VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

JUKEBOX - ROZHRANÍ PRO PŘEHRÁVÁNÍ HUDBY

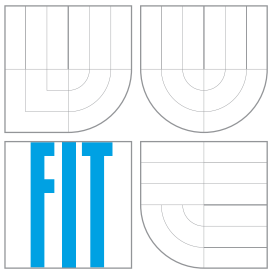
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER ŠOŠOVIČKA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

JUKEBOX - ROZHRANÍ PRO PŘEHRÁVÁNÍ HUDBY

JUKEBOX - MUSICAL RECORD USER INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER ŠOŠOVIČKA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. PAVEL ZEMČÍK

BRNO 2009

Abstrakt

Tato práce se zabývá problematikou tvorby grafických uživatelských rozhraní a problematikou přehrávání MP3. Dále se zabývá analýzou a návrhem programu pro přehrávání hudby pro komerční účely („Jukebox“). V práci je popsáno, jak jsou jednotlivé části aplikace implementovány.

Abstract

This bachelor thesis deals with GUI development and MP3 player task. Additionally, the thesis presents analysis and design of musical record interface application for commercial use (“Jukebox“). This thesis also describes how individual parts of the application are implemented.

Klíčová slova

mp3, audio, wxWidgets, jukebox

Keywords

mp3, audio, wxWidgets, jukebox

Citace

Peter Šošovička: Jukebox - rozhraní pro přehrávání hudby, bakalářská práce, Brno, FIT VUT v Brně, 2009

Jukebox - rozhraní pro přehrávání hudby

Prohlášení

Prohlašuji, že jsem tuto bakalářsku práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Peter Šošovička
19. mája 2009

Poděkování

Děkuji tímto svému vedoucímu, panu Zemčíkovi, za odbornou pomoc a cenné rady při řešení této bakalářské práce.

Obsah

Kapitola 1

Úvod

Počítače sa v dnešnej dobe stali neoddeliteľnou súčasťou bežného života. S príchodom hudby v digitalizovanej podobe sa medzi súčasti využívania počítačov zaradilo, v súčasnosti veľmi populárne, prehrávanie hudobných súborov.

Cieľom tejto bakalárskej práce bolo, ako je uvedené v zadaní, navrhnúť a vytvoriť softvér pre prehrávanie hudby pre komerčné účely „Jukebox“. Tento softvér, by mal pomocou grafického rozhrania, umožniť užívateľovi zvoliť si a následne prehrať hudobný súbor podľa vlastného výberu z databázy hudobných súborov.

Jedným z hlavných dôvodov, prečo som si vybral túto prácu, bola moja túžba dozvedieť sa, ako sa vytvára podobný typ softvéru, ale i skutočnosť, že sa jedná, o pre mňa veľmi zaujímavú oblasť využitia počítačov. Pri výbere práce taktiež zavážilo to, že v budúcnosti by som sa chcel venovať projektom z podobnej oblasti, založených najmä na interakcii s užívateľom.

V nasledujúcej kapitole je popísaná história tvorby grafických užívateľských rozhraní, využiteľné komunikačné kanály pri jej tvorbe a problematika návrhu grafických užívateľských rozhraní z pohľadu programátora a užívateľa. Ďalej sa v kapitole nachádza prehľad nástrojov pre tvorbu grafických užívateľských rozhraní. Druhá kapitola rozoberá spôsoby kompresie zvuku a prezentuje históriu formátu MP3. Ďalej sa venuje spôsobu kompresie technológie MP3, formátu MP3 súboru a predstavuje dostupný softvér pre prehrávanie hudby. Ďalšia kapitola popisuje analýzu a návrh riešenia. Návrh štruktúry softvéru je rozdelený do troch častí a to na návrh užívateľského rozhrania, návrh jadra prehrávača a návrh databázy. Predposledná kapitola sa venuje implementačnej časti a sú v nej prezentované techniky použité pri tvorbe aplikácie. Rozoberá jednotlivé časti aplikácie a popisuje implementáciu grafického rozhrania, implementáciu prehrávacieho jadra aplikácie a tvorbu databázového modulu. Záverečná kapitola popisuje dosiahnuté výsledky a predkladá nápady, ako by bolo možné túto prácu v budúcnosti vylepšiť.

Kapitola 2

Tvorba grafických užívateľských rozhraní

Táto kapitola preberá problematiku tvorby grafických užívateľských rozhraní a to od jej histórie, cez využiteľné komunikačné kanály pri jej návrhu a tvorbe po problematiku návrhu grafických užívateľských rozhraní. V závere predstavuje niektoré nástroje umožňujúce ich tvorbu. V tejto kapitole sú zmienené fakty, relevantné pre túto bakalársku prácu.

2.1 História tvorby GUI

História grafických užívateľských rozhraní (Graphical User Interface – GUI) je takmer taká stará, ako história počítačov samotných. Myšlienka ovládať počítače pomocou grafického rozhrania sa zrodila v hlave Vannevara Busha už v roku 1945, kedy predstavil koncept analógového počítača *memex*[1]. Avšak prvé grafické užívateľské rozhranie sa podarilo zrealizovať až o niekoľko rokov neskôr. V roku 1973 bol vo firme Xerox vyvinutý osobný počítač *Alto*. Boli v ňom použité i niektoré princípy, ktoré poznáme z dnešných počítačov, ako napríklad pracovná plocha, odpojiteľná klávesnica alebo trojtlačidlová myš. Hoci bol tento počítač na tú dobu prevratný, pre svoju vysokú cenu nezaznamenal väčší úspech. Ďalší pokrok v histórii grafických užívateľských rozhraní nastal takmer po 10 rokoch, keď vo firme Apple Computer začiatkom roku 1983 predstavili osobný počítač *Apple Lisa*. Jeho súčasťou bol i operačný systém, ktorý používal princípy ako roletové menu, posuvníky a dokonca kôš na ploche, ktorým sa realizovalo vymazávanie súborov. I keď ani tento počítač nezaznamenal veľký marketingový úspech, dá sa považovať za predka dnešných grafických užívateľských rozhraní. O rok neskôr, teda v roku 1984 bol uvedený počítač *Apple Macintosh*, ktorý sa vďaka svojej cene stal veľmi úspešným. V tom istom roku sa do hry dostáva ďalší hráč na poli grafických užívateľských rozhraní – *X Window System*. Ten bol vyvinutý na MIT v rámci projektu *Athena*. Jedná sa o hlavné jadro grafického subsystému operačných systémov unixového typu. Koncom roku 1985 uzrel svetlo sveta i prvý grafický produkt spoločnosti Microsoft nazvaný *Windows 1.0*. Súčasne s vývojom grafických operačných systémov sa vyvíjali aj aplikácie s grafickým prostredím. Ďalší zlom nastal v 90. rokoch minulého storočia, keď sa grafické užívateľské rozhrania začali rozširovať do najrôznejších typov zariadení ako notebooky, mobilné telefóny alebo vreckové počítače. V súčasnej dobe sa vývoj užívateľských rohraní dostal i do tretieho rozmeru, vďaka vzniku

[1] Viac informácií na http://en.wikipedia.org/wiki/Memex

3D grafických rozhraní ako *Compiz* (X Window System), *Quartz Extreme* (Mac OS X) či *Aero* pre Windows Vista. [2, 7]

2.2 Komunikačné kanály v GUI

Pre komunikáciu medzi človekom a strojom pripadajú do úvahy len tie komunikačné kanály, ktoré spĺňajú určité podmienky. Musia byť schopné informácie prenášať v reálnom čase, reprodukovateľne a pokiaľ možno nezkreslene. Podľa smeru prenosu informácie sa informačné kanály dajú rozdeliť na dva typy – pre prenos informácie od stroja k človeku alebo pre prenos informácie od človeka k stroju.
Pre prenos informácie od stroja k človeku je možné využiť nasledujúce kanály:

- Obraz (zrak) – obraz je považovaný za najvýhodnejšie médium pre prenos informácie od stroja k človeku. Dôvodom je hlavne veľmi vysoká informačná priepustnosť a možnosť „náhodného prístupu“ pozorovateľa k informáciam obsiahnutým v obraze. V dnešnej dobe je realizovaný pomocou klasických (CRT) alebo LCD monitorov, ale taktiež i pomocou projektorov, ktoré obraz prenášajú na plátno. Takisto existujú i zariadenia schopné prenášať obraz trojrozmerne, avšak v súčasnosti sú skôr v štádiu vývoja. K zobrazovacím zariadeniam tiež patria tlačiarne či plotre, ktoré prenášajú obraz na papier alebo iné médium.

- Zvuk (sluch) – pre prenos menšieho množstva informácií od stroja k človeku je výhodný i zvuk, napríklad ako doplnenie obrazu. Nevýhodou oproti obrazu je nižšia priepustnosť a „sériový prístup“. Jednou z výhod je skutočnosť, že zvuk môže na seba upozorňovať.

- Hmat – doposiaľ je využívaný výnimočne, avšak je perspektívny pre budúce aplikácie v spojení s virtuálnou realitou. A to vo forme buď hmatovej alebo silovej spätnej väzby. V súčasnej dobe je využívaný hlavne ako prostriedok pre komunikáciu nevidiacich so strojom.

- Čuch, chuť – v súčasnosti tieto kanály nie sú pre komunikáciu použiteľné, pretože stav súčasnej chémie a techniky ešte ani zdaleka neumožnuje syntetizovať chuť ani vôňu v reálnom čase. Avšak v prípade, že by to bolo v budúcnosti zrealizovateľné, mohol by sa tento komunikčný kanál uplatniť ako doplnok pre virtuálnu realitu.

Pre prenos informácie opačným smerom, teda od človeka k stroju je možné využiť nasledujúce komunikačné kanály:

- Pohyb (hmat) – najobvyklejším prostriedkom pre predávanie informácií človekom do stroja je takmer od počiatku histórie počítačov klávesnica. A to v najrôznejších podobách. Klávesnice bývajú často doplňované mechanickými polohovacími zariadeniami, ako je napríklad myš. V súčasnosti sa objavujú i nové zariadenia pre zadávanie informácií, ako sú dotykové displeje či dokonca rukavice snímajúce pohyb ruky. Možné je však skonštruovať množstvo zariadení podobného typu.

- Zvuk (reč) – tento spôsob komunikácie, teda komunikácia pomocou reči, je veľmi perspektívna. V blízkej budúcnosti môžme očakávať, že sa tento spôsob komunikácie stane silnou konkurenciou klávesniciam. Dokonca, v niektorých typoch aplikácií je

možné, že ju tento typ komunikácie aj nahradí. Doposiaľ však nie je tento typ komunikácie prakticky použiteľný, pretože kvôli množstvu jazykov a dialektov ale i iných dôvodov, nie je uspokojivo vyriešené porozumenie ľudskej reči na strane stroja.

- Obraz (gestá) – realizácia predania informácie pomocou gesta tela, ruky, prípadne mimiky je pravdepodobne veľmi vzdialená. Napriek tomu, niektoré aplikácie rozpoznávania obrazu pre komunikáciu človeka so strojom sú reálne už dnes. V budúcnosti by sa tento spôsob komunikácie mohol stať súčasťou systému virtuálnej reality.

Informácie v tejto kapitole boli prebraté z [9].

2.3 Návrh GUI

Návrh grafických užívateľských rozhraní je možné rozdeliť na dve časti. Jedna časť je pohľad programátora na aplikáciu, to sa týka najmä operačného systému pre ktorý aplikáciu vytvára a aké prostriedky budú použité pre vývoj aplikácie. Naopak, z pohľadu užívateľa je zaujímavejšie ako sa aplikácia ovláda, prípadne aký má vzhľad.

Pohľad programátora

Pri návrhu grafického užívateľského rozhrania aplikácie je potrebné vedieť, pre aký operačný systém je aplikácia vyvíjaná. Ak je v požiadavkách zadané, že stačí, ak aplikácia bude fungovať len pod konkrétnym operačným systémom, môžeme pre tvorbu grafického rozhrania využiť knižnice, ktoré ponúka operačný systém. Pre operačný systém Windows je jednou z týchto knižníc *Windows API*. Ak je požadované, aby aplikácia bola funkčná pod rôznymi operačnými systémami, je pre tvorbu GUI možné použiť niektorú z knižníc ponúkajúcich možnosť prekladu pod rôznymi operačnými systémami. Príkladom takejto knižnice je *wxWidgets*. Ďalšou možnosťou, je použitie niektorého z takzvaných interpretovaných jazykov, pri ktorých sa zdrojový kód neprekladá do spustiteľného súboru, ale len do medzikódu. Tento kód je potom za behu interpretovaný virtuálnym strojom. Medzi tieto interpretované jazyky patrí napríklad *Java*. Jednou z možností návrhu grafického rozhrania je využitie takzvaných návrhových programov. Tie umožňujú navrhnúť si rozloženie okien, menu, ovládacích panelov a iných prvkov. Potom je týmto programom vygenerovaný zdrojový kód. Programátor sa tak môže starať len o programový kód, kedže GUI už má navrhnuté. Nevýhodou týchto programov je, že vygenerovaný kód môže byť neprehľadný. Medzi takéto programy patrí napríklad *wxDesigner*, dostupný pre knižnicu wxWidgets.

Pohľad užívateľa

Z pohľadu užívateľa, sú pri návrhu grafického rozhrania aplikácie dôležitejšie iné aspekty ako akým spôsobom je aplikácia implementovaná. Dôležitejšie je dbať na to, pre aký typ užívateľa je aplikácia vyvíjaná. Ak je aplikácia vyvíjaná pre bežného užívateľa – laika, je vyžadované skôr prehľadné prostredie, bez zbytočných panelov a okien, ktorým by nemusel tento užívateľ ihneď porozumieť a tým by sa ovládanie aplikácie stalo pre neho nezrozumiteľným. Na druhej strane, ak je aplikácia vyvíjaná pre pokročilého užívateľa, je vhodné použitie rôznych panelov, ktorými môže nastavovať funkčnosť či konfiguráciu aplikácie. Avšak v prípade, že dopredu nevieme či našu aplikáciu bude používať laik alebo pokročilý užívateľ, je potrebné návrh grafického rozhrania tomu prispôsobiť. Teda zvoliť rozumný

kompromis toho, aby bolo rozhranie prehľadné pre oba typy užívateľov. Ďalšou možnosťou je vytvorenie rôznych verzií programu napríklad pre laika či pokročilého užívateľa.

2.4 Prehľad nástrojov pre tvorbu GUI

QT

Qt je jedna z dvoch najpopulárnejších multiplatformových knižníc pre vytváranie programov s grafickým užívateľským rozhraním. Qt spoločne s GTK+ nahradila starší Motif. Medzi najznámejší softvér využívajúci Qt patrí: prostredie KDE, webový prehliadač Opera, Google Earth, Skype. Táto knižnica je vyvíjaná pre štvoricu platforiem – X Window, Mac OS, Windows a Embedded. Medzi jej výhody patrí – dostupnosť knižníc pre C++ či malá zmena kódu pri kompilovaní na rôznych platformách. Avšak má i niekoľko nevýhod, napríklad viacero typov licencií.

wxWidgets

wxWidgets je open source multiplatformový widget toolkit. Je to knižnica základných elementov pre tvorbu grafického rozhrania. WxWidgets umožňuje skompilovať a spustiť program na rôznych počítačových platformách s minimálnymi alebo žiadnymi zmenami kódu. To zahŕňa systémy ako MS Windows, Mac OS, Linux/Unix, OpenVMS a OS/2. Knižnica je implementovaná v C++. wxWidgets je najlepšie popísaný ako natívny toolkit, čiže namiesto napodobňovania grafiky prvkov, používa natívne grafické prvky na podporovaných platformách. Výhodami toolkitu wxWidgets sú knižnice prístupné pre C++, voľné použitie aj pre komerčne účely či malá zmena kódu pri kompilovaní na rôznych platformách. Bližšie informácie o toolkite wxWidgets v [5].

Kapitola 3

Prehrávanie MP3 súborov

Táto kapitola sa zaoberá problematikou prehrávania hudobných súborov MP3. Postupne v nej nájdeme históriu MP3, princípy kompresie zvuku, princíp kompresie MP3, formát MP3 súboru a prehľad súčasného softvéru pre prehrávanie MP3. V tejto kapitole sú zmienené fakty, relevantné pre túto bakalársku prácu.

3.1 Kompresia zvuku

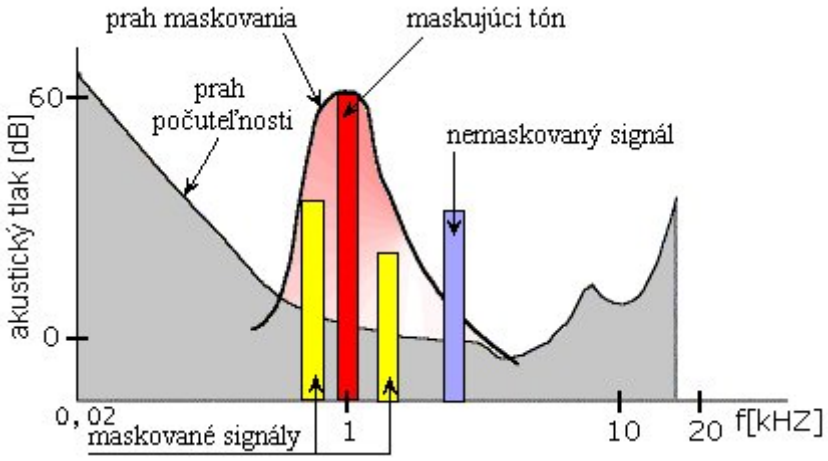
Každý užívateľ, ktorý niekedy počúval hudbu na počítači sa nepochybne stretol so skomprimovanými hudobnými súbormi. Nespracované hudobné súbory, napríklad súbory z audio CD sú veľmi veľké, pretože obsahujú omnoho viac informácií ako dokáže ľudský mozog spracovať. Napríklad, ak sú dva zvukové záznamy veľmi podobné a blízko pri sebe, mozog dokáže spracovať len jeden z nich. Ak sú dva zvuky rozdielne, ale jeden z nich je hlasnejší, mozog nie je schopný vnímať ten tichší. I keď je to veľmi individuálne, vo všeobecnosti, ľudské ucho nedokáže zachytiť frekvencie pod 20 Hz a takisto nedokáže zachytiť ani frekvencie nad 20 kHz. Štúdiom týchto sluchových javov sa zaoberá psychoakustika. Vďaka týmto štúdiam mohli vzniknúť rôzne metódy kompresie zvuku. Tie umožňujú hudobnú nahrávku skomprimovať na $\frac{1}{10}$ až $\frac{1}{12}$ povôdnej nahrávky pri akceptovateľnej úrovni kvality.

Experimenty ukázali, že ľudské ucho má 24 frekvenčných pásiem. Frekvencie v týchto, takzvaných kritických pásmach, sú ľudským uchom ťažko rozlíšiteľné. Predpokladajme, že v audio signále je prítomný prevládajúci signál. Potom tento silnejší signál je prahovou hodnotou a udáva hranicu, pod ktorou sú zvuky v rovnakom kritickom pásme nepočuteľné. Tým tento silnejší signál maskuje slabšie (obr. 3.1). Tento jav vo frekvenčnej oblasti je známy ako frekvenčné maskovanie (simultaneous masking) a bol spozorovaný v rámci výskumu kritických pásiem.

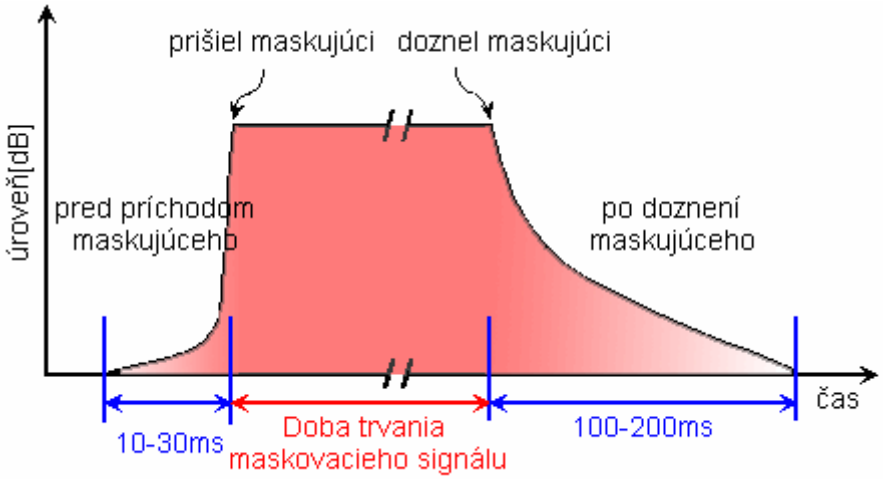
Časové maskovanie (temporal masking) sa odohráva v časovej oblasti. Silnejšia tónová zložka (tzv. masker) zamaskuje slabšiu zložku, ak sa objaví počas krátkeho časového intervalu. Maskovací signál zamaskuje slabšie signály pred (pre-masking) a po (post-masking) tomto signáli. Pre-masking zvyčajne trvá 50 ms, zatiaľ čo postmasking trvá od 50 ms do 300 ms, v závislosti na sile a dobe trvania maskovacieho signálu (obr. 3.2) [4].

3.2 História MP3

V roku 1987 sa sformoval výskumný spolok medzi Erlangen-Nuremberg University a Fraunhofer Institute for Integrated Circuits. Za podpory fondov Európskej únie EUREKA vznikol



Obrázok 3.1: Frekvenčné maskovanie (zdroj [6]).



Obrázok 3.2: Časové maskovanie (zdroj [6]).

projekt EU147, pre prenos digitálneho zvuku. V tom istom roku sa vo výskumnom centre Fraunhofer začal vývoj na projekte audio kódovania s vysokou kvalitou a nízkym dátovým tokom. Výskumná skupina vedená profesorom Heinzom Gerhäuserom vytvorila funkčný real-time kodek pre LC-ATC (Low Complexity Adaptive Transform Coding) algoritmus. Do tejto doby existoval len ako počítačová simulácia a mohol byť testovaný len na obmedzenom množstve audio materiálu. Real-time kodek umožnil testovanie LC-ATC v reálnych podmienkach a viedol tak k výraznej optimlaizácii algoritmu. V roku 1988 bola pod vedením oragnizácie ISO založená pracovná skupina nazvaná MPEG (The Moving Picture Experts Group). Zaoberala sa vývojom štandardov pre kompresiu zvuku a obrazu. V roku 1989 dokončil Karlheinz Brandenburg, nazývaný tiež ako „otec mp3“ doktorskú dizertáciu na tému Optimálne kódovanie vo frekvenčnej doméne (OCF), predstavujúcu niekoľko charaktristík eventuálnej technológie MP3. Softvérová časť bola pod vedením profesora Gerhäusera vyvíjaná Bernhardom Grillom. V tejto fáze bola technológia OCF rožšírená o systém schopný kódovať audio signál pri 64 kbit/s. Týmto mohlo byť dosiahnuté prenosu hudby v reálnom čase po telefónnej linke. Spoločnosti Fraunhofer bol udelený patent na

technológiu v apríli 1989. V roku 1991 je za prispenia Hannover University a firiem AT & T a Thomson prezentovaný nový výkonný kodek nazvaný ASPEC, ako výsledok ďalšieho vylepšovania OCF. V roku 1992 dokončil MPEG prvý kompresný ISO štandard MPEG-1. V jeho audio časti boli špecifikované 3 kompresné schémy (Layer 1, Layer 2 a Layer 3). Čoskoro sa pre svoju výkonnosť stal Layer 3 populárnym pre ukladanie hudby na relatívne malých pevných diskoch a prenos hudobných súborov cez internet. Od roku 1994 je Layer 3 tiež súčasťou audio špecifikácie MPEG-2 štandardu. V roku 1995 vznikol názov „mp3“, keď .mp3 bola výskumníkmi vo Fraunhoferi jednohlasne zvolená, ako prípona pre súbory MPEG Layer 3 [1, 8].

3.3 Kompresia MP3

Technológia MP3 môže použiť jednu z dvoch kompresných techník pre dosiahnutie redukcie veľkosti pre neskomprimované audio súbory – stratovú alebo bezstratovú.

Stratová kompresia je taká, ktorá z audio záznamu odstraňuje tie signály, ktoré ľudské ucho nepočuje. Avšak, to je len jedna časť z celého procesu, ktorý zahŕňa omnoho viac práce. Tento proces môže byť rozdelený na niekoľko menších úloh:

- Signál sa rozdelí na malé časti, nazývané rámce (frames), kde každý z nich trvá zlomok sekundy.
- Analyzuje sa signál a na celom spektre počuteľných frekvencií sa určí, ako musia byť bity rozdelené pre dosiahnutie čo najlepšieho výsledku zakódovania audio súboru.
- Určí sa bitová rýchlosť a vypočíta sa maximálny počet bitov pridelený pre každý rámec. Napríklad, ak je súbor zakódovaný na 128 kbps, je určená horná hranica množstva dát uložených v rámci. Tento krok rozhoduje o tom, koľko dát sa zo zdrojového hudobného súboru použije i v skomprimovanom súbore.
- Frekvencia je pre každý rámec porovnaná s matematickým modelom ľudskej psychoakustiky, ktorá je v kodeku uložená ako referenčná tabuľka. Z tohto modelu sa dá určiť, ktorá frekvencia musí ostať presne zachovaná, pretože sa dá vnímať ľudským uchom, a ktorá môže byť zredukovaná alebo úplne vyradená z výsledného súboru, pretože aj tak ju nedokáže ľudské ucho zachytiť.
- Tento tok dát prejde procesom Huffmanovho kódovania, ktorý skomprimuje nadbytočné informácie. Huffmanovo kódovanie[1] nepracuje na princípe psychoakustického modelu, ale vykonáva kompresiu tradičnejším spôsobom. Tento krok neodstraňuje žiadne dáta, len ich skomprimuje. Na základe toho môžme vidieť, že proces kompresie MP3 je dvojprechodový. Prvý prechod je kompresia na základe psychoakustiky. Druhý prechod skomprimuje prebytočné dáta tradičnou komprimačnou metódou.
- Nakoniec sú rámce skompletizované do výsledného dátového toku, s hlavičkou predchádzajúcou každému rámcu. Hlavička obsahuje metadáta špecifické pre daný rámec (informácie v kapitole 3.4).

Bezstratová kompresia je odlišná tým, že po dekompresii sú dáta v skomprimovanom súbore identické ako dáta v originálnom súbore. Jednou z najznámejších bezstratových komprimačných metód je komprimačná metóda ZIP [3].

[1] Viac informácií je možné nájsť na http://en.wikipedia.org/wiki/Huffman_coding

3.4 Formát MP3

Každý MP3 súbor je rozdelený na menšie časti, nazývané rámce. Každý rámec obsahuje 1152 zvukových vzorkov a trvá 26 ms. To znamená, že rýchlosť je približne 38 fps. Každý rámec sa skladá z dvoch častí, ktoré obsahujú 576 vzorkov. Pretože dátový tok určuje veľkosť každého vzorku, zvyšovaním dátového toku, zväčšujeme veľkosť rámca. Veľkosť taktiež závisí na vzorkovacej frekvencii podľa nasledovného vzorca:

$$\frac{144 * \textit{dátový tok}}{\textit{vzorkovacia frekvencia}} + \textit{výplň}\ [\text{bajty}]$$

kde výplň je špeciálny bit umiestnený na začiatku rámca. V niektorých rámcoch je používaný na to, aby presne vyhovovali požiadavkám na dátový tok. Ak je výplňový bit nastavený, rámec je vyplnený jedným bajtom. Veľkosť rámca je vždy celočíselná, napr. $144 * 128000/44100 = 417$ [4]

Štruktúra rámca

Rámec je uložený v dátovej štruktúre nazývanej *frame* a pozostáva z piatich častí – Header, CRC, Side Information, Main Data a Ancillary data ako je ukázané v tabuľke 3.1

Header	CRC	Side Information	Main Data	Ancillary Data

Tabuľka 3.1: Schéma rámca (zdroj [4]).

Hlavička

Hlavička (Header) má veľkosť 32 bitov a obsahuje synchronizačné slovo spolu s popisom rámca.

Sync word (12 bitov)	Je na začiatku každej MP3 a na začiatku každého rámca a odlišuje hlavičku od ostatných dát. Všetky bity musia byť nastavené na 1.
ID (1 bit)	Špecifikuje MPEG verziu, nastavený bit znamená, že rámec je kódovaný MPEG-1, nenastavený bit znamená MPEG-2. Niektoré štandardy používajú len 11 bitové synchronizačné slovo a 2 bity pre ID.
Layer (2 bity)	Určuje typ kompresnej schémy (Layer 1, Layer 2, Layer 3).
Protection bit (1 bit)	Ak je nastavený, použije sa CRC súčet.
Bitrate (4 bity)	Určujú dátový tok pre rámec.
Frequency (2 bity)	2 bity určujúce vzorkovaciu frekvenciu.
Padding bit (1 bit)	Výplňový bit.
Private bit (1 bit)	Určený pre zvláštne spúšťače.
Mode (2 bity)	Špecifikujú použitý kanálový mód (Stereo, Joint Stereo, Dual Channel, Single Channel).
Mode extension (2 bity)	Tieto 2 bity sa používajú len v Joint Stereo móde a špecifikujú použitú metódu.
Copyright bit (1 bit)	Ak je bit nastavený, znamená to, že kopírovanie obsahu je nelegálne.
Home bit (1 bit)	Nastavený bit znamená, že rámec je na originálnom médiu.
Emphasis (2 bity)	Vyjadruje, či sú zvýraznené frekvencie nad cca. 3,2 kHz.

Kontrolný súčet

Toto pole (16 bajtov) existuje, iba ak je *Protection bit* v hlavičke nastavený a umožňuje kontrolovať „citlivé dáta“ na prenosové chyby. Citlivé dáta sú definované štandardom ako bity 16 až 31 v hlavičke a v *side info*. Ak sú tieto dáta chybné, poškodia celý rámec napriek tomu, že chyba v dátach skresluje len časť rámca. Poškodený rámec potom môže byť buď stlmený alebo nahradený predchádzajúcim rámcom.

Ďalšie informácie

Ďalšie informácie sú uložené v štruktúre nazvanej Side information. Je to časť rámca pozostávajúca z informácií potrebných pre dekódovanie hlavných dát. Veľkosť závisí na kanálovom móde. Ak je to jednokanálový mód (Single Channel), veľkosť je 17 bajtov, ak nie, veľkosť je 32 bitov. Jednotlivé časti poľa *side info* sú prezentované v tabuľke 3.2 a popísané nižšie.

Main data begin	Private bits	scfsi	Side info granule 0	Side info granule 1

Tabuľka 3.2: Side info (zdroj [4]).

Veľkosti jednotlivých polí sú odlišné pre mono mód a pre ostatné módy:

Main data begin (9 bitov)	Technika nazývaná „bit reservoir“ umožňuje používať voľné miesto v oblasti *main data* nasledujúcimi rámcami. Aby sme boli schopný s určitosťou zistiť kde začínajú dáta, dekodér musí prečítať hodnotu *main data*. Táto hodnota je záporný ofset od prvého bajtu *synchronizačného slova*. Ak je táto hodnota rovná 0, potom dáta začínajú ihneď za *side info*.
Private bits (5bitov pre mono mód, 3 bity pre ostatné)	Vyhradené bity, ktoré ISO nebude v budúcnosti využívať.
scfsi (ScaleFactor Selection Information) (4 bity pre mono mód, 8 bitov pre ostatné)	Rozhoduje o tom či je rovnaký *scalefactor* prenášaný pre *side info granule 0* aj pre *side info granule 1* alebo pre každý *side info granule* zvlášť.
side info granule 0, side info granule 1	Tieto 2 časti rámca majú rovnakú štruktúru, sú zložené z menších častí a obsahujú jednotlivé informácie pre každú „granule“ osobitne.

Hlavné dáta

Dátová časť rámca pozostáva z 2 častí – scalefactors a bitov Huffmanovho kódovania:

scalefactors	účelom „scalefactorov“ je zredukovať šum. Ak sú vzorky v jednotlivých pásmach „scalefactora“ správne nakalibrované, šum je zamaskovaný úplne.
bity Huffmanovho kódovania	Obsahuje bity Huffmanovho kódovania. Informácia o tom, ako ich odkódovať sa nachádza v časti *Side info*.

Vedľajšie dáta

Vedľajšie dáta sú voliteľné a počet bitov nie je explicitne daný. Vedľajšie dáta sa nachádzajú za bitmi Huffmanovho kódovania a určujú kam bude ukazovať *Main data begin* nasledujúceho rámca.

3.5 Softvér pre prehrávanie hudby

V dnešnej dobe existuje množstvo prehrávačov pre prehrávanie hudby. Táto podkapitola ponúka prehľad známych a populárnych prehrávačov pre rôzne operačné systémy.

Winamp

Winamp je jeden z najpopulárnejších audio prehrávačov pre operačný systém Windows. Jeho vývoj začal v roku 1997. V súčasnosti je vo verzii 5, ponúka veľké množstvo pluginov a možnosť zmeny vzhľadu pomocou skinov. Je dostupný v troch nespoplatnených verziách (Lite, Full, Bundle) a štvrtej (Pro), spoplatnenej verzii. Vo všetkých verziách okrem verzie Lite umožňuje i prehrávanie video súborov. Viac informácií je uvedených na domovskej stránke www.winamp.com

foobar2000

foobar2000 je multimediálny audio prehrávač pre operačný systém Windows XP a vyšší. Je známy vďaka svojmu veľmi prispôsobiteľnému rozhraniu. Umožňuje pokročilú podporu ID3 tagov. Vďaka systému otvorenej tvorby komponentov, je možné pridávať ďalšie rôzne funkcie. V súčasnosti je dostupný vo verzii 0.9.

Windows Media Player

Windows Media Player je multimediálny prehrávač vyvíjaný spoločnosťou Microsoft, ktorý je štandardnou súčasťou operačného systému Windows. Umožňuje prehrávanie audio a video súborov a ich organizáciu a takisto zmenu vzhľadu pomocu skinov a rožšírenia pomocou pluginov. V súčasnosti je dostupný vo verzii 11.

XMMS

XMMS (X MultiMedia System) je voľne dostupný hudobný prehrávač fungujúci na veľkom počte Linuxových aj Unixových operačných systemov. Je veľmi podobný prehrávaču Winamp a umožňuje rožšírenia pomocou veľkého množstva pluginov. Vývoj programu vo verzii 1.2 bol zastavený a v súčasnosti sa vyvíja verzia 2.

Amarok

Amarok je prehrávač hudby pre unixové systémy. Umožňuje výbornú organizáciu hudobnej zbierky podľa viacerých kriterií (napr. podľa štýlu, albumu, autora), pridávať texty piesní alebo vytvárať vlastné skripty. Takisto je prepojený s populárnou službou *last.fm*. V súčasnosti je vyvíjaný vo verzii 1.4 a verzii 2.0 súčasne. Verzia 2.0 bola vydaná v roku 2008 a priniesla kompletnú zmenu dizajnu prostredia a podporu pre operačné systémy Windows a Mac OS X.

Audacious

Audacious je voľne dostupný audio prehrávač, založený na GTK2 a fungujúci pod Linuxom a inými Unixovými platformami. Je zameraný najmä na audio kvalitu a podporu množstva hudobných kodekov. Audacious ponúka možnosť rožšírenia pomocou pluginov a dobrú podporu pre vývojárov. V súčasnosti je dostupný vo verzii 1.5.

Kapitola 4

Analýza a návrh riešenia

Táto kapitola preberá návrh štruktúry softvéru zariadenia pre prehrávanie hudby pre komerčné účely. Ďalej kapitola v stručnosti rozoberá výhody a nevýhody zvoleného riešenia. Na konci kapitoly je rozobratá analýza možných rozšírení aplikácie v budúcnosti.

4.1 Návrh štruktúry softvéru

Štruktúra softvéru tohoto typu sa dá rozdeliť na 3 časti. A to na časť grafického rozhrania, časť prehrávacieho jadra a databázovú časť. Pri analýze a návrhu štruktúry som vychádzal z návrhu rozhrania už známych zariadení typu „Jukebox“. Ale taktiež som si sám určil niektoré požiadavky, ktoré by mal tento softvér spĺňať. Tieto požiadavaky sú rozobraté v niekoľkých nasledujúcich odstavcoch.

Grafické rozhranie

Z hľadiska grafického rozhrania by mala aplikácia počas celého svojho behu byť spustená v celoobrazovkovom režime. Keďže sa jedná o špecifický typ softvéru, aplikácia bude ovládaná výhradne klávesnicou. Aplikácia by mala mať dva typy menu. Jedno menu pre výber albumov a druhé menu pre výber piesní. Navigácia v týchto menu by mala byť realizovaná pomocou šípok na klávesnici. Taktiež musí byť užívateľovi pomocou klávesnice umožnené prepínať medzi týmito menu a potvrdiť svoj výber. Z hľadiska toho, že vývojové nástroje pre tvorbu grafických užívateľských rozhraní ponúkajú len obmedzené množstvo ovládacích prvkov, v návrhu aplikácie sa počíta s vlastným dizajnom ovládacích prvkov. Ďalej by grafické rozhranie malo byť schopné podať užívateľovi informáciu o tom, koľko kreditov ešte môže použiť (koľko piesní ešte môže vybrať) a koľko piesní sa ešte nachádza v playliste. Takisto by grafické rozhranie aplikácie malo ponúknuť informáciu o aktuálne hranej piesni. Jednou z posledných požiadaviek na aplikáciu, je schopnosť ponúknuť užívateľovi rozhranie, pomocou ktorého by mohol rýchlo vyhľadávať piesne v databázi. To by malo byť umožnené zadávaním textu pomocou klávesnice do okna aplikácie. Poslednou požiadavkou na aplikáciu je, že v prípade, že v playliste sa nenachádza žiadna pieseň, aplikácia zmení svoj stav do „demo“ režimu. V tomto režime sa budú prehrávať náhodne vybraté piesne z databázy. Vhodným nástrojom pre implementáciu týchto prvkov grafického užívateľského rozhrania je *wxWidgets* v spojení s jazykom C++.

Prehrávacie jadro

Prehrávacie jadro by malo byť schopné prehrať najmä MP3 súbory a bolo by vhodné, keby ponúkalo možnosť čítať ID3 tagy. Taktiež je výhodné, ak by malo v rámci svojej implementácie systém, ktorý by umožňoval zistiť pozíciu na ktorej sa v audio súbore aktuálne nachádza a systém, ktorý by upozorňoval na skončenie prehrávania audio súboru. Všetky z týchto požiadaviek spĺňa knižnica *xine*[1], hoci sa jedná o knižnicu pre operačné systémy unixového typu. Okrem iného, je na týchto operačných systémoch veľmi populárna a ponúka celkom dobrú podporu pre vývojárov.

Databázová časť

Keďže aplikácia pracuje s relatívne veľkým množstvom hudobných súborov, je výhodné mať informácie o nich uložené v databáze. Databázový systém použitý v tejto aplikácii by mal byť efektívny a takisto by bolo vhodné, ak by pre svoju funkčnosť nepotreboval databázový server. Veľkou výhodou by bolo, ak by mal príkazy čo najviac podobné dotazovaciemu jazyku SQL a to najmä kvôli veľkej dostupnosti rôznych príkladov. Všetky z týchto podmienok spĺňa databázový systém *SQLite*[2]. Ďalšou nespornou výhodou je aj fakt, že toolkit *wxWidgets* ponúka rozhranie pre tento databázový systém. Toto rozhranie sa nazýva wxSQLite a umožňuje jednoduchšiu implementáciu a prácu s týmto databázovým systémom.

4.2 Výhody a nevýhody zvoleného riešenia

Z hľadiska grafického rozhrania je zvolené riešenie výhodné vďaka niekoľkým aspektom. A to napríklad, že je umožnená pomerne ľahká zmena vizuálnej stránky aplikácie jednoduchou zmenou bitmapy pre pozadie. Ďalšou výhodou je rozmiestnenie ovládacích prvkov podobným spôsobom, ako je to na súčasných zariadeniách tohoto typu, takže sa prostredie stáva pre užívateľa pomerne intuitívne a ľahko sa mu prispôsobí. Tento návrh riešenia ponúka dokonca možnosť textových vstupov od užívateľa a je otvorený ďalším možným vylepšeniam. Taktiež medzi ďalšie výhody zvoleného riešenia, je možné nepochybne zaradiť aj použitie toolktiu *wxWidgets*, vďaka čomu sa dá aplikácia bez väčších zmien preniesť i na ostatné platformy.

V rámci časti prehrávacieho jadra, je výhodou použitie knižnice *xine*. Keďže je knižnica prístupná pre jazyk C++, mala by byť implementácia prehrávacieho jadra do grafického rozhrania bez väčších problémov. Táto knižnica je voľne použiteľná i pre komerčné účely, čo je v tomto type aplikácie nespornou výhodou. Ďalej podporuje veľké množstvo audio i video formátov, čo môže byť výhodné najmä v prípade ďalších rozšírení aplikácie. Nevýhodou zvoleného riešenia, najmä v prípade, že by sme chceli aplikáciu preniesť na iný operačný systém, môže byť to, že knižnica je dostupná len pre operačné systémy unixového typu.

Pri analýze a návrhu databázovej časti aplikácie, bolo uvedené, že by mala aplikácia používať databázový systém *SQLite* a rozhranie *wxSQLite* pre prístup z toolkitu *wxWidgets*. Výhoda zvoleného riešenia je najmä v tom, že je umožnená ľahká prenositeľnosť na rôzne platformy a taktiež, že celá databáza je obsiahnutá v jednom súbore. Vďaka tomuto riešeniu aplikácia nepotrebuje mať spustený databázový server a je umožnená i jednoduchá záloha

[1] Viac informácií na http://www.xine-project.org/home
[2] Viac informácií na http://www.sqlite.org/

dát z databázy. V rámci analýzy neboli zaznamenané žiadne nevýhody pri použití tohoto návrhu oproti iným dostupným databázovým systémom.

4.3 Automatická aktualizácia súborov

Pri návrhu sa ukázalo, že by bolo výhodné, ak by správca hudobných súborov (majiteľ zariadenia) nemusel mať na starosti aktualizáciu hudobnej databázy. Za podmienky, že počítač, na ktorom je spustená aplikácia je pripojený k sieti Internet, je jedným z možných riešení tohoto problému umožniť aplikácii automaticky aktualizovať hudobné záznamy zo servera. Jedným z vhodných spôsobov realizácie tejto vlastnosti, by mohlo byť spustenie skriptu, ktorý sa pripojí na vopred určený server. Následne si tento skript zistí zmeny hudobných súborov na serveri a porovná ich s databázou aplikácie. V prípade, že sa na serveri nachádzajú piesne, ktoré ešte aplikácia nemá v databáze, stiahne nové piesne do vopred určeného adresára. Následne, pri ďalšom spustení aplikácie je databáza aktualizovaná a užívateľovi sú ponúknuté na výber už i nové piesne.

4.4 Bezdrôtová komunikácia s užívateľom

Ďalším zaujímavým rozšírením aplikácie, najmä pre vačšie pohodlie, je ponúknuť užívateľovi možnosť komunikovať s aplikáciou bezdrôtovo. Ako najvhodnejšie riešenie sa zdá použitie technológie Bluetooth. Táto technológia je v súčasnosti veľmi populárna a rozšírená najmä na mobilných telefónoch a PDA prístrojoch. Jedným z možných riešení tohoto rozšírenia, je prenechať úlohu servera na tejto aplikácii. Tento server by obsluhoval požiadavky od klienta. Pre tvorbu serverovej časti aplikácie, pretože sa jedná o aplikáciu pre operačný systém Linux, je výhodné použiť bluetooth stack BlueZ, ktorý má svoje knižnice dostupné i pre jazyk C++. Nevýhodou by bol prenos na operačný systém Windows, kde tento bluetooth stack nie je implementovaný. Na klientskej strane by bola Java aplikácia, určená pre mobilné telefóny, prípadne PDA. Taktiež by bolo vhodné, ak by užívateľ mal možnsť nechať si odoslať túto aplikáciu priamo z rozhrania „jukeboxu“ na svoje mobilné zariadenie. Možnosti použitia v tejto aplikácii sú veľmi široké. V rámci komunikácie s užívateľom, by mu mohla aplikácia ponúknuť zobrazenie aktuálne hranej skladby, vybratie skladby, vyhľadanie piesní v databáze aplikácie či prípadne zobrazenie textu ľubovoľnej piesne uloženej v databáze. To všetko by prebiehalo na displeji mobilného zariadenia užívateľa i niekoľko metrov od zariadenia, bez nutnosti byť v priamom fyzickom kontakte so zariadením.

Kapitola 5

Implementácia

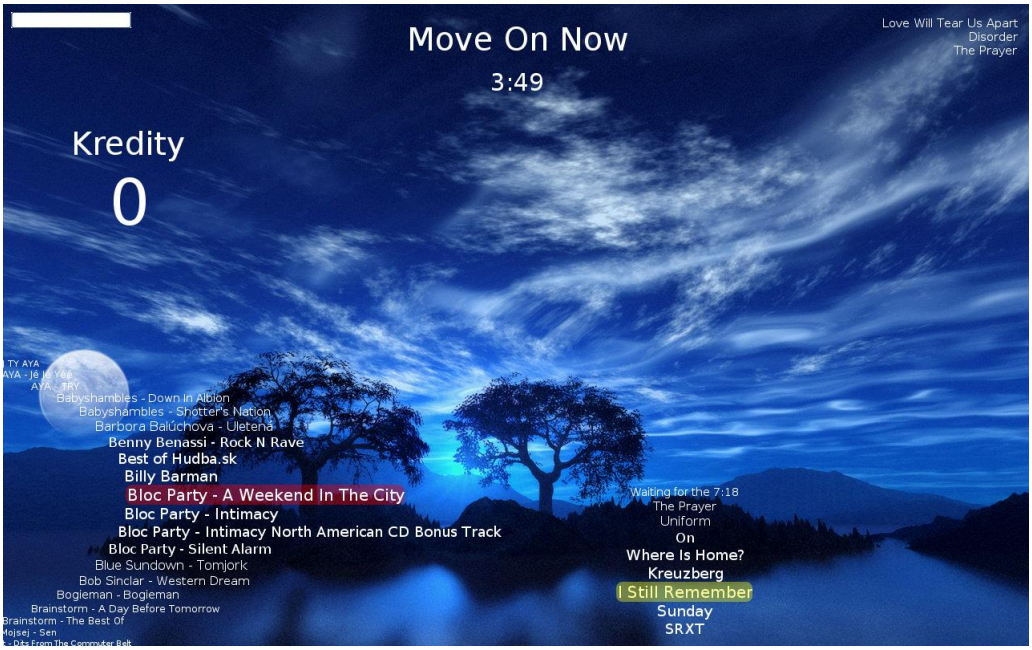
Táto kapitola rozoberá implementáciu aplikácie. Jednotlivé kroky implementácie sa dajú rozdeliť do troch častí a to na implementáciu grafického rozhrania, implementáciu prehrávania a implementáciu databázovej časti. Na konci kapitoly sú stručne rozobrané metódy akými bola aplikácia testovaná.

5.1 Grafické rozhranie

Základ grafického rozhrania tvorí okno `MyFrame` zdedené od základnéj triedy `wxFrame` knižnice *wxWidgets*. V inicializačnej časti okna `MyFrame` sa toto okno nastaví do režimu na celú obrazovku (fullscreen). Keďže celá aplikácia je ovládaná pomocou klávesnice, nevyužívaný kurzor myši je nastavený ako priehľadný. Na pozadí celého okna je umiestnená bitmapa, ktorá tvorí celkový vzhľad aplikácie. Zmenou tejto bitmapy je možné jednoduchým spôsobom meniť vzhľad aplikácie (obr. 5.1). Metóda použitá na vykreslenie grafiky aplikácie je `OnPaint`, ktorá je volaná vždy, keď je vygenerovaná požiadavka na prekreslenie okna. V rámci tejto metódy je zabezpečené aj vykreslovanie pozadia aplikácie. V tejto metóde sú takisto volané vykreslovacie metódy jednotlivých ovládacích panelov. Tým je zaistené, že sa vždy prekreslia všetky potrebné prvky. Avšak kvôli niektorým panelom, ako napríklad panelu na zobrazenie priebehu piesne, je nutné prekreslovať okno častejšie. To je zabezpečené pomocou časovača, ktorý je volaný periodicky každú sekundu a umožní tak zobrazovať aktuálne informácie. Jednotlivé ovládacie prvky umiestnené v okne `MyFrame` sú definované ako samostatné triedy a vytvorenie ich objektov prebieha v konštrukcii okna `MyFrame`. Medzi tieto prvky patria panely pre výber albumov a piesní a informačné panely. V hlavnom okne aplikácie je umiestnený i ovládací prvok `wxTextCtrl`, ktorý umožňuje vstup od užívateľa z klávesnice. Popis implementácie týchto ovládacích prvkov je nižšie.

Panel pre výber albumov

Ako bolo v kapitole 4 uvedené, tento typ aplikácie vyžaduje vlastný návrh dizajnu ovládacích prvkov, takže nebolo možné použiť štandardné ovládacie prvky ako napríklad *ListBox*. Preto som pri implementácii panelu pre výber albumov zvolil vlastný spôsob vykreslovania. Hlavnou metódou pre vykreslenie tohoto panelu, čiže objektu triedy `ArtistControl` je metóda `Paint()`. Základ tvoria dva vektory obsahujúce objekty štruktúry `Album`. Táto štruktúra obsahuje názov albumu a index pod akým je album uložený v databáze. Prvý z týchto vektorov je vždy po spustení naplnený dátami z databázi (všetky albumy uložené



Obrázok 5.1: Grafické rozhranie aplikácie.

v databáze) a druhý vektor je prázdny. Prvý vykreslovaný album je ten, ktorý je ako prvý v prvom vektore. Je to zároveň aj aktuálne vybraný album, čo je znázornené priehľadným zaobleným obdĺžnikom, ďalej sa vykreslujú ostatné albumy z toho istého vektora, pričom kvôli vizuálnemu efektu sa pri každom ďalšom albume zmenšuje veľkosť písma o jeden stupeň. Dáta z prvého vektora sa vykreslujú len ak nie je dosiahnutá hranica obrazovky v ose y. Potom sa vykreslia dáta z druhého vektora. Tieto dáta sa vykreslia nad aktuálne vybraným albumom. Takisto sa mení veľkosť písma a vykreslujú sa dáta len pokiaľ sú viditeľné v ose x, to znamená, že vždy sa vykreslia len údaje viditeľné na obrazovke, nie celý obsah vektorov. Informácia o veľkosti vypisovaného textu je získavaná metódou `GetTextEntent()`, ktorá pre zadaný reťazec vráti jeho výšku a šírku v pixeloch. O presun dát medzi jednotlivými vektormi sa stará metóda `MakeMove()`, ktorá týmto presunom zabezpečuje i samotný posun položiek v menu. Presun dát sa realizuje tým, že dáta zmazané z jedného vektora sú presunuté do vektora druhého. Metóda `GetSelection()` vráti index albumu z databázy pre aktuálne vybranú položku.

Panel pre výber piesní

Vykreslenie ponuky piesní je realizované pomocou metódy `Paint` triedy `SongControl`. Základom je vektor obsahujúci objekty štruktúry `Song`. Ak tento vektor nie je prázdny, znamená to, že sa v databáze našli pesničky priradené k vybranému albumu. Inak je tento vektor naplnený dátami podľa toho aký album má užívateľ označený v paneli pre výber albumov. Ak je na panel pre výber piesní zaostrené, zobrazí sa to vykreslením zaobleného obdĺžnika pod prvou piesňou v ponuke. Počet vykreslených piesní určuje premenná `max_visible`. Veľkosť písma nad a pod označenou piesňou sa kvôli vizuálnemu efektu zmenšuje. Najdôležitejšou metódou pre triedu tohoto panela je metóda `MakeMove()`. Táto metóda je volaná po každom stlačení navigačných kláves (šípka dole, šípka hore). Podľa

typu stlačenej klávesy sa určí, či ide o pohyb v menu smerom dole alebo smerom nahor a následne sa zmenia alebo posunú položky v menu. Takisto má táto metóda na starosti meniť i pozíciu „markera“, čiže prvku, ktorý určuje pieseň, ktorú ma užívateľ práve zvolenú. Tento prvok je znázornený zaobleným obdĺžnikom. Pozícia tohoto prvku je určená hodnotou v premennej `marker_pos` a tento prvok sa pohybuje v rámci menu dynamicky, ako je to možné vidieť na menu v niektorých typoch mobilných zariadení. Metóda `GetSelection()` slúži na vrátenie indexu piesne na ktorú aktuálne ukazuje „marker“.

Informačné panely

Medzi informačné panely patrí panel pre zobrazenie playlistu, zobrazenie priebehu piesne a zobrazenie informácií o počte kreditov a aktuálne hranej skladbe. Panel na zobrazenie playlistu je vykreslovaný metódou `Paint()` a číta údaje z playlistu, ku ktorému pristupuje panel pre výber piesní (pridáva piesne) a vlákno pre prehrávanie (odstraňuje piesne z playlistu). V tejto metóde sa takisto vypisuje aj informácia o aktuálne hranej skladbe, ktorá je čítaná z premennej `actual`. Pre zobrazenie priebehu piesne je použitá metóda `Paint()` triedy `ProgressC`. Táto metóda číta zo štruktúry `ProgressStr` aktuálne dáta o priebehu hranej skladby a prevedie ich do formátu v ktorom zobrazuje čas, ktorý ostáva do konca piesne. Zobrazenie informácie o počte kreditov je pre jednoduchosť vykreslované metódou `Paint()` z okna `MyFrame`.

Ovládanie

Keďže ovládanie aplikácie je výhradne pomocou klávesnice, všetky vstupy od užívateľa sú spracovaváné v metóde `onKeyDown`. Táto metóda spracováva vstupy po stlačení klávesy. Pre jednoduchosť sú na navigáciu použité len 3 klávesy (šípka hore, šípka dole a tabulátor). Po stlačení klávesy s kódom `WXK_UP` (šípka hore) sa zistí na ktorý ovládací panel (albumy/pesničky) je zaostrené a podľa toho sa volá metóda `MakeMove()` konkrétneho panelu, ktorá zaistí posun dát vo vektoroch a tým i posun v zobrazenom menu. Po každom takomto posune je volaná metóda `RefreshRect()`, ktorá zabezpečí prekreslenie okna a zobrazí tak aktuálne dáta. Obdobný postup je použitý i pri stlačení klávesy s kódom `WXK_DOWN` (šípka dole). Metódy zabezpečujúce prepínanie medzi panelmi pre výber albumu a pre výber piesne sú volané po stlačení klávesy `WXK_TAB` (tabulátor). Po stlačení tejto klávesy sa zistí na ktorý z menu panelov je zaostrené a hodnoty premenných určujúce zaostrenie pre každý panel sa zmenia na opačné. Klávesa `WXK_RETURN` (Enter) slúži pre potvrdenie výberu piesne. Ak je zaostrené na paneli s piesňami a premenná `credits`, ktorá uchováva počet kreditov, nie je 0, pridá sa vybraná pieseň do playlistu a odčíta sa 1 kredit z celkového počtu kreditov. Pre simuláciu pridávania kreditov je v aplikácii použitá klávesa `WXK_RIGHT` (šípka vpravo). Po stlačení tejto klávesy je zavolaná metóda `AddCredits()`, ktorá pridá jeden kredit a následne sa obnoví okno, aby bolo možné vidieť aktuálny stav kreditov. Ostatné klávesy, napríklad znaky písmen a číslic sa spracovávajú bežným spôsobom a je tak umožnené, aby ich mohol spracovávať ovládací prvok `text`, slúžiaci na vstup od užívateľa, ktorý je objektom triedy `wxTextCtrl`.

5.2 Prehrávanie MP3

Základom jadra pre prehrávanie MP3 súborov v tejto aplikácii je knižnica *xine*. Jednou zo základných požiadaviek pre správne fungovanie tejto knižnice je, že musí byť spustená vo vlastnom vlákne. Toto vlákno je vytvorené pri konštrukcii okna `MyFrame`. Po vytvorení tohto vlákna je následne spustené. Samotné vlákno má názov `xineThread` a tvorí jadro prehrávania aplikácie. Knižnica *xine*, na ktorej je aplikácia postavená, má vlastný zachytávač udalostí nazvaný `event_listener` v ktorom môže zachytávať rôzne typy správ. V tejto aplikácii je využívaná na zachytenie správy o ukončení prehrávania piesne. Po spustení vlákna prebehne konfigurácia nastavení potrebných pre *xine*. Potom sa program dostane do nekonečnej slučky v ktorej – ak nie je playlist prázdny vyberie sa prvá pieseň z playlistu a absolútna cesta k tejto piesni sa uloží do premennej `playsong`. Následne sa táto pieseň z playlistu odstráni. Ak playlist je prázdny, stav aplikácie sa nastaví do režimu „demo“ (obr. 5.2). Ak je aplikácia v tomto režime znamená to, že pieseň, ktorá bude hrať je zvolená náhodným výberom zo všetkých piesní uložených v databáze. Potom sa aplikácia pomocou funkcií `xine_open()` a `xine_play()` pokúsi otvoriť a spustiť daný súbor, ku ktorému je cesta v premennej `playsong`. Ak sa to podarí dostáva sa do ďalšieho cyklu v ktorom vykonáva svoju činnosť dovtedy, kým sa nezmení stav premennej `running` (skončí pieseň). V tomto cykle je nutné volať metódu `Sleep()` s parametrom určujúcim čas v ms, ktorá zabezpečí, že ostatné veci mimo vlákna budú mať dostatočný čas na vykonanie svojich rutín. V rámci tohoto cyklu sa pomocou funkcie `xine_get_pos_length()` pravidelne zisťuje pozícia v ktorej sa práve pieseň nachádza. Zistené údaje (trvanie piesne a už odohratý čas v milisekundách) sa uložia do štruktúry `ProgressStr`, z ktorej ich čítaju metódy triedy `ProgressC` a zobrazujú užívateľovi. Ak sa aplikácia nachádza v „demo“ režime a z piesne sa už odohrala dopredu zadaná časť, implicitne nastavená na $\frac{1}{3}$, pošle sa správa o ukončení prehrávania skladby a pokračuje sa ďalšou piesňou. Taktiež, ak počas prehrávania v „demo“ režime užívateľ vyberie skladbu, je prehrávanie aktuálne hranej skladby ukončené a pokračuje sa ďalšou skladbou z playlistu. Po odohratí každej piesne je zavolaná funkcia `Refresh()`, na prekreslenie okna, čiže zobrazenie aktuálneho stavu aplikácie.

5.3 Databáza

Vychádzajúc z analýzy v kapitole 4, bol pre túto aplikáciu zvolený databázový systém SQLite. Databázová časť aplikácie je umiestnená v samostatnom module. Celkový návrh databázy počíta i s možnými rozšíreniami pre ďalšie využite v budúcnosti. Do databázy je možné ukladať napríklad cestu k obalu albumu alebo cestu k textu piesne. Základom databázy v tejto aplikácii je trieda `musicdb`.

Práca s databázou

O vytvorenie databázy sa stará metóda `CreateDatabase()`, ktorá vytvorí databázu s názvom `jukebox.sqlite` v adresári, kde sa nachádza aplikácia, ak táto databáza ešte neexistuje. Na prácu s databázou slúžia aj metódy `OpenDatabase()` a `CloseDatabase()`, ktoré zabezpečujú otvorenie/uzavretie súboru s databázou.

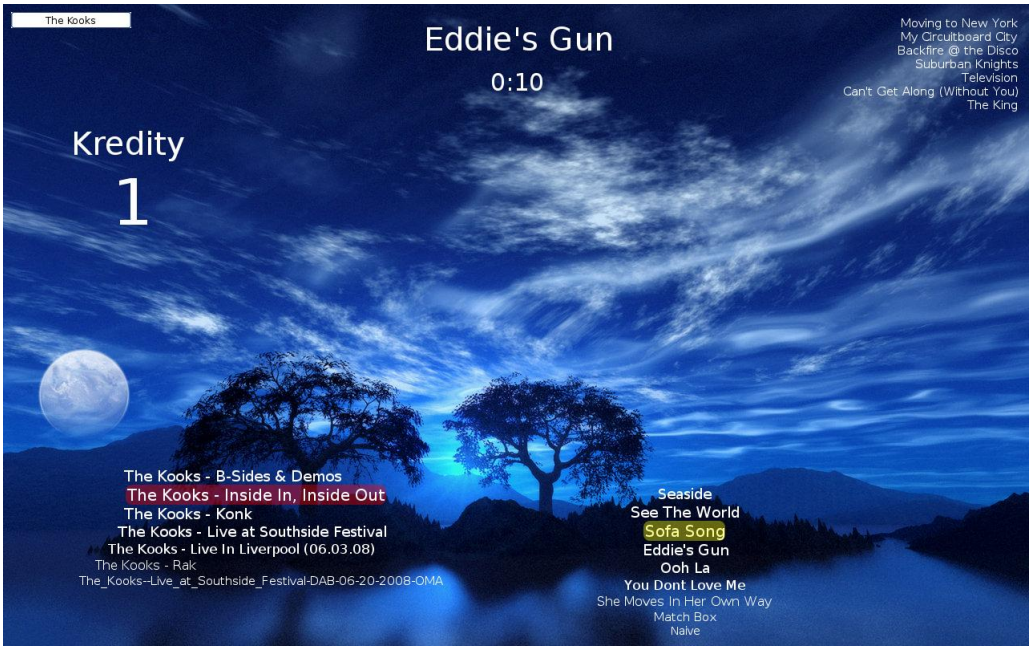
Obrázok 5.2: Aplikácia v „demo“ režime.

Vyhľadávanie v databáze

Pre vyhľadávanie v databáze má trieda `musicdb` niekoľko metód. Metóda `FindAlbums()` vyhľadá všetky albumy v databáze a uloží ich do vektoru pre albumy. Metóda `FindSongs()` vyhľadá pre zadaný index albumu piesne priradené k tomuto albumu a uloží ich do vektoru pre piesne. Ďalšou metódou je `GetSongCount()`, ktorá vráti celkový počet piesní v databáze. Základnou metódou pre vyhľadávanie podľa vstupu od užívateľa je metóda `FindText()`. Táto metóda dokáže spracovať i viacslovné vyhľadávacie názvy. Na spracovanie reťazca, ktorý obsahuje viacero slov slúži funkcia `parse()`, ktorá má ako vstupný parameter zadaný reťazec a pomocou regulárneho výrazu ho rozdelí na jednotlivé slová a vráti ich späť vo vektore. Potom je pre tieto slová vytvorený jeden *SQL* dotaz `SELECT`, ktorý sa odošle databázovému systému. Ten vďaka tomuto dotazu naraz vyhľadá všetky slová zadané užívateľom v tabuľkách pre albumy i piesne. Výsledok vyhľadávania táto metóda vráti vo vektore pre albumy a následne môžu byť zobrazené užívateľovi (obr. 5.3). Metóda `FindPath()` slúži na vyhľadanie kompletnej cesty k piesni, ktorej index bol zadaný ako parameter metódy. V tejto metóde sa musia použiť údaje zo všetkých 3 tabuliek databázy, ktoré sú potom spojené do jedného reťazca určujúceho cestu k súboru. Poslednou metódou, ktorá slúži na vyhľadávanie údajov v databáze je metóda `FindSong()`, ktorá vráti názov piesne pre zadaný index. Tento údaj sa používa pri výpise aktuálne hranej piesne.

Aktualizácia databázy

Na aktualizáciu databázy slúžia metódy `UpdateDatabase()` a `UpdatePlays()`. Metóda `UpdateDatabase()` postupne prejde všetky adresáre uložené v súbore `paths`. Ak sa cesta k tomuto adresáru ešte nenachádza v databázovej tabuľke `paths`, je do nej pridaná. Potom sú postupne vyhľadané názvy všetkých adresárov s albumami v tomto adresári. Ak sa už daný adresár nachádza v tabuľke `Albums`, je ignorovaný a pokračuje sa na ďalšom



Obrázok 5.3: Vyhľadávanie viacslovných názvov.

adresári. Ak je nájdený adresár s albumom, ktorý sa ešte v databáze nenachádza, je tam pridaný. V ďalšom kroku sa prehľadá jeho obsah a ak sa tam nachádza aspoň jeden súbor s príponou mp3, je pridaný do tabuľky `Songs`. Tento súbor je uložený s jedinečným indexom a cudzím kľúčom odkazujúcim na album ku ktorému daná skladba patrí. Každý mp3 súbor je pred samotným pridaním otvorený a pomocou funkcie `xine_get_meta_info()` sa prečítajú potrebné ID3 tagy, čiže názov piesne (*TITLE*) a umelec (*ARTIST*). Súčasťou ukladaných dát, je aj presný názov súboru. Ďalšia metóda slúžiaca na aktualizáciu databázy je `UpdatePlays()`, ktorá zvýši hodnotu `plays` v tabuľke `Songs` pre aktuálne hranú pieseň. Tento údaj má len štatistický charakter.

5.4 Testovanie

V rámci implementácie aplikácie som vykonal niektoré testy, ktoré mali demonštrovať funkčnosť aplikácie.

Užívateľský test

Kedže pri tomto type aplikácie sa kladie dôraz na užívateľskú prívetivosť, jeden z testov bol zameraný práve na túto situáciu. Priebeh testu bol taký, že užívateľovi, ktorý sa predtým s touto aplikáciou nestretol bola aplikácia spustená. Prvotná reakcia užívateľa bola relatívne rýchla a vďaka podobnosti navigácie v menu, zo známych zariadení typu jukebox, nemal väčší problém orientovať sa v prostredí programu.

Test prehrávacieho jadra

V rámci tohto testu, sa zisťovala funkčnosť implementácie knižnice *xine*, slúžiacej na prehrávanie súborov. Počas tohto testu bola aplikácia spustená nepretržite niekoľko hodín a v určitých časových intervaloch boli simulované vstupy od užívateľa. Počas celého behu aplikácie nenastal žiaden problém súvisiaci s ovládaním aplikácie ani žiaden problém s implementáciou jadra prehrávača.

Databázový test

V rámci databázových testov, bola testovaná rýchlosť funkcie pre ukladanie piesní do databázy. Táto funkcia sa vďaka tomu, že neprehľadáva celý obsah súboru, ale zo súboru číta len potrebné údaje (podrobnosti v kapitole 5.3) ukázala ako veľmi efektívna. V rámci testov bola funkcia otestovaná na vzorke dát približne 46.8 GB. Pridanie tohto objemu dát, čo znamenalo približne 8350 piesní prebehlo za 200 sekúnd. Výhodou tejto funkcie je, že načítanie a uloženie dát prebehne len raz a pri ďalších spusteniach programu sa ukladajú len novo pridané piesne, čo výrazne urýchli proces aktualizácie databázy. V rámci databázových testov bola otestovaná aj rýchlosť užívateľských dotazov na vyhľadávanie, kde sa ukázalo, že čas potrebný na vyhľadanie názvu zadaného užívateľom bol dostatočne krátky, čiže nevznikali žiadne odozvy a vyhľadávanie nepôsobilo rušivo.

Kapitola 6

Záver

Hlavným cieľom práce bolo vytvorenie softvéru pre prehrávanie hudby pre komerčné účely. Tento cieľ bol splnený.

Poznatky z prvého bodu zadania, týkajúce sa tvorby grafických užívateľských rozhraní sú diskutované v druhej kapitole.

Súčasťou prvého bodu zadania bolo tiež preštudovať problematiku prehrávania súborov MP3. Touto problematikou sa zaoberá tretia kapitola, ktorá pojednáva o spôsobe kompresie zvuku, histórii, štruktúre a kompresii MP3 súborov a ponúka prehľad populárnych programov pre prehrávanie hudby.

Nasledujúca, teda štvrtá kapitola vychádza z druhého a tretieho bodu zadania. Táto kapitola je určená návrhu a analýze požiadavkov pre softvér pre prehrávanie hudby. Takisto sa venuje návrhu štruktúry softvéru a popisuje výhody a nevýhody zvoleného riešenia.

V piatej a zároveň poslednej kapitole sa zameriavam na popis implementácie aplikácie. V tejto kapitole je v troch častiach postupne rozobratá implementácia grafického rozhrania, prehrávacieho jadra aplikácie a tvorba databázovej časti. Na konci tejto kapitoly sú v stručnosti prezentované techniky akými bola aplikácia testovaná.

V rámci tejto bakalárskej práce sa podarilo implementovať kompletný program pre prehrávanie MP3 súborov. Program bol vyvíjaný a otestovaný pre operačný systém Linux (distribúciu Kubuntu). Aplikácia bola otestovaná niekoľkými spôsobmi, popísanými v kapitole 5. Pri testovaní aplikácia fungovala bez problémov a vykazovala úplnu funkčnosť odpovedajúcu zadaniu. Vďaka použitiu toolkitu *wxWidgets* je po úpravách možné tento program preniesť aj na iné platformy. Pri tvorbe aplikácie som sa nestretol s výraznými problémami.

Program bol vypracovaný v rozsahu zadania a bol vyvíjaný s ohľadom na ďalšie rozšírenia. V budúcnosti by som chcel aplikáciu rozšíriť o možnosť aktualizovať hudobné záznamy automaticky zo servera. Možnosti niektorých ďalších rožšírení sú popísané v tretej kapitole. Takisto ďalším možným a najmä užitočným rozšírením aplikácie by bolo spraviť nenáročný program, ktorý by spravoval súbor s adresármi hudobných záznamov.

Osobne má práca na tomto projekte bavila a vďaka tejto bakalárskej práci som sa zdokonalil v jazyku C++ a získal veľa užitočných poznatkov z oblasti tvorby užívateľských rozhraní i prehrávania audio súborov na počítači. Najväčšou výhodou ale je, že mám k dispozícii všetky zdrojové kódy tejto aplikácie.

Literatúra

[1] Cole, J.: The History of the MP3 File Format [online]. September 2006.
URL http://www.buzzle.com/articles/history-mp3-file-format.html

[2] Fišnar, J.: *Editor elektronických schemat* [diplomová práca]. Brno: FIT VUT v Brně, 2006.

[3] Hacker, S.: *MP3: The Definitive Guide*. Sebastopol: O'Reilly, Marec 2000, iSBN 1-56592-661-7.

[4] Raissi, R.: *The Theory Behind Mp3*. mp3-tech.org, December 2002.

[5] Smart, J.; Hock, K.; Csomor, S.: *Cross-Platform GUI Programming with wxWidgets*. Pearson Education, Inc., 2006, iSBN 0-13-147381-6.

[6] Sovič, D.: *MPEG audio (Layer 1,2 a 3)* [online]. [cit. 2009-05-08].
URL http://www.pakuj.host.sk/mpeg_audio/layer.html

[7] Wikipedia: *History of the graphical user interface* [online]. [cit. 2009-05-14].
URL
http://en.wikipedia.org/wiki/History_of_the_graphical_user_interface

[8] WWW stránky: The mp3 History [online]. [cit. 2009-05-10].
URL http://www.iis.fraunhofer.de/EN/bf/amm/mp3history/

[9] Zemčík, P.: *Tvorba uživatelských rozhraní* [učebný text]. Brno: VUT v Brně, November 2006.

Dodatok A

Obsah CD

Priložené CD obsahuje:

- Zdrojové kódy aplikácie (adresár *src*)
  - zdrojové súbory aplikácie
  - makefile
  - konfiguračný súbor *paths*
  - súbor *readme*
  - program v spustiteľnej forme
- Technickú správu (adresár *doc*)
  - zdrojové súbory technickej správy vo formáte TEX
  - technickú správu vo formáte PDF

Dodatok B

Manuál

Kompilácia a spustenie programu

Preklad programu sa vykoná príkazom `make` v adresári s programom. Po skompilovaní aplikáciu spustíme príkazom `./jukebox`.

Návod na použitie

Pred spustením programu je treba nastaviť cesty k adresárom s hudobnými súbormi na disku. Toto nastavenia sa vykoná pomocou editácie súboru *paths*

Príklad formátu súboru paths:
/home/media/USB/!myMusic/

/home/main/music/

Po spustení programu sa začnú prehľadávať tieto adresáre a postupne je databáza plnená dátami. O priebehu aktualizácie je užívateľ informovaný v konzole. Po spustení sa aplikácia dostáva do celoobrazovkového režimu. Pre pohyb v menu smere nahor slúži klávesa šípka nahor (↑) a pre pohyb opačným smerom, klávesa šípka dole (↓). Prepínanie medzi jednotlivými panelmi s menu slúži klávesa tabulátor (TAB). Pre simuláciu pridávania kreditov, je nutné použiť klávesu šípka vpravo (→). Na výber zvolenej piesne slúži klávesa Enter. Ďalej, je umožnené užívateľovi zadávať vyhľadávacie kritéria v databáze a to pomocou klasických alfanumerických znakov klávesnice.