku 1779 vysvetlil fyziologické rozdiely medzi piatimi dlhými samohláskami (/a/, /e/, /i/, /o/, /u/). Tieto samohlásky potom umelo vytváral pomocou zostavených akustických rezonátorov (viď. *obr.1*), ktoré aktivoval pomocou vibrujúcej trstiny.



Obr.1: Kratzensteinove akustické rezonátory [1]

O niečo neskôr, v roku 1791, predstavil Wolfgang von Kempelen prvý hovoriaci stroj (viď. *obr.2*). Jednotlivé časti zariadenia napodobňovali artikulačné orgány človeka (napríklad pľúca boli reprezentované hlavnými mechmi) a údajne bol stroj schopný generovať dokonca jednoduché vety. Kempelenov výskum trval viac ako 20 rokov a jedným z jeho výsledkov bol poznatok, že hlasový trakt významne vplýva na proces tvorby reči. Dovtedy sa za centrum vytvárania reči považovali výhradne hlasivky.



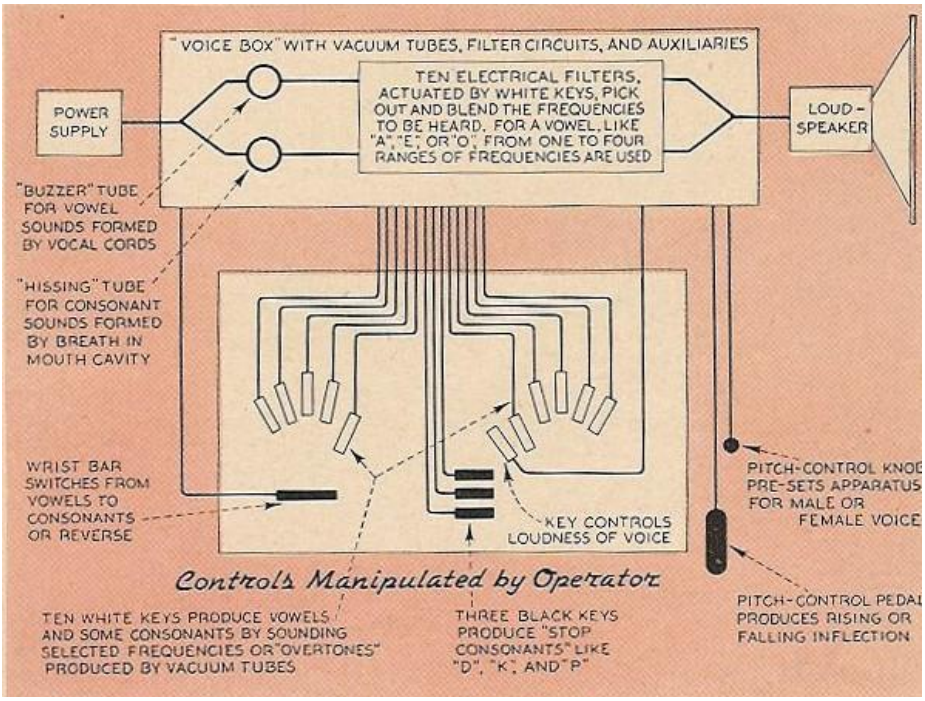
Obr.2: Hovoriaci stroj Wolfganga von Kempelen [2]

V roku 1922 vytvoril Stewart prvý elektronický rečový syntetizátor, ktorý modeloval prvé dva formanty hlasového traktu pomocou rezonančných obvodov. V roku 1932 pridali Obata a Teshima tretí formant, čím zvýšili zrozumiteľnosť generovaných samohlások. Tri formanty sú všeobecne považované za postačujúce pre vytvorenie zrozumiteľnej syntetickej reči.

V roku 1939 predstavil Homer Dudley v New Yorku VODER (*Voice Operating Demonstrator*) – prvý elektronický syntetizátor súvislej reči (viď. *obr.3*). Desať kláves vytváralo na základe polohy zápästného prepínača buď znelé (generované periodickým signálom, ktorého frekvencia sa dala meniť stláčaním pedálu) alebo neznelé (generované šumom) zvuky.

Prvý formantový syntetizátor PAT (*Parametric Artificial Talker*), ktorý pozostával z troch paralelne zapojených formantových rezonátorov, predtavil Walter Lawrence v roku 1953. Približne v rovnakom čase predstavil Gunnar Fant syntetizátor OVE I (*Orator Verbis Electris*), v ktorom boli rezonátory zapojené do kaskády.

S nástupom číslicových počítačov nastal zlom vo vývoji syntézy reči, keďže bolo pomocou nich možné simulovať syntetizátory miesto ich hardwarovej realizácie. S vývojom rečových modelov sa tiež vyvíjala problematika vytvárania reči z textu (TTS).



Obr.3: Schéma elektronického syntetizátora VODER [3]

V roku 1976 vytvoril Raymond Kurzweil prístroj na čítanie tlačených stránok, určený pre nevidiacich. Išlo o počítač, ktorý pomocou scanneru načítal stránku, zdigitalizoval ju a získaný text previedol na reč.

Približne v rovnakom období vznikol prvý integrovaný obvod pre syntézu reči – Votrax. Čip obsahoval kaskádový formantový syntetizátor a dolnopriepustné filtre pre vyhladzovanie reči.

V roku 1985 predstavili Charpentier a Moulines metódu PSOLA (*pitch-synchronous overlap-and-add*) pre úpravu prozodických charakteristík zreťazovaných rečových jednotiek.

V 90. rokoch sa začínajú objavovať prvé systémy využívajúce korpusovú syntézu reči, ktorých rečové jednotky sú vytvárané automaticky na základe rozsiahlych rečových korpusov.

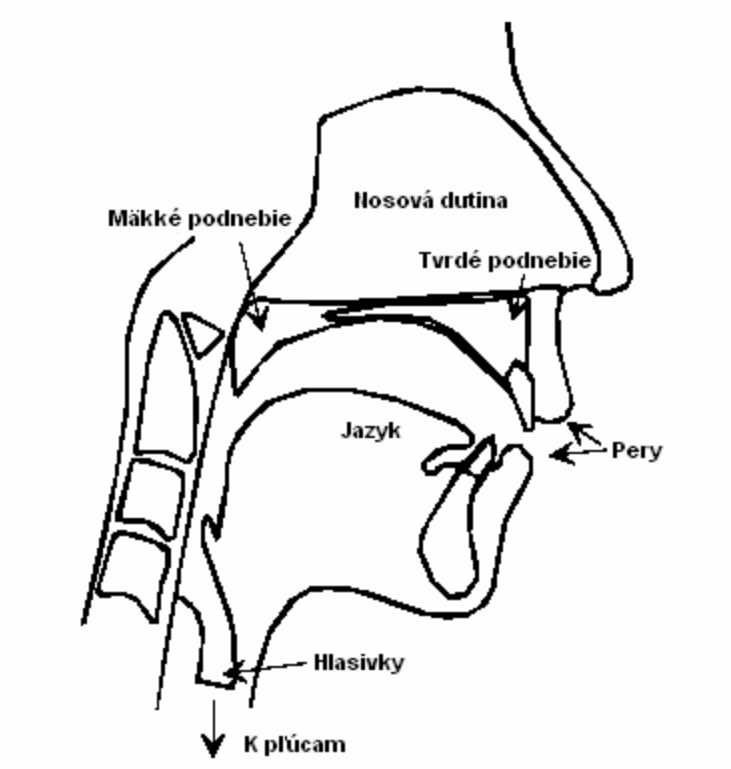
3. Proces tvorby a vnímania reči človekom

Na vytváraní reči sa v ľudskom tele podieľa viacero orgánov a častí tela. Paradoxne pre väčšinu z nich platí, že ich primárna funkcia nie je spojená s tvorbou reči. Vnímanie reči je zabezpečené sluchovými orgánmi, ktoré sa nachádzajú v uchu.

3.1 Produkcia reči

Na vzniku reči sa podieľajú tri zložky – respirácia, fonácia a artikulácia. Preto aj rečové ústroje delíme na:

- dýchacie (respiračné) ústroje
    - bránica,
    - pľúca,
- hlasový (fonančný) ústroj
    - hlasivky,
- upravujúce ústroje
    - pery,
    - zuby,
    - čeľusť,
    - ďasná,
    - tvrdé podnebie,
    - mäkké podnebie,
    - sánka.



Obr.4: Rečové orgány človeka [4]

Tvorba reči začína v pľúcach, kedy prúd vzduchu, ktorý je vydychovaný, prejde hrtanom a narazí na hlasivky, ktoré prúdom vzduchu rozkmitá. Prechodom vzduchu hlasivkami vzniká vzduchová vlna, ktorú ľudia vnímajú ako zvuk. Svaly ovládajúce hlasivky držia štrbinu medzi hlasivkami otvorenú, pokiaľ jedninec nerozpráva, čím umožňujú bežné dýchanie.
Pri tvorbe samohlások je hlasivková štrbina takmer úplne zatvorená a svaly sú napäté. Vzniká tak pravidelné kmitanie a zvuk, ktorý je čisto tónový. Pri znelých spoluhláskach je v hlasivkách nižšie napätie než pri samohláskach a to spôsobuje menej pravidelné kmitanie a zvuk, ktorý nie je čisto tónový. Pri neznelých spoluhláskach sú hlasivky v pokoji, keďže tieto neobsahujú základný hlasivkový tón. Tvoria sa teda až v dutinách nad hrtanom (hlasivky sú v tomto prípade povolené úplne a teda hlasová štrbina nie je otvorená tak ako pri bežnom dýchaní).
Poslednou zložkou reči je artikulácia. Jej význam spočíva v schopnosti vytvárať množstvo odlišných zvukov, ktoré sú charakteristické pre jednotlivé jazyky. Zúčastňujú sa jej dutiny a artikulačné orgány. Dutiny menia frekvenčné vlastnosti základného hlasového tónu a artikulačné orgány ovplyvňujú prechod dutinami – v rôznych miestach zužujú alebo zastavujú prechod vzduchu. Vydychovaný vzduch prechádza cez tieto „prekážky“ a vzniká šum. Šumov a teda aj výsledných zvukov je veľké množstvo, keďže aj spôsobov ako artikulačnými orgánmi meniť priechod vzduchu je veľmi veľa.

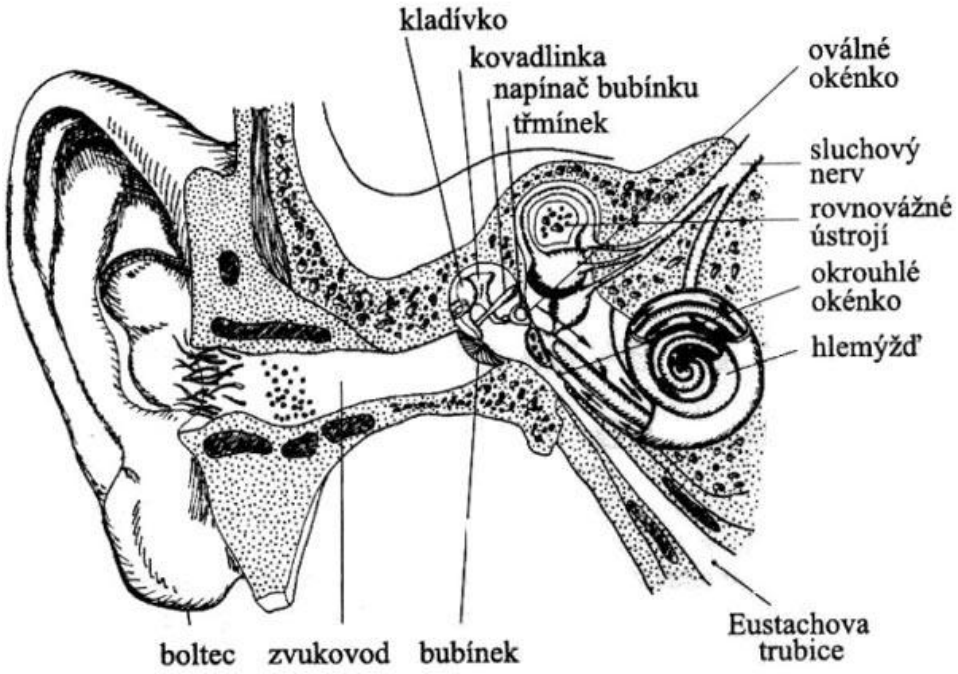
3.2 Vnímanie reči

Človek vníma reč aj všetky zvuky pomocou sluchových orgánov, nachádzajúcich sa v uchu. Ucho delíme na vonkajšie stredné a vnútorné a tie sa skladajú z nasledovných častí:

- vonkajšie ucho
    - ušný boltec,
    - zvukovod (dlhý približne 2,5 cm),
- stredné ucho
    - bubienok,
    - napínač bubienka,
    - ušné kostičky – prenášajú chvenie z bubienka na tekutinu v slimáku
        - kladivko,
        - nákovka,
        - strmienok,
        - väzivo,

- Eustachova trubica,

- vnútorné ucho
  - slimák,
  - Cortiho orgán,
  - sluchový nerv,
  - rovnovážne centrum.



Obr.5: Anatómia ucha [14]

Samotné spracovanie zvuku pomocou týchto orgánov prebieha tak, že ušný boltec zachytáva vzduchové vibrácie a smeruje ich do zvukovodu. Zvukovod slúži ako akustická trubica a privádza vlnenie k bubienku. Ten sa nárazom vzduchových vĺn rozochvieva (pre toto rozochvenie je nevyhnutný napínač bubienku, ktorý ho drží napnutý). Chvenie ďalej prechádza cez ušné kostičky. Strmienok ako posledná kostička prenáša chvenie na tekutinu v slimáku. Riasnaté receptory v Cortiho orgáne zaznamenávajú vibrácie tekutiny a premenia túto mechanickú energiu na elektrické impulzy. Pomocou nervových zakončení sa zbierajú do sluchového nervu, ktorý vedie do spánkového laloku, kde mozog tieto signály ďalej spracováva.

Rozsah frekvencií, ktoré je zdravé ľudské ucho schopné zachytiť sa uvádza v rozmedzí 20 – 20 000 Hz, avšak receptory prenášajúce chvenie na elektrické impulzy nie sú schopné regenerovať, preto aj ľudské ucho stráca schopnosť vnímať spomínané spektrum frekvencií.

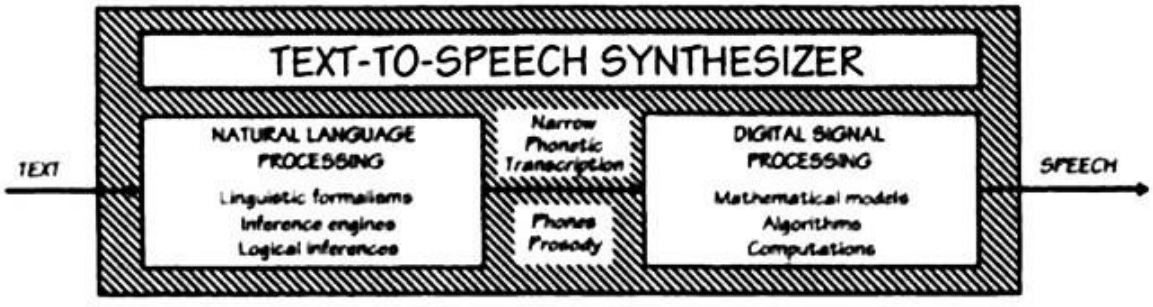
4. Princípy syntézy reči

Syntéza reči je proces vytvárajúci zo vstupného textu umelý rečový signál. Tento proces má niekoľko oblastí použita a rôzne parametre hodnotenia kvality. Nejde o triviálny problém. Aj v prípade, že by sme mali dostupný rozsiahly slovník bežných slov v slovenčine, systém syntézy reči sa musí vysporiadať so skratkami, obrovským počtom vlastných mien a inými nástrahami jazyka. Pokiaľ má naviac znieť prirodzene, mala by mať vygenerovaná reč primeranú intonáciu. Zásadnou otázkou je tiež kompromis medzi požiadavkami na čo najvyššiu kvalitu reči a čo najnižšiu pamäťovú náročnosť a výpočtovú rýchlosť.
Najdôležitejšími parametrami hodnotenia kvality syntézy reči sú zrozumiteľnosť a prirodzenosť. [4]

- Zrozumiteľnosť je dôležitým parametrom pre možnosť použitia systému v hlučnom prostredí a v dialógoch, kde výstup neobsahuje takmer žiadne syntaktické a sémantické redundancie.
- Prirodzenosť je dôležitým parametrom pre prijatie systému používateľom ako celku, keďže počúvanie syntetizovanej reči vyžaduje zvýšenú koncentráciu a dokáže poslucháča pomerne rýchlo unaviť.

Pokiaľ je oblasť výberu vstupného textu neobmedzená a systém nezohľadňuje sémantiku (nevie, čo hovorí), označujeme tento proces ako TTS (*text-to-speech*). Okrem TTS sa v súčasnosti používa CTS (*content-to-speech*), kde sa jedná o hovorený výstup z databáz informácií alebo dialógového systému a tiež *reproduktívna syntéza reči* (*reproductive speech synthesis*), v ktorej sa reč vytvára skladaním dlhších (vopred nahratých) vzoriek. [4]
TTS syntéza sa skladá z viacerých krokov, ktoré môžeme pre zjednodušenie rozdeliť do dvoch základných častí: [5]

- NLP (*natural language processing*), označovanú tiež ako syntéza vyššej úrovne a
- DSP (*digital signal processing*), označovanú aj ako syntéza nižšej úrovne.



Obr.6: Jednoduchá schéma TTS systému [5]

4.1 Syntéza vyššej úrovne

Úlohou bloku NLP je spracovať vstupný text do podoby údajov, na základe ktorých môže blok DSP vygenerovať výstupnú reč. Týmito údajmi sú postupnosť foném (fonetická transkripcia) a informácia o prozódii. Nevyhnutným krokom v tomto procese je syntakticko-morfologická analýza, v priebehu ktorej sa napríklad hľadajú hranice viet a súvetí na základe interpunkčných znamienok, identifikujú sa skratky, číslovky, cudzie slová a iné elementy, ktorých ortoepický prepis nemožno urobiť podľa slovenských pravidiel.
Spracovanie vstupného textu v tomto bloku je teda možné rozdeliť na tri kroky: [1]

- predspracovanie textu, v priebehu ktorého sa špeciálne znaky, skratky a číselné výrazy prepisujú na textové vyjadrenie,
- analýza výslovnosti, ktorej výsledkom je prepis textu do fonetickej abecedy,
- analýza prozódie, v priebehu ktorej sú určené prozoické vlastnosti vstupného textu.

4.2 Syntéza nižšej úrovne

Blok DSP je zložkou, ktorá vytvára časový priebeh výstupného signálu, teda ide o samotný proces syntézy reči. Tento proces sa dá podľa modelu použitého na generovanie reči rozdeliť na tri typy: [6]

- artikulačnú syntézu,
- formantovú syntézu,
- konkatenačnú syntézu.

Zatiaľ čo prvé dva typy spadajú do skupiny *syntézy podľa pravidiel*, v prípade konkatenačnej syntézy ide o *syntézu riadenú dátami*. Prvá skupina syntetizuje reč pomocou súboru manuálne vytvorených pravidiel, kým v druhej sú parametre syntetizátora získavané automaticky z rečových dát. Najmodernejšia syntéza podľa pravidiel je síce ľahko pochopiteľná, ale znie neprirodzene, keďže do malého množstva pravidiel nie je jednoduché zachytiť všetky nuansy bežného jazyka.

4.2.1 Artikulačná syntéza

Artikulačná syntéza priamo modeluje ľudské orgány zapojené do procesu vytvárania reči. Predstavuje teda potenciálne najlepšiu metódu syntézy reči, ale kvôli značnej náročnosti

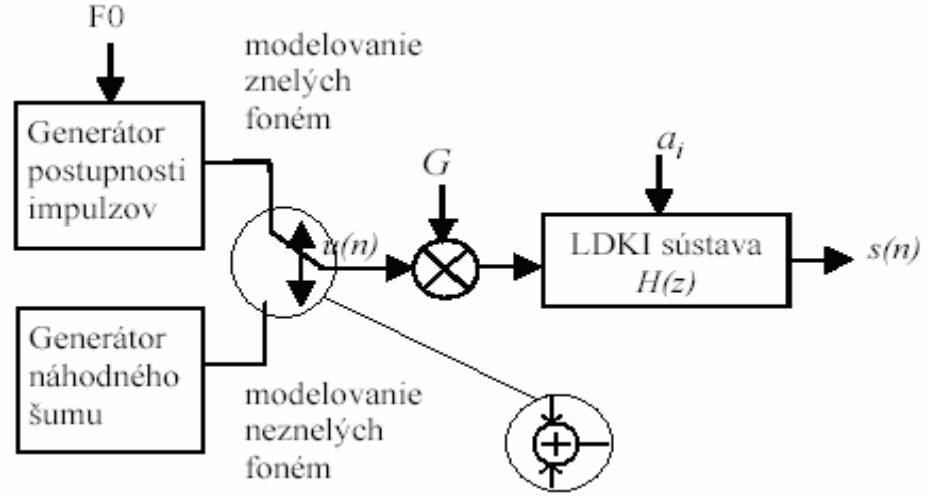
implementácie a podstatne vyššej výpočtovej zložitosti sa jej v praxi venuje menej pozornosti oproti iným metódam. [1]

Pri artikulačnej syntéze je kritickým faktorom dostatočný počet správnych parametrov, ktoré však nie je jednoduché získať. Parametrizovať vokálny trakt je možné nasledovnými metódami: [4]

- využitím reflekčných koeficientov vypočítaných pri LPC *(linear predictive coding)* analýze reči Levinson-Durbin algoritmom, čo však neposkytuje dostatočnú kvalitu,
- využitím zobrazovania pomocou magnetickej rezonancie, ktorou sa dá získať 3D model vokálneho traktu, no metóda je časovo aj finančne veľmi náročná,
- meraním akustickej impedancie vokálneho traktu, pri ktorom sa do vokálneho traktu vysielajú prietokové signály pri artikulácii jednotlivých zvukov a po odraze od hlasiviek sa zmerajú pri perách, alebo
- skusmo, kedy sa syntetizovaná reč s meniacimi sa parametrami porovnáva s prirodzenou rečou a pri najlepšej zhode sú nastavené parametre vyhodnotené ako výsledné.

4.2.2 Formantová syntéza

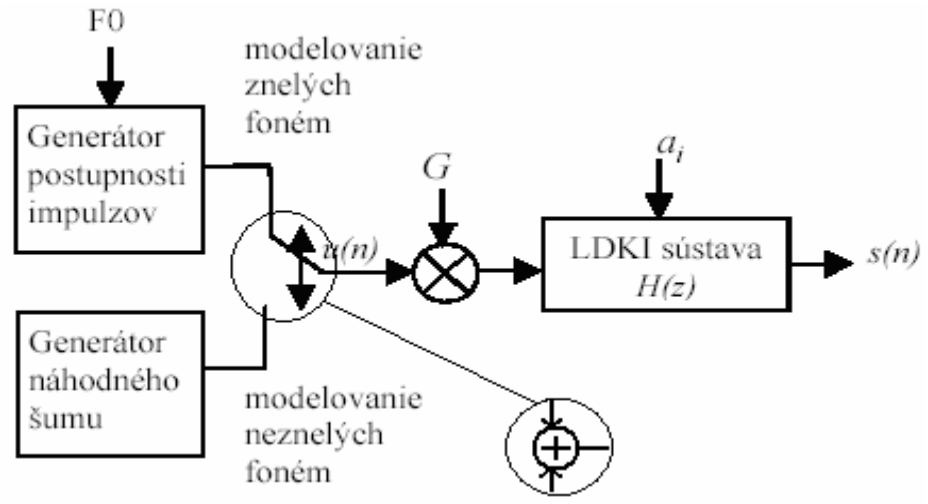
Formantová syntéza je založená na teórii zdroja a filtra (*source-filter theory*), podľa ktorej je vytváranie reči modelované dvomi navzájom nezávislými zložkami. Zdroj budenia generuje kváziperiodický sled hlasivkových impulzov (hlasivkový signál) pre znelé zvuky a šum pre neznelé zvuky, zatiaľ čo akustický filter reprezentuje frekvenčnú odozvu vokálneho traktu.



Obr.7: Source-filter model reči [4]

Táto metóda je schopná vyprodukovať nekonečný počet zvukov, čo ju robí flexibilnejšou oproti konkatenačnej syntéze. Aby sa pri nej dosiahla zrozumiteľná reč, sú potrebné minimálne tri formanty. Pre kvalitnú reč je potrebné generovať aspoň 5 formantov. [4]

Najznámejším príkladom formantovej syntézy je Klattov syntetizátor (viď. *obr.8*), ktorý kombinuje kaskádové a paralelné zapojenie rezonátorov (hybridná štruktúra). Znelá vetva je budená pulzným generátorom, ktorý simuluje činnosť hlasiviek. Rezonátory zapojené do série simulujú vokálny trakt a nazálny filter v sérii s rezonančným trubicovým modelom simuluje dominantné rezonancie a anti-rezonancie nosovej dutiny. Neznelé zvuky vychádzajú z napodobenia frekvenčných magnitúd neznelých alofón pomocou frekvenčných magnitúd filtrovaného šumu. [4]



Obr.8: Základná architektúra vokálneho traktu v Klattovom formantovom syntetizátore [4]

Medzi hlavné výhody formantovej syntézy patria: [2]

- nízky počet parametrov – pre zrozumiteľnú reč potrebuje formantový syntetizátor asi 40 – 60 parametrov, najčastejšie nimi sú: [4]
    - základná hlasivková frekvencia,
    - pomer otvorenia hlasiviek k celkovej perióde pri znelom budení,
    - stupeň znelosti v budení,
    - formantové frekvencie a ich amplitúdy,
    - frekvencia prídavného nízkofrekvenčného rezonátora,
    - intenzita nízkofrekvenčnej a vysokofrekvenčnej oblasti,
- jednoduché riadenie prozodických charakteristík – zmena frekvencie, časovania a intenzity hlasivkového tónu,

- zachytenie koartikulačných javov – pokiaľ sú nájdené pravidlá pokrývajúce prechodové a koartikulačné javy, je ich možné syntetizovať rovnako ako ostatné zvuky,
- konštantná kvalita – všetky vytvorené vety znejú približne rovnako kvalitne,
- jednoduchá zmena hlasu – zmenou niekoľkých parametrov umožňuje zmeniť hlas z mužského na ženský, detský a pod.

Hlavnými nevýhodami formantovej syntézy sú: [2]

- prácne manuálne hľadanie a nastavovanie pravidiel – hľadanie pravidiel, ktoré opisujú nastavovanie jednotlivých parametrov syntetizátora v závislosti na fonetickom kontexte,
- vzájomná interkcia medzi hodnotami parametrov – hodnoty jedného parametra môžu ovplyvniť iné parametre, alebo môžu byťobmedzené ich hodnotami, preto je v praxi nutná dôkladná testovacia fáza pre overenie nastavenia parametrov,
- zložité vytváranie niektorých zvukov – napríklad krátkodobé dynamicky sa meniace zvuky je veľmi náročné vytvárať podľa pravidiel,
- nízka prirodzenosť reči – napriek vysokej kvalite a zrozumiteľnosti je prirodzenosť reči pomerne nízka.

4.2.3 Konkatenačná syntéza

V konkatenačnej syntéze je rečový segment zosyntetizovaný jednoduchým pospájaním príslušných rečových segmentov. Výhodou tohto prístupu je, že nepotrebuje žiadne pravidlá a každý segment je prirodzený, preto sa dá očakávať prirodzený výstup.
Konkatenačná syntéza sa dá deliť podľa viacerých hladísk: [2]

- Podľa spôsobu vytvárania rečových jednotiek v rečovom korpuse na
    - ručné vytváranie, ktoré je veľmi prácne a časovo náročné a naviac by ho mal vykonávať iba trénovaný človek,
    - automatické vytváranie, pri ktorom sa rečové jednotky vytvoria bez zásahu človeka na základe daného rečového korpusu.
- Podľa spôsobu reprezentácie rečových segmentov na
    - neparametrickú reprezentáciu, ktorá býva algoritmicky nenáročná, ale poskytuje len obmedzené možnosti spektrálnych a prozodických modifikácií,

- parametrickú reprezentáciu, pri ktorej sa jednotlivé segmenty na základe zvoleného rečového modelu zakódujú pomocou daných parametrov a počas syntézy sa dekódujú a zreťazujú.

- Podľa toho, či pri nej dochádza k prozodickým a spektrálnym modifikáciám na
  - syntézu bez modifikácií, pri ktorej sa segmenty zreťazujú presne v tvare, v akom boli uložené,
  - syntézu s modifikáciami, pri ktorej sa upravujú prozodické alebo spektrálne charakteristiky s cieľom minimalizovať nespojitosti na hraniciach jednotlivých segmentov.
- Podľa rozsahu „slovnej zásoby“ v generovanej reči na
  - syntézu s obmedzeným slovníkom, ktorá je schopná generovať slová a vety iba z vopred danej špecifickej oblasti (napr. vlaková stanica alebo predpoveď počasia),
  - syntézu s neobmedzeným slovníkom, pri ktorej sa rečové jednotky vytvárajú univerzálne, aby bolo pomocou nich možné syntetizovať ľubovoľný text.
- Podľa druhu rečovej databázy na [4]
  - syntézu využívajúcu uzavretú množinu elementov, pri ktorej je každý element v databáze reprezentovaný jedinou zvukovou realizáciou a napájanie elementov sa riadi pravidlami a nevyhľadáva sa pri ňom najlepší kandidát z viacerých,
  - syntézu výberom jednotiek premenlivej dĺžky z databázy, pri ktorej sa ako databáza využíva rečový korpus, ktorým je viacero rôznych prehovorení jedného rečníka s čo najbohatším foneticko-prozodickým obsahom. Samotná syntéza prebieha tak, aby sa výsledný signál skladal z čo najmenšieho počtu segmentov. Tento typ syntézy sa tiež označuje ako *korpusová syntéza*.

4.2.3.1. Difónová syntéza

Difónová syntéza je podmnožinou konkatenačnej syntézy využívajúcej uzavretú množinu elementov – difón.

Difóna je úsek rečového signálu od polovice jednej hlásky po polovicu nasledujúcej hlásky (viď. *obr.9*). Jej výhodou je, že vo svojom strede zachováva prechodovú koartikulačnú informáciu a ustálené okrajové časti sú vhodné na spájanie s inými elementami. Pokiaľ obsahuje jazyk N foném, existuje N^2 potenciálnych difón. V praxi sa ale mnoho z nich

nevyskytuje, takže postačuje menší počet. Difóny boli medzi prvými jednotkami používanými v konkatenačných systémoch, keďže poskytujú veľmi dobré výsledky.

Obr.9: Rozdelenie slova počítač na jednotlivé difóny

5. Syntéza reči použitím jazyka PHP

PHP (*PHP: Hypertext Preprocessor*) je celosvetovo rozšírený mnohoúčelový skriptovací jazyk určený primárne pre webových vývojárov. Tri hlavné oblasti, v ktorých sa používa: [8]

- skriptovanie na strane servera, čo je masovo najrozšírenejším použitím a je k nemu potrebný PHP parser, webserver a webový prehliadač,
- skriptovanie z príkazového riadku, ku ktorému postačuje PHP parser,
- vytváranie desktop aplikácií, ktoré je možné s využitím rozšírenia PHP-GTK.

PHP beží na všetkých hlavných operačných systémoch a má podporu väčšiny dnešných webserverov. Jednou z jeho významných vlastností je podpora širokej škály databáz.

5.1 Analýza možností PHP pre realizáciu rečového syntetizátora

Absencia projektov syntézy reči v jazyku PHP dáva tušiť, že tento jazyk nie je v konkurencii častejšie sa objavujúcich riešení v jazykoch C a Java najvhodnejším variantom.

Pre realizáciu syntetizátora sú nevyhnutné nástroje na prácu s reťazcami, matematické operácie a prácu so súbormi. Tieto PHP ako každý štandardný programovací jazyk natívne podporuje. Takisto je nevyhnutný nástroj na prácu so zvukom (v našom prípade so vzorkami zvukových súborov vo formáte *wave*), ktorý pri jazyku PHP absentuje. Vďaka podpore práce s binárnymi dátami (funkcie *pack* a *unpack*) je však možné takýto nástroj vytvoriť. Na druhej

strane poskytuje PHP jednoduché rozhrania pre prácu s databázou, ktorú môžeme vhodne využiť pre indexáciu rečovej databázy a vyhľadávanie v slovníku.

PHP je primárne určené pre vytváranie webových aplikácií a rozdiely vo výkone oproti jazykom ako C, C++, alebo Java sú značné. Je preto otázne, či bude algoritmicky pomerne náročný rečový syntetizátor implementovaný v PHP použiteľný v reálnom čase vo webovom prostredí. Na druhej strane syntetizátor realizovaný v PHP je možné jednoducho použiť na akomkoľvek štandardnom webhostingu v rámci webovej aplikácie.

5.2 Databáza MySQL

MySQL je jedným z najpoužívanejších relačných databázových systémov (RDBMS). Relačný databázový systém ukladá dáta do samostatných tabuliek, čo je oveľa rýchlejšie a flexibilnejšie oproti ukladaniu dát do jedného veľkého skladového priestoru. Pre všetky operácie sa používa jazyk SQL, ktorý je najbežnejším štandardizovaným jazykom pre prístup k databázam. MySQL je Open Source projekt a má duálnu licenciu – pre komerčné projekty s vnorenou databázou je potrebné zakúpiť komerčnú licenciu, pre všetky ostatné projekty je možné použiť licenciu GNU GPL (ktorá je zdarma).

MySQL je veľmi rýchly a spoľahlivý systém. Dosahuje výborné výsledky pri prehliadaní veľkých tabuliek a je rokmi overený v systémoch náročných na výkon. [9]

6. Rečová databáza

Pre realizáciu syntetizátora máme k dispozícii databázu s mužským hlasom nahratú na KTL. Databáza pozostáva zo zvukového súboru, v ktorom sú v slede za sebou umiestnené vyhovorenia jednotlivých difón a z textového (tzv. indexovacieho) súboru, v ktorom sa nachádzajú logicky usporiadané záznamy o začiatočných a koncových vzorkách jednotlivých difón vo zvukovom súbore.

Keďže v databáze nie sú zachytené všetky difóny, ktoré je možné v slovenskom jazyku zostaviť, nachádzajú sa v nej okrem difón aj všetky existujúce fonémy. V prípade požiadavky na difónu, ktorá sa v databáze nenachádza, sa táto umelo vytvorí spojením dvoch polovíc príslušných foném.

6.1 Formát indexovacieho súboru

Formát súboru je nasledovný: [10]

- pre fonémy sú na každom riadku súboru tabulátormi oddelené hodnoty:
    - *fonéma* (príslušná fonéma)
    - *offset1* (bajt, na ktorom začínajú záznamy o difónach príslušnej skupiny v indexovacom súbore)
    - *offset2* (vzorka, na ktorej sa začínajú difóny príslušnej skupiny vo zvukovom súbore)
    - *left* (vzorka, na ktorej sa začína príslušná fonéma vo zvukovom súbore)
    - *right* (vzorka, na ktorej končí príslušná fonéma vo zvukovom súbore)
    - *middle* (vzorka, ktorá sa použije ako stred pre prípad, že sa potrebná difóna bude vytvárať z dvoch foném)
- pre difóny sú na každom riadku súboru tabulátormi oddelené hodnoty:
    - *ph1* (prvá fonéma v difóne)
    - *ph2* (druhá fonéma v difóne)
    - *left* (vzorka, na ktorej sa začína príslušná difóna v danej skupine difón)
    - *right* (vzorka, na ktorej sa končí príslušná difóna v danej skupine difón)

6.2 Transformácia indexovacieho súboru na databázové tabuľky

Opísaný formát nie je priveľmi vhodný pre priame použitie v aplikácii jednak z hľadiska zložitosti jeho použitia a tiež z hľadiska výkonu aplikácie. Súbor sme preto rozdelili na fonémy a difóny do dvoch MySQL databázových tabuliek *db_phonemes* a *db_diphones*. Pre možnosť rozšíriť v budúcnosti aplikáciu o viac rečových databáz, sme pridali tabuľku *db_files*, ktorá obsahuje informáciu o názve zvukového súboru.

Dotazy na vytvorenie tabuliek v jazyku SQL:

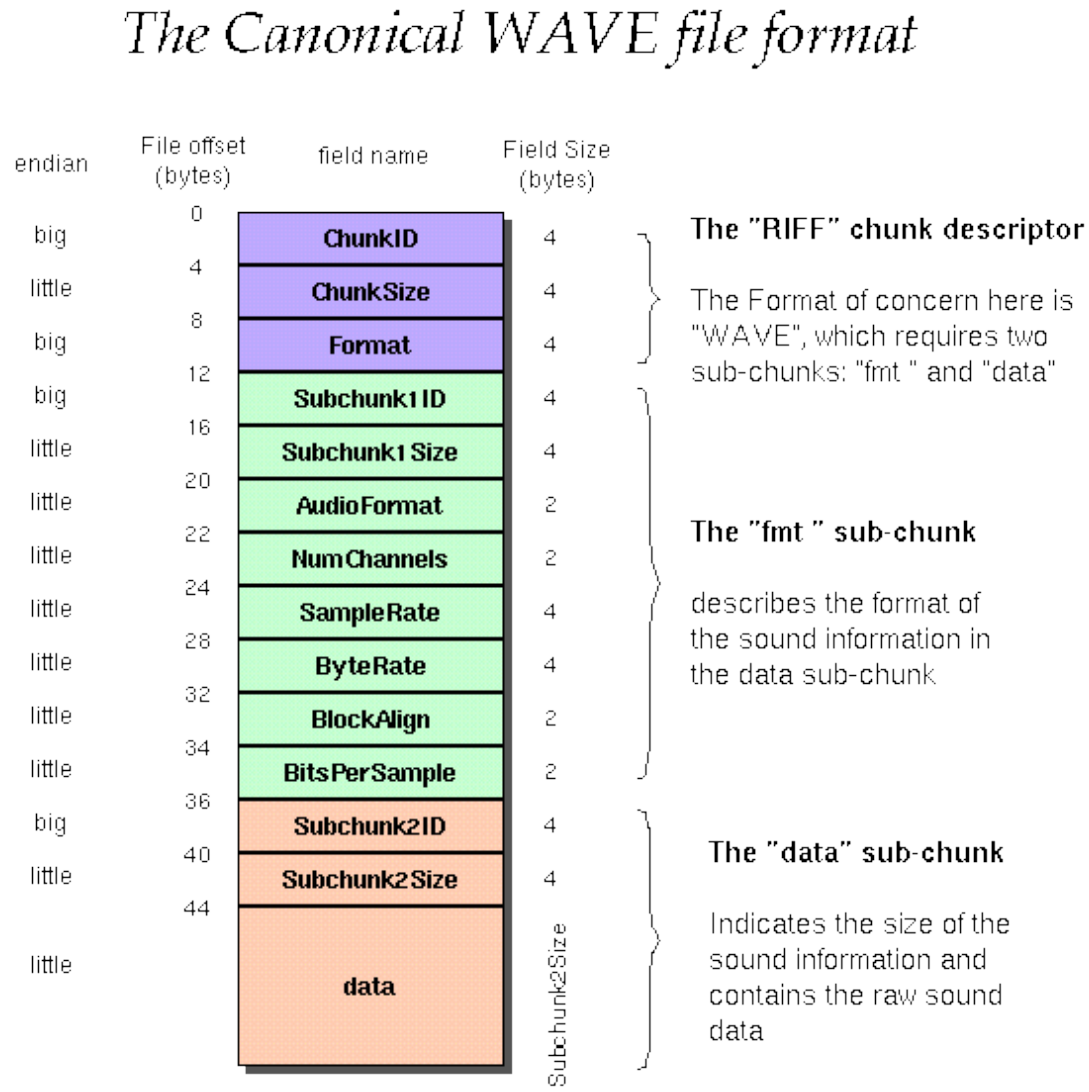
```
CREATE TABLE `db_phonemes` (
`sampa` varchar(4) NOT NULL,
`db_id` varchar(2) NOT NULL,
`offset` int(7) NOT NULL,
`pos_left` int(6) NOT NULL,
`pos_right` int(6) NOT NULL,
`pos_middle` int(6) NOT NULL,
```

```
PRIMARY KEY (`sampa`,`db_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `db_diphones` (
`sampa1` varchar(4) NOT NULL,
`sampa2` varchar(4) NOT NULL,
`db_id` int(2) NOT NULL,
`pos_left` int(6) NOT NULL,
`pos_right` int(6) NOT NULL,
`pos_middle` int(6) NOT NULL,
PRIMARY KEY (`sampa1`,`sampa2`,`db_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

CREATE TABLE `db_files` (
`db_id` int(2) NOT NULL AUTO_INCREMENT,
`title` varchar(30) NOT NULL,
`filename` varchar(30) NOT NULL,
PRIMARY KEY (`db_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Nadväznosť jednotlivých polí v tabuľkách *db_phonemes* a *db_diphones* na pôvodné polia v indexovacom súbore je zrejmá. Stĺpec *offset* obsahuje hodnotu poľa *offset2*, keďže pole offset1 stratilo vďaka použitiu databázy význam. Stĺpec *middle* v tabuľke *db_diphones* sme pridali až dodatočne, kvôli jeho potrebe pri postprocessingu vygenerovaných zvukových súborov. Obsahuje číslo vzorky, na ktorej sa nachádza hranica foném v rámci danej difóny.
Tabuľka *db_files* obsahuje názvy súborov pre jednotlivé rečové databázy. Fonémy a difóny v tabuľkách *db_phonemes* a *db_diphones* sú na konkrétnu rečovú databázu viazané vzdialeným kľúčom *db_id*.

7. Formát WAVE PCM

Súborový formát WAVE (viď. *obr.10*) je podmnožinou Microsoft RIFF špecifikácie, ktorá slúži na ukladanie multimediálnych dát.



Obr.10: Kánonický formát WAVE súboru [11]

Hlavička súboru má veľkosť 44 bajtov, zvyšok tvoria dáta. Jednotlivé polia hlavičky obsahujú nasledovné údaje: [11]

ChunkID	obsahuje text „RIFF“ v ASCII zápise (0x52494646 big-endian)
ChunkSize	dĺžka súboru v bajtoch počnúc týmto blokom (celková dĺžka - 8)
Format	obsahuje text „WAVE“ v ASCII zápise (0x57415645 big-endian)
Subchunk1ID	obsahuje text „fmt “ v ASCII zápise (0x666d7420 big-endian)
Subchunk1Size	16 pre PCM
AudioFormat	1 pre PCM, iné hodnoty indikujú určitú formu kompresie
NumChannels	Mono = 1, Stereo = 2, atď.
SampleRate	8000, 44100, atď.
ByteRate	SampleRate * NumChannels * BitsPerSample / 8

BlockAlign	NumChannels * BitsPerSample / 8
BitsPerSample	počet bitov na jednu vzorku
Subchunk2ID	obsahuje text „data“ v ASCII zápise (0x64617461 big-endian)
Subchunk2Size	veľkosť bloku Data v bajtoch

Blok Data obsahuje samotné dáta v little-endian zápise. 8-bitové vzorky sú reprezentované unsigned bajtmi s hodnotami od 0 do 255. 16-bitové vzorky sú reprezentované celými číslami s hodnotami od -32768 do 32767.

8. Algoritmus syntézy

Proces syntézy pozostáva z troch základných častí:

- fonetický prepis vstupného textu do SAMPA abecedy,
- vytvorenie postupnosti rozsahov vzoriek, ktoré budeme vyberať z rečovej databázy,
- výber jednotlivých vzoriek a ich spojenie do výstupného zvukového súboru.

Ešte pred prvým krokom urobíme na vstupnom texte niekoľko úprav pre dosiahnutie vyššej kvality syntézy:

- nahradenie nealfanumerických znakov,
- nahradenie niektorých cudzích znakov za čo možno najpodobnejšie slovenské znaky,
- prepis čísloviek na textové vyjadrenie.

8.1 Fonetický prepis vstupného textu do SAMPA abecedy

SAMPA (*Speech Assessment Methods Phonetic Alphabet*) je fonetická abeceda zrozumiteľná pre počítač. Pozostáva zo symbolov IPA (*International Phonetic Alphabet*) namapovaných na ASCII znaky a na rozdiel od iných podobných projektov nejde o výtvor jediného autora, ale o výsledok práce výskumníkov z mnohých krajín. SAMPA symboly sú medzinárodne štandardizované [12].

Pred prepisom rozdelíme vstupný text na samostatné slová. Samotný prepis potom prebieha pre každé slovo v dvoch krokoch:

- vyhľadaním slova v slovníku Ábela Kráľa,
- v prípade, že sa slovo v slovníku nenachádza, prepis podľa LTS pravidiel.

8.1.1 Slovník Ábela Kráľa

V slovníku Ábela Kráľa sa nachádza 45 290 slovenských slov a im prislúchajúci fonetický prepis do SAMPA abecedy.

8.1.2 Pravidlá pre fonetický prepis

Pokiaľ je vyhľadávanie v slovníku neúspešné, fonetický prepis slova urobíme na základe LTS (*letter-to-sound*) pravidiel pre slovenčinu. Tieto pravidlá sú zapísané vo forme binárneho CART stromu, ktorého koreňmi sú vstupné písmená a listami sú zodpovedajúce fonémy. Ortoepický prepis je potom jednoduché sledovanie kontextu písmena podľa otázok reprezentovaných uzlami stromu. [13]

8.2 Vytvorenie postupnosti rozsahov vzoriek

Algoritmus nájdenia rozsahov vzoriek rečovej databázy:

- Pre každé dve susediace fonémy vo fonetickom prepise je potrebné nájsť hranice vzoriek pre príslušnú difónu vo zvukovom súbore rečovej databázy.
- Ak sa hľadaná difóna v rečovej databáze nenachádza, nahradíme ju spojením druhej polovice prvej fonémy a prvej polovice druhej fonémy.

8.3 Vytvorenie výstupného zvukového súboru

Na základe rozsahov vzoriek získaných v predchádzajúcom kroku sa načítajú vzorky zo súboru obsahujúceho rečovú databázu. Tieto sa potom pospájajú a zapíšu do výstupného zvukového súboru.

9. Realizácia syntetizátora v jazyku PHP

Samotný syntetizátor pozostáva z niekoľkých tried:

- *lts_parser* – prevod LTS pravidiel z jazyka Lisp do formátu použiteľného v aplikácii
- *transcript* – predspracovanie vstupného textu a prepis do SAMPA abecedy
- *phonetic* – rodičovská trieda triedy *transcript* zabezpečujúca prepis prostredníctvom LTS pravidiel

- *sampa2index* – zistenie rozsahov vzoriek na výber z rečovej dtabázy
- *sampa2wav* – výber vzoriek z rečovej databázy a ich pospájanie do výstupného súboru

Tieto triedy využívajú triedu *mysql* ako rozhranie pre prístup k databáze a triedu *wave* ako rozhranie pre prácu so zvukovými súbormi.
Keďže pracujeme s kódovaním UTF-8, v ktorom majú základné znaky veľkosť 1 byte a rozšírené znaky veľkosť 2 byty, musíme pre prácu s reťazcami používať rozšírenie *Multibyte String*.

9.1 Predspracovanie vstupného textu

Predspracovanie vstupného textu je zabezpečené metódou *preprocessing* triedy *transcript*. V budúcnosti je preto veľmi jednoducho rozšíriteľné upravením tejto metódy.
Nahradenie nealfanumerických znakov zabezpečuje PHP funkcia *preg_replace* volaná s regulárnym výrazom definovaným v časti pre deklaráciu premenných triedy. Aktuálne nahrádza všetky nealfanumerické znaky a tieto sú syntetizátorom ignorované. V budúcnosti by sa pomocou neho dala generovať informácia pre úpravu prozódie.
Nahradenie cudzích znakov za príslušné slovenské znaky je riešené jednoduchým nahradením hľadaného znaku za zadefinovaný slovenský znak. Nahradzované znaky sú tiež definované v časti pre deklaráciu premenných triedy.
Prepis čísloviek zabezpečuje metóda *rewrite_numbers* pomocou PHP funkcie *preg_replace_callback*, ktorá volá metódu *rewrite_general_numbers*. Aktuálne je pomocou tejto metódy možné prepísať čísla do veľkosti rádovo stoviek miliónov. Úpravmi metódy je možné pridať prepisy pre radové číslovky, roky a pod. Viacnásobné volanie funkcie *preg_replace_callback* však môže ovplyvniť rýchlosť procesu predspracovania.

9.2. Fonetický prepis do SAMPA abecedy

Dáta slovníka Ábela Kráľa sme vložili do MySQL tabuľky *dictionary*, vďaka čomu je prehľadávanie slovníka veľmi rýchle. Napomáha tomu aj vhodná indexácia polí tabuľky. Proces vyhľadávania v slovníku sa vykonáva volaním metódy *find_in_dictionary* triedy *transcript*.

Dotaz na vytvorenie tabuľky pre slovník:

```
CREATE TABLE IF NOT EXISTS `dictionary` (
  `word` varchar(25) NOT NULL,
  `sampa` varchar(50) NOT NULL,
  PRIMARY KEY (`word`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

LTS pravidlá sú napísané v jazyku Lisp a pre priame použitie v programe sú nevhodné. Aby sme ich mohli aplikovať v syntetizátore, vytvorili sme parser pre ich transformáciu do multidimenzionálneho poľa, s ktorým sa dá v PHP jednoducho pracovať. Prvým parametrom konštruktora triedy *lts_parser* je adresa súboru s LTS pravidlami v jazyku Lisp. Druhým parametrom je adresa súboru, do ktorého sa pravidlá uložia pretransformované do poľa (spracovaného PHP funkciou *serialize*), ktoré sa používa v aplikácii. Odbúraním potreby parsovania pravidiel počas behu programu sa šetrí výpočtová náročnosť a veľmi významne sa skracuje čas potrebný na prepis.

Vďaka existencii parsera je aplikovanie prípadných zmien v LTS pravidlách veľmi jednoduché – postačuje vytvoriť inštanciu triedy *lts_parser* a zmeniť adresu súboru s pravidlami v konfiguračnom súbore syntetizátora.

Prepis pomocou LTS pravidiel je spracovaný v triede *phonetic*, ktorá je rodičovskou triedou triedy *transcript*.

9.3 Vytvorenie postupnosti rozsahov vzoriek

Proces nájdenia rozsahov vzoriek je spracovaný v triede *sampa2index*. Pre každé dve susediace fonémy sa hľadá záznam v tabuľke *db_diphones*. Pokiaľ záznam existuje, do výstupného poľa sa uloží pozícia začiatočnej a koncovej vzorky rozsahu (polia *pos_left* a *pos_right*, ku ktorým je nutné pripočítať prílušnú hodnotu *offset* z tabuľky *db_phonemes*). Pri neexistujúcom zázname sa uložia rozsahy (polia pos*_left*, *pos_middle* a *pos_right*) častí príslušných foném z tabuľky *db_phonemes* – prvá fonéma od *pos_middle* po *pos_right*, druhá od pos*_left* po *pos_middle*.

9.4 Vytvorenie výstupného zvukového súboru

Výstupom syntetizátora je zvukový súbor vo formáte WAVE PCM, ktorý musí spĺňať zadanú štruktúru. Tento je vytvorený pomocou triedy *index2wav*, ktorá používa k práci so zvukovými súbormi triedu *wave*. Pomocou metódy *read* sú najskôr načítané všetky vzorky v rozsahoch

získaných z výstupu triedy *sampa2index*. Tiero sú následne metódou *write* zapísané do výsledného zvukového súboru.

9.4.1 PHP trieda *wave*

Trieda *wave* má dve základné úlohy:

- načítať obsah zvukového súboru do poľa s číselným vyjadrením jednotlivých vzoriek,
- zapísať pole s číselným vyjadrením vzoriek do zvukového súboru.

Takáto reprezentácia nám umožní jednoduchšie spracovanie zvukových súborov a aplikáciu operácii pri ďalšom post-processingu. Trieda je vzoru singleton, keďže nám postačuje jediná inštancia počas celého behu programu.

Dostupné metódy:

- *get_instance()* – vracia jedinú inštanciu triedy *wave*
- *read(názov súboru, začiatočná vzorka, koncová vzorka)* – načíta obsah súboru od začiatočnej po koncovú vzorku a vracia pole s číselnou reprezentáciou vzoriek
- *write(názov súboru, vzorky)* – vytvorí zvukový súbor zodpovedajúci číselnej reprezentácii vzoriek vo vstupnom poli

V konfiguračnom súbore je možné nastaviť nasledovné parametre, podľa formátu zvukového súboru rečovej databázy:

- počet kanálov (1 = mono, 2 = stereo),
- vzorkovaciu frekvenciu (v Hz),
- počet bitov na vzorku.

10. Ďalšie spracovanie vytvoreného zvukového súboru

Na výstupnom zvukovom súbore vykonávame ďalšie úpravy (*postprocessing*):

- zrýchlenie alebo spomalenie vyhovorenia textu,
- úprava hlasitosti výstupného súboru.

V rámci týchto úprav tiež rozkladáme vstupný text na slabiky. Táto metóda bola implementovaná pre možnosť na napojenie na existujúce GUI naprogramované vo Flashi, ktoré sa ale nerealizovalo.

10.1 Rozklad vstupného textu na slabiky

Pre rozklad slov na slabiky platia v slovenčine 3 pravidlá:

- spoluhláska medzi dvoma samohláskami patrí nasledujúcej slabike (ry-ba),
- pri dvoch spoluhláskach medzi dvoma samohláskami je predel medzi nimi (met-la),
- pri viacerých spoluhláskach je prvá v prvej, ostatné v druhej slabike (zaj-tra).

Rozdelenie slov na slabiky zabezpečuje metóda *syllabify* triedy *postprocess*.

10.2 Úprava rýchlosti reči

Zrýchlenie je zabezpečené vkladaním opakujúcich sa periód pomedzi periódy znelých hlások. Spomalenie naopak odobratím periód zo znelých hlások. Dĺžku neznelých hlások upravovať nevieme.

Rozsah periódy identifikujeme vďaka indexovaciemu súboru pre peaky v rečovej databáze. Pre jednoduchšiu prácu s týmito dátami sme ich pomocou metódy *process_file* triedy *postprocess* vložili do MySQL tabuľky *db_peaks*.

Dotaz na vytvorenie tabuľky:

```
CREATE TABLE IF NOT EXISTS `db_peaks` (
  `position` int(10) NOT NULL,
  `db_id` int(2) NOT NULL,
  PRIMARY KEY (`position`,`db_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

10.3 Úprava hlasitosti výstupného súboru

Úpravu hlasitosti zabezpečuje trieda *volume*. Pri zavolaní metódy *change_volume* s parametrom, koľkonásobne sa má hlasitosť zvýšiť, prenásobí každú vzorku zvukového súboru daným koeficientom.

Keďže pri 16-bitovom zvukovom súbore nadobúdajú hodnoty vzoriek rozsah od -32768 do 32767, je potrebné zabrániť pretečeniu nastavením vyššej hodnoty. Táto metóda ošetruje pretečenie nastavením hodnoty na maximálnu možnú, vďaka čomu môže prísť pri vysokých koeficientoch k skresleniu.

11. Použitie syntetizátora vo webovej aplikácii

Pre demonštráciu možností syntetizátora sme vytvorili jednoduchú aplikáciu (súbor *index.php*), ktorej základom je formulár s tromi vstupnými poliami (viď. *obr.11*). Používateľ zadá vstupný text, zrýchlenie a zmenu hlasitosti. Výstupom sú dva súbory – prvý je zosyntetizovaný text bez ďalších úprav, v druhom je aplikovaná zmena rýchlosti a hlasitosti.

Obr.11: Aplikácia demonštrujúca možnosti syntetizátora

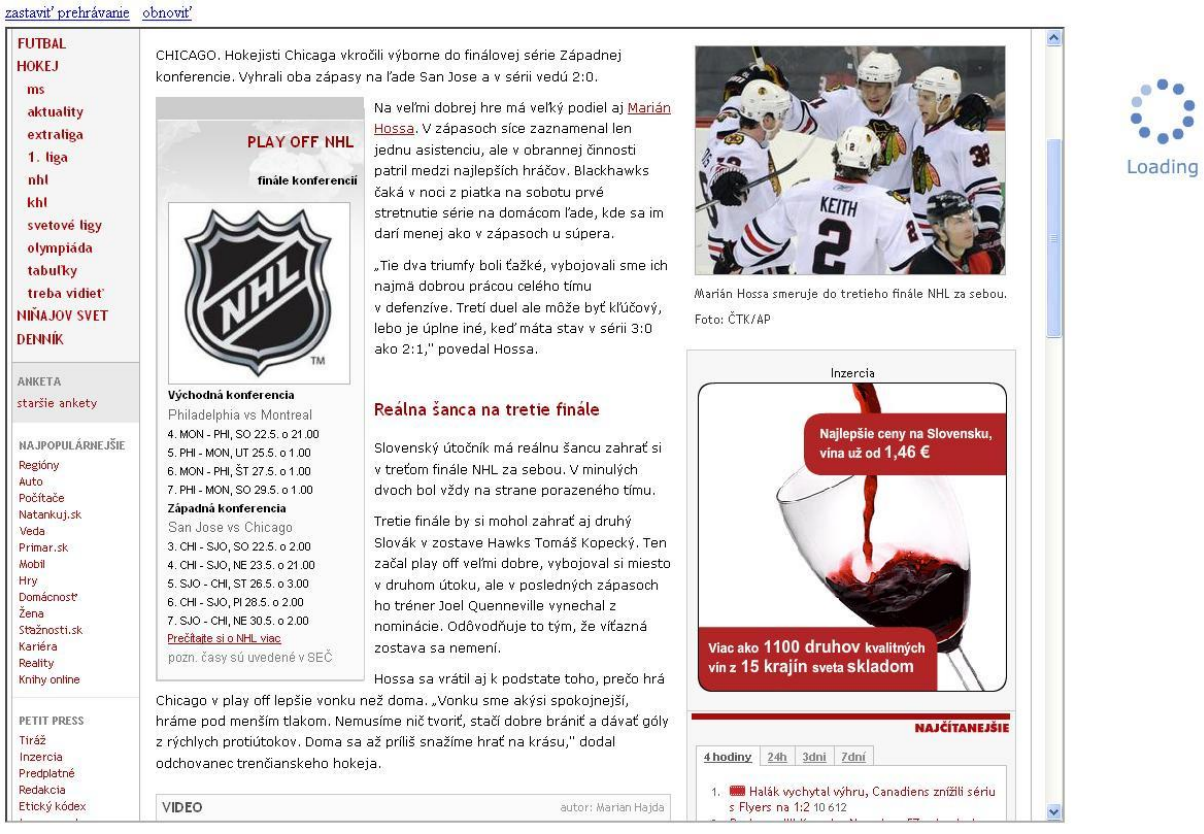
11.1 Použitie v reálnej webovej aplikácii

Pre simulovanie použitia v reálnej aplikácii sme vytvorili aplikáciu čítajúcu texty článkov na portáli www.sme.sk (súbor *sme.php*).

Používateľovi je v menšom rámci zobrazená úvodná stránka portálu, s ktorou môže pracovať rovnako, ako keby sa nachádzal na samotnom portáli. Akonáhle zobrazí podstránku s článkom, začne aplikácia čítať text článku. Syntetizovanie textu prebieha kvôli náročnosti po častiach – jednotlivých odsekoch článku. Pre dosiahnutie tohto efektu používame technológiu AJAX (*Asynchronous JavaScript and XML*), vďaka ktorej sa na server (syntetizátor) posielajú dáta na syntetizovanie postupne, zatiaľ čo užívateľ si hneď môže stránku prezerať a počúvať zosyntetizované úvodné časti textu. Pokiaľ stránka v niekorom momente čaká na odpoveď syntetizátora, používateľ je o tom informovaný zobrazením ikonky (viď. *obr.12*).

Keďže prehrávanie zvuku v prehliadači prostredníctvom JavaScriptu je veľmi problematické, použili sme na tento účel program SoundManager napísaný vo Flashi. [15]

Flash nedokáže pracovať so súbormi vo formáte *wave*, preto je nutné konvertovať súbor do formátu *mp3*, k čomu používame program LAME (*LAME Ain't an MP3 Encoder*). Tento program spúšťame z príkazového riadku použitím PHP funkcie *exec*. [16]



Obr. 12: Použitie syntetizátora v aplikácii čítajúcej články portálu www.sme.sk

11.2 Testovanie aplikácie

Testovanie prvej aplikácie pozostávalo zo zadávania náhodných reťazcov do vstupného poľa a kontroly výstupov. Vďaka vkladaniu mnohých rozmanitých slov sa nám podarilo nájsť a odstrániť výnimky v delení slov na slabiky napríklad v slove *podstata* (text obsahujúci *ds*, ktoré sa číta ako *c*), alebo *foxtrot* (text obsahujúci *x*, ktoré sa číta ako *ks*).

V upravovaných súboroch je pri spomalení a zrýchlení čítania textu počuteľné značné zníženie kvality. Tento nedostatok pripisujeme absencii hodnôt nesúcich informáciu o polohe prechodu medzi fonémami v rámci difóny (pole *pos_middle*) v indexovacom súbore.

Zmena hlasitosti je pri nízkych hodnotách koeficientu bez počuteľných skreslení. Pri vyšších hodnotách (okolo 20) začína kvalita výrazne klesať.

Aplikáciu čítajúcu texty článkov sme testovali náhodným výberom článkov na portáli www.sme.sk. Pri tomto testovaní sa ukázali najväčšie nevýhody realizácie syntetizátora v jazyku PHP. Napriek vyladeniu nastavenia servera (neobmedzený čas pre beh skriptu a dostatočne veľký limit na spotrebu pamäte počas behu skriptu) a samotnej aplikácie (asynchrónne HTTP volania syntetizátora s menšími blokmi textu) trvá vytvorenie výstupného súboru príliš dlhú dobu. Naviac väčšina prehliadačov takúto komunikáciu nezvláda veľmi dobre a často sa stáva, že aplikácia „zamrzne“. Najkomfortnejšie prehliadanie sme dosiahli pri použití prehliadača Mozilla Firefox verzie 3.6.3. Aj tu sú však odozvy medzi jednotlivými blokmi textu príliš dlhé, preto by bolo nasadenie v reálnom webovom projekte len ťažko uskutočniteľné .

Záver

Výsledkom tejto práce je difónový syntetizátor realizovaný kompletne v jazyku PHP, čo je nielen v slovenských ale aj v zahraničných pomeroch pomerne unikátne riešenie. Syntetizátor je funkčný a v rámci práce sme vytvorili dve ukážky jeho implementácie do webových aplikácií.

Pre reálnu možnosť nasadenia potrebuje tento program ešte množstvo vylepšení, počnúc fázou predspracovania textu (kvalitný prepis čísloviek a skratiek), cez samotný proces syntézy (optimalizácia času behu skriptu a spotreby pamäte) a dodatočnom spracovaní generovaného zvukového súboru. Program je vhodne rozdelený do niekoľkých základných objektov, kde každý objekt sa stará o jeden z krokov syntézy, takže pridávanie ďalších funkčností by malo byť pomerne jednoduché.

Napriek tomu, že jazyk PHP spĺňa všetky predpoklady pre realizáciu difónového syntetizátora, táto úloha je náročná výkonovými obmedzeniami prostredia, v ktorom aplikácia beží. Aj algoritmicky menej náročné časti kódu je potrebné značne optimalizovať, aby bol syntetizátor z hľadiska času behu skriptu vôbec použiteľný. Pri tvorbe tohto syntetizátore je mnoho takýchto optimalizácií použitých, ale napriek tomu časy syntézy nie sú veľmi uspokojujúce. Vyvstáva preto otázka, či je vhodné s týmto syntetizátorom ďalej pracovať a implementovať do neho vylepšenia, ktoré sú pre nasadenie v reálnych systémoch nevyhnutné, alebo radšej venovať pozornosť implementáciám v iných „rýchlejších“ programovacích jazykoch a radšej pre ne vytvárať rozhranie pre možnosť použitia v PHP.

Zoznam použitých skratiek

TTS	Text-to-speech
NLP	Natural Language Processing
DSP	Digital Signal Processing
PHP	PHP: Hypertext Preprocessor
RDBMS	Relational Database Management System
SQL	Structured Query Language
KTL	Katedra telekomunikácií
SAMPA	Speech Assessment Methods Phonetic Alphabet
LTS	Letter-to-sound
CART	Classification and Regression Tree
VODER	Voice Operating Demonstrator
PAT	Parametric Artificial Talker
OVE	Orator Verbis Electris
PSOLA	Pitch-synchronous overlap-and-add
CTS	Content-to-speech
LAME	Lame Ain’t an MP3 Encoder

Použitá literatúra

[1] LEMMETTY S.: Review of Speech Synthesis Technology, Helsinki University of Technology, 1999

[2] PSUTKA J.: Mluvíme s počítačem česky, Academia, 2006

[3] SZONDY D.: The Voder, http://www.davidszondy.com/future/robot/voder.htm [cit. 2010-05-20]

[4] ROZINAJ G. a iní: Úloha výskumu a vývoja Inteligentné rečové komunikačné rozhranie štátneho programu Budovanie informačnej spoločnosti – D 1.3 – Modul syntézy reči (Analýza súčasného stavu a návrh riešenia), Košice, jún 2004

[5] DUTOIT T.: An Introduction to Text-to-Speech Synthesis, Springer, 2001

[6] HUANG. X, ACERO A., HON X.-W.: Spoken Language Processing, Prentice Hall, 2001

[7] PSUTKA J.: Komunikace s počítačem mluvenou řečí, Academia, 1995

[8] ACHOUR M. a iní: PHP Manual, http://www.php.net/manual/en/index.php [cit. 2010-05-20]

[9] AXMARK D. a iní: MySQL Reference Manual, http://dev.mysql.com/doc/refman/5.1/en/ [cit. 2010-05-20]

[10] MICHALKO O.: Difónový syntetizátor slovenskej reči – Diplomová práca, Bratislava, máj 2007

[11] SAPP C.: WAVE PCM soundfile format, https://ccrma.stanford.edu/courses/422/projects/WaveFormat/ [cit. 2010-05-20]

[12] WELLS. J.: SAMPA computer readable phonetic alphabet http://www.phon.ucl.ac.uk/home/sampa [cit. 2010-05-20]

[13] CERŇAK M.: Využitie objektívnych meraní kvality pri korpusovej syntéze reči – Dizertačná práca, Bratislava, júl 2004

[14] SVOBODA V.: Sluchové ústrojí – střední ucho, http://www.audiocity.cz/clanek.php?id=86 [cit. 2010-05-20]

[15] SCHILLER S.: SoundManager 2 – A Javascript Sound API supporting MP3 and MPEG4 audio, http://www.schillmania.com/projects/soundmanager2/ [cit. 2010-05-20]

[16] HEGEMANN R. a iní: About LAME, http://lame.sourceforge.net/about.php [cit. 2010-05-20]