VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

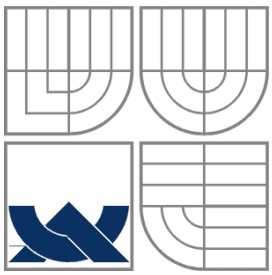
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

HARMONIZACE MELODIE

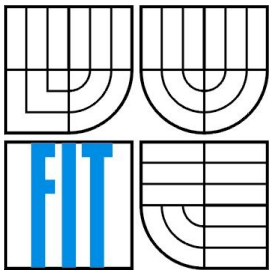
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ONDŘEJ MAŇÁK
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

HARMONIZACE MELODIE

MELODY HARMONIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ MAŇÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL FAPŠO

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Maňák Ondřej**
Obor: Informační technologie
Téma: **Harmonizace melodie**
Kategorie: Umělá inteligence

Pokyny:

1. Zoznámte sa so základnými pravidlami harmonizácie melódie a s metódami riešenia CSP (Constraint Satisfaction Problem).
2. Navrhnite softvér, ktorý k zadanej jednoduchej melódii vytvorí ďalšie harmonické hlasy a výsledok vhodným spôsobom zobrazí alebo prehrá.
3. Implementácia predchádzajúceho bodu
4. Prezentácia funkčnosti programu na niekoľkých cvičeniach z Učebnice harmonie.

Literatura:

- ABC hudební nauky, Luděk Zenkl, 2003 (môžem požičať)
- Učebnice harmonie, Jaroslav Kofroň, 2002 (môžem požičať)
- Podľa pokynov vedúceho

Při obhajobě semestrální části projektu je požadováno:

- 1, 2

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Fapšo Michal, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Ondřej Maňák**
Id studenta: 79317
Bytem: Obřanská 146, 614 00 Brno
Narozen: 13. 12. 1984, Jeseník
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Harmonizace melodie
Vedoucí/školitel VŠKP: Fapšo Michal, Ing.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato bakalářská práce se zabývá problematikou automatické harmonizace melodie. Program využívá principů harmonizace a je implementován v programovacím jazyce Perl. Vstupem je textový soubor, který obsahuje zadanou melodii. Výstupem je midi soubor obsahující melodii s klavírním doprovodem.

Klíčová slova

Harmonizace melodie, midi, akord, polyfonie

Abstract

The focus of this bachelor's thesis is an automatic harmonization of melody. The programme uses axioms of harmonization and is implemented in Perl language. The input is a text file that contains the melody that should be harmonized. The output is a midi file that contains the melody with the piano harmonization part.

Keywords

Melody harmonization, midi, chord, polyphony

Citace

Maňák Ondřej: Harmonizace melodie. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Harmonizace melodie

Prohlášení

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

……
Jméno Příjmení
Datum

Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé práce Ing. Michalu Fapšovi, který mi byl oporou a velkou pomocí při tvorbě této práce.

Obsah

Úvod

Tato práce je textovou částí mé bakalářské práce. Již z tématu vyplývá, že se nejedná o problematiku, na kterou se dá nahlížet čistě z hlediska informačních technologií. Je třeba brát v úvahu i hudební teorii, ze které vychází algoritmy použité v programu. Je tedy nemožné se vyhnout hudebním termínům. Tyto termíny jsem se pokusil velmi stručně vysvětlit ve slovníčku pojmů, který je jednou z příloh této práce. Pro lepší čitelnost textu jsou pojmy, které jsem použil poprvé, vysvětleny rovněž v poznámce pod čarou.

Mnoho lidí zastává názor, že hudbu mají dělat lidé a ne počítače. Já se však zapojení výpočetní techniky do tvorby hudby nebráním. Hudba se dá totiž z velké části popsat pomocí matematických pravidel, a to k využití počítačů v kompozici přímo vybízí. Počítače jsou pak velmi dobrým pomocníkem pro tvoření hudby. Invence by však měla stále ležet na skladateli. Počítač může být schopen napodobovat hudební styly, ale pochybuji, že může dosáhnout takové geniality jako třeba J. S. Bach, aby nové styly tvořil. K tomu je potřeba citu. Myslím, že kdybychom nechali tvorbu hudby čistě na počítačích, omezili bychom tím pokrok v hudbě.

Mým cílem bylo napsat program, který je nejen užitečným pomocníkem hudebního skladatele, ale i prostředkem pro vytvoření jednoduchého doprovodného partu pro člověka, který chce hrát písně, k nimž nemá k dispozici doprovod, a není schopen si ho vytvořit. Základem jednoduchého doprovodného partu jsou akordy přiřazené jednotlivým tónům melodie. Výstupem programu na harmonizaci jsou potom variace na tento akordický základ.

Vstupem programu je melodie zapsaná ve formě textového souboru, výstupem je midi soubor. Midi výstup je důležitým aspektem mé práce, protože většina programů pro práci s hudbou umožňuje midi vstup. Tím je umožněna další práce s mým výstupem. Z tohoto výstupu pak lze například vygenerovat notový zápis nebo ho použít k ozvučení příslušnými hudebními nástroji.

Dalším aspektem, na který jsem se zaměřil, je jednoduchost pro uživatele. Hudebník totiž není ve většině případů odborník na práci s počítači, proto je pro něj důležité, aby se byl schopen jednoduše a rychle naučit s programem pracovat. Proto je součástí programu také nápověda, která je v podstatě jednoduchým návodem k použití.

V první kapitole se budu zajímat o principy harmonizace obecně, přičemž jednotlivé metody harmonizace budu vztahovat ke své práci. Zaměřím se zejména na možnosti pojetí harmonizace. Dále na harmonizaci melodie pomocí kvintakordů[1]. Nakonec budu psát o převodu vertikály do horizontály, což je jeden z alternaivních přístupů při harmonizaci.

Kapitola 2 je věnována programům zabývajícím se harmonizací. Zejména zde budu pojednávat o Strasheele, jakožto kompozičním systému vytvořeným Andersem Torstenem. Tento projekt byl pro mě určitě značnou inspirací.

1 kvintakord = trojzvuk složený ze dvou tercií

Kapitola 3 obsahuje návrh programu. Nejprve budu psát o vstupu, který je ve formě textového souboru, a výstupu ve formě midi souboru. Dále se pokusím osvětlit princip fungování programu. Program funguje vpodstatě ve třech krocích: načtení vstupu, vlastní harmonizace melodie a vytvoření výstupního souboru. Nakonec se budu věnovat vlastnímu návrhu tříd. Během vývoje programu jsem se rozhodl původní návrh částečně pozměnit. Změny a důvody k nim jsou vysvětleny na konci kapitoly 3.

Kapitola 4 je věnována vlastní implemantaci programu. Nejprve budu psát o Perlu, jakožto o programovacím jazyku, který jsem si pro tento projekt vybral. Pokusím se objasnit jeho výhody a nevýhody pro tvorbu projektu typu mé bakalářské práce. Dále zmíním Ubuntu 7.10, kde jsem program vyvíjel. Následuje vlastní popis implementace jednotlivých tříd. Nakonec zmíním několik zkušeností, které jsem během vývoje projektu získal.

V kapitole 5 budu psát o možnostech dalšího vývoje mé práce. Do projektu bych chtěl zavést zejména principy řešení úloh s omezujícími podmínkami. Dále bych se rád věnoval kontrapunktu[2]. Platným rozšířením by bylo rovněž zavedení grafického rozhraní a midi vstupu.

Kapitola 6 je věnována zhodnocení a přínosu této práce.

2 kontrapunkt = nauka o polyfonním skladebném způsobu, znamená tedy spojení dvou nebo více samostatných hlasů v hudební celek

1 Principy harmonizace

Harmonie popisuje, třídí, vysvětluje a hodnotí strukturu akordů (tvar, zápis, funkce, napětí...) a jejich vzájemné vztahy řazení a spojování v harmonické větě (zákonitosti a pravidla funkčních vztahů - spoje, nástupy, rozvody...). Harmonizace melodie (sopránu) je vlastně proces vyhledávání skryté harmonie, kterou melodie obsahuje. Během historického vývoje hudby a hudební teorie prošel proces harmonizace dlouhým a spletitým vývojem. [1]

Například v počátcích evropské hudby byly za konzonance[3] považovány pouze čisté intervaly[4] (prima, kvarta, kvinta a oktáva), naopak velké intervaly (sekunda, tercie, sexta a septima) byly považovány za dizonance. V současnosti je brána velká tercie jako konzonantní interval, na němž je vystavěn celý kvintakordový systém, který je základním akordickým systémem v Evropě už od středověku.

Liší se rovněž pojetí různých kultur. Z tohoto pohledu jsou zajímavé například systémy stupnic používaných na Dálném východě. Když se řekne hudba Dálného východu, představíme si všichni známou pentatoniku, sedneme ke klavíru, začneme hrát pouze na černých klávesách a říkáme si, jak hrajeme čínskou hudbu. Ve skutečnosti to vůbec není tak jednoduchá záležitost. Čínský stupnicový systém je sice vystavěn na pentatonice, ale je mnohem komplikovanější a složitější. Navíc Čína není jedinou zemí Dálného východu.

Nejpozoruhodnější a nejsložitější systém stupnic je používán v Indii, kde existuje několik stovek různých modů[5] (pro srovnání evropská kultura používá převážně 2 mody - dur / moll), které často používají i mikrointervaly[6]. Každá z těchto stupnic je potom používaná pro specifické účely, například k posezení u odpoledního čaje apod. Charakter indické hudby i harmonizační principy jsou potom zcela odlišné od postupů, které jsou známé našim uším.

Pro další příklady nemusíme chodit daleko. Moderní hudební skladatelé mnohdy používají svoje vlastní specifické principy harmonizace. Často se využívají polytonalita[7], mikrointervaly, staré církevní mody, dodekafonie[8], postupy založené na náhodě nebo různé matematické teorie aplikované v hudbě. Výsledek se potom může většině lidem jevit jako chaos dizonantních[9] zvuků, ale problém je spíše v tom, že na danou hudbu nejsme zvyklí. Vzpomeňme si například na to, že tercie byla dříve považována za dizonanci.

---

3 konzonance = souzvuk dvou nebo několika tónů, který na náš sluch působí příjemně

4 interval = výšková vzdálenost mezi dvěma tóny

5 modus = stupnice

6 mikrointerval = interval menšího rozsahu než půltón

7 polytonalita = vztah k několika tonálním centrům současně

8 dodekafonie = zrovnoprávnění všech dvanácti tónů temperované chromatiky v evropské hudbě

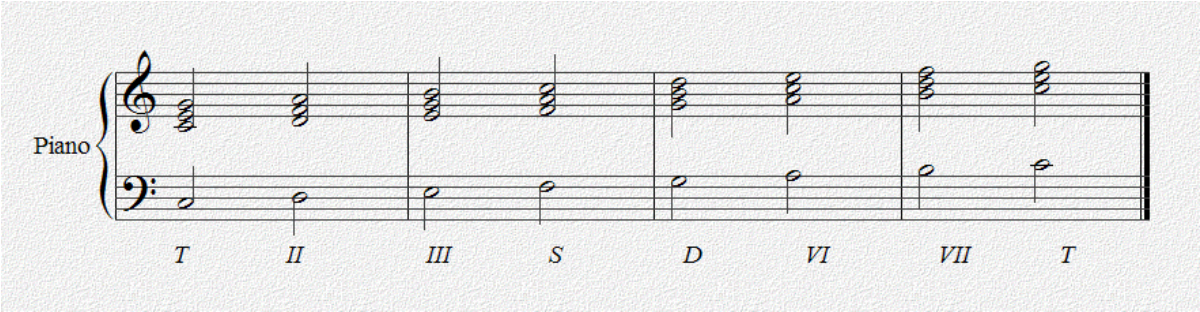
9 dizonance = souzvuk dvou nebo několika tónů, který působí drsně až nepříjemně

1.1 Harmonizace melodie kvintakordy

Pro svůj program jsem se rozhodl použít princip harmonizace melodie pomocí kvintakordů. Tento způsob je totiž nejběžnějším principem používaným v evropské hudební kultuře a lze jej jednoduše rozšířit o používání sextakordů[10], kvartsextakordů[11], septakordů apod. [2]

Omezil jsem se pouze na používání doškálních kvintakordů[12] a předpokládám, že v melodii se objevují pouze doškální tóny[13]. Nejčastěji používám hlavní kvintakordy (tónika[14], dominanta[15], subdominanta[16]). V některých případech jsou potom tyto kvintakordy nahrazeny kvintakordy vedlejšími (kvintakordy vystavěné na vedlejších stupních).

Následující obrázek ukazuje doškální kvintakordy stupnice[17] C-dur (T = tónika, D = dominanta, S = subdominanta, II = kvintakord druhého stupně atd.)



Obr. 1-1 Doškální akordy stupnice C-dur

1.1.1 Postup harmonizace kvintakordy

Při harmonizaci melodie pomocí kvintakordů je možno postupovat dle následujícího schématu.

Stanovíme závěr - nejčastěji bývá na konci melodie spoj dominanta - tónika (závěr autentický celý), méně často subdominanta - tónika (závěr plagální celý). Jestliže nelze použít tóniku ke zharmonizování posledního tónu melodie, použijeme spoj tónika - dominanta (závěr autentický poloviční) případně tónika - subdominanta (závěr plagální poloviční). [1]

Přiřadíme jednotlivým tónům melodie příslušné akordy - akordy přiřazujeme podle priority (tónika, dominanta, subdominanta, dále akordy na šestém, druhém, třetím a sedmém stupni), přičemž přiřazovaný akord musí obsahovat tón, kterému má být přiřazen (např. tónu G ve stupnici C-dur lze

10 sextakord = první obrat kvintakordu, basový tón se přesune o oktávu výš, novým basovým tónem je potom tercie původního akordu, která s novým sopránem tvoří charakteristickou sextu

11 kvartsextakord = druhý obrat kvintakordu, basovým tónem je kvinta původního akordu

12 doškální akord = akord skládající se pouze z tónů daného modu

13 doškální tón = tón vyskytující se v danému modu

14 tónika = doškální kvintakord vystavěný na prvním stupni daného modu

15 dominanta = doškální kvintakord vystavěný na pátém stupni daného modu

16 subdominanta = doškální kvintakord vystavěný na čtvrtém stupni daného modu

17 stupnice = stoupající nebo klesající uspořádaná řada tónů (často v rozmezí jedné oktávy)

přiřadit akord C-dur, protože akord C-dur obsahuje tón G). Dále musí být dodržován přísný zákaz paralelního postupu mezi basem (základní tón akordu) a sopránem (melodií). To znamená, že bas a soprán nesmějí postupovat o stejný interval kromě čisté primy, tedy dvou stejných tónů po sobě (např. pokud je tónu G přiřazen akord C-dur, nemůže být následujícímu tónu A přiřazen akord D-dur, protože bas i soprán by postupovaly o stejný interval).

Doplníme ostatní hlasy - pokud používáme čtyřhlas, přiřadíme basu základní tón a ostatním hlasům potom druhý a třetí tón akordu. Můžeme však i různě rytmizovat nebo figurovat doprovod v závislosti na tom, pro jaké účely jej používáme.

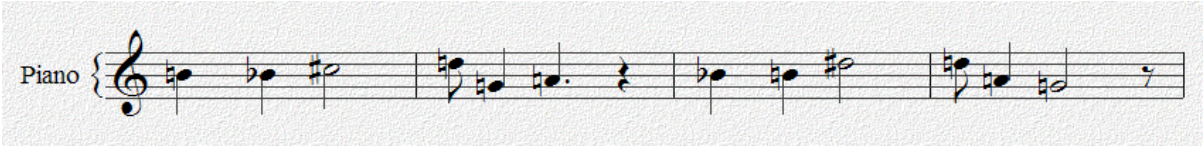
Z předešlých odstavců je vidět, že harmonizace melodie se dá popsat několika jednoduchými pravidly. Vše, co lze popsat pomocí pravidel, lze i naprogramovat. Výpočetní technika se potom může stát platným pomocníkem pro hudebníka. Proto je zde snaha vytvořit software, který umí harmonizovat melodii.

1.2 Převod horizontály do vertikály

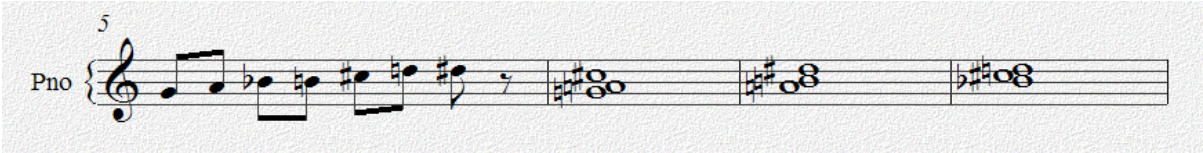
Převod horizontály do vertikály je jedním z alternativních přístupů k harmonizaci melodie používaných v moderní vážné hudbě.

Neuvažuje se žádný modus, proto se tento postup obzvláště hodí k harmonizaci melodie, která popírá tonalitu. Jde vpodstatě o vytvoření skupiny akordů, které nemusí být vystavěny terciovým způsobem, z tónů dané melodie. Následně se tyto akordy přiřazují jednotlivým tónům melodie. Přiřazování akordů může probíhat intuitivně, podle nějakých pravidel (priorita akordů, zákaz paralelních postupů apod.) nebo třeba zcela náhodně. Výsledkem je potom hudba kterou spojuje příbuzný tónový základ.

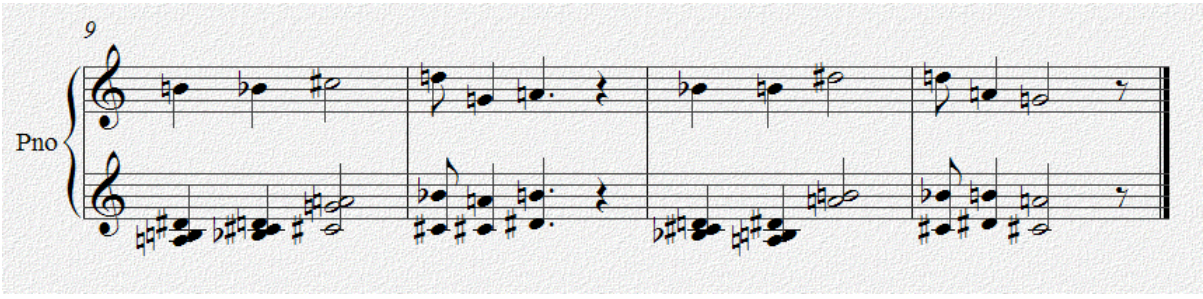
Následující série příkladů ukazuje uplatnění této metody v praxi.



Obr. 1-2 Melodie zadaná na vstupu



Obr. 1-3 Vytvoření akordů pro doprovod



Předchozí příklad bude asi většině lidem znít poněkud disharmoniciky, jelikož při tvorbě akordů byly vybrány souzvuky obsahující poměrně dizonantní intervaly. Ale i to je jedna z možností, kudy se může harmonizace ubírat. Není podmínkou, že se k harmonizaci melodie musí používat pouze konzonance.

Výhodou této metody je její univerzálnost. Dá se použít například k harmonizaci nějakých přechodů, modulací[18] nebo melodií, které jinak zharmonizovat nedovedeme. Tuto metodu zde uvádím zejména proto, že její využití je jedním z možných rozšíření mé bakalářské práce do budoucna.

18 modulace = přechod z jedné tóniny do druhé

2 Programy zaměřené na harmonizaci

Hledání programů zabývajících se harmonizací melodie byla jedna z prvních věcí, na které jsem se zaměřil v rámci práce na mém bakalářském tématu. Záhy jsem zjistil, že můj program nebude prvním softwarem snažícím se o zharmonizování melodie.

Vyvstala tedy otázka, zda má vůbec smysl něco programovat. Mohl jsem si například vybrat nějaký open source projekt a pracovat na něm. Nakonec jsem si řekl, že bude lepší začít pěkně od začátku na zelené louce. Člověk je potom schopen lépe pochopit danou problematiku. Navíc spousta programů je napsána vpodstatě vícekrát, ale každý jeden program je v něčem jiný, a proto může být právě on přínosem v dané tematice.

Přesto si myslím, že je dobré vyjít vždy z toho, co už existuje. Práce jiných nám pak může být inspirací, nápovědou nebo radou k tomu, jak daný problém řešit.

2.1 Strasheela

Díky vedoucímu své práce Ing. Michalu Fapšovi jsem se seznámil s projektem Strasheela. Jedná se o práci Anderse Torstena, jejímž výsledkem je kompoziční systém založený na omzeujících podmínkách.

Anders Torsten se narodil roku 1968 v Güstrow (Německo). Studoval muzikologii a teologii na Humboldtské univerzitě v Berlíně. Později se specializoval na elektroakustiku. Ve svých výzkumech se zabývá možností využití počítačů při komponování hudby. To bylo motivací k vyvinutí Strasheely jakožto programu, který využívá umělou inteligenci ke kompozici hudby. [3]

Torsten vychází z předpokladu, že harmonizace se dá popsat souhrnem pravidel, který musí být dodržen při vytváření doprovodných hlasů. Těchto pravidel je velké množství a jejich řešení vede k poměrně složité problematice. K řešení problémů tohoto typu se s výhodou využívá metod založených na CSP (Constraint Satisfaction Problem).

CSP se dá popsat následujícím způsobem. Je dána konečná množina proměnných, každé proměnné je přiřazena konečná množina hodnot. dále je dána konečná množina podmínek omezujících hodnoty. Úkolem CSP je potom najít úplné ohodnocení proměnných, které splňuje všechny podmínky. Můžeme hledat jedno řešení, všechna řešení nebo optimální řešení. [4]

Výhodou Strasheely je to, že nabízí uživateli možnost do značné míry definovat si svá pravidla, čímž může každý skladatel dosáhnout výsledků, které odpovídají jeho představám. Je potom možno vytvářet hudbu podobným způsobem jako například v baroku. Jinou cestou je definování svých vlastních stylů. [5]

Pravidlem může být například zakázaný paralelní postup mezi dvěma hlasy, pravidla pro rozvod citlivého tónu, pravidla pro tvoření závěrů, dodržování vzdáleností mezi hlasy, dodržování sazby apod.

Jistou nevýhodou tohoto přístupu je jeho složitost z hlediska uživatele.

3 Návrh programu

3.1 Vstup a výstup programu

3.1.1 Vstup

Na vstupu mého programu je textový soubor obsahující melodii ke zharmonizování. Název souboru je dán parametrem `-i VSTUPNI_SOUBOR`, jeho struktura je následující.

Na prvním řádku souboru je zadána stupnice, ve které má být melodie zharmonizována. Zápis názvu stupnice se skládá ze dvou částí. První část označuje tón a případnou posuvku (# nebo b), která jej modifikuje (např. C# = Cis). Druhá část značí, zda se jedná o stupnici durovou (major) nebo molovou (minor).

Na druhém řádku vstupního souboru je zadán takt. Ten je určen celým číslem udávajícím počet dob daného taktu v rozmezí 1 - 9.

Zbytek vstupního souboru tvoří posloupnost tónů harmonizované melodie. Pro každý tón musí platit, že se skládá ze dvou částí: výška a délka. Výška je určena podobným způsobem jako první část názvu stupnice. Jediným rozdílem je možnost přidat číslo značící oktávu daného tónu (3 = malá oktáva, 4 = jednočárkovaná oktáva, 5 = dvoučárkovaná oktáva, 6 = tříčárkovaná oktáva). Pokud není číslo značící oktávu zadané, přepokládá se tón v jednočárkované oktávě. Výška tónu se zapisuje bez mezer (např. Eb4 značí Es1, Eb3 značí es, Eb značí Es1). Délka tónu je číslo, které udává délku tónu ve dvaatřicetinách (např. 8 = 8 / 32 = 1 / 4, tedy nota čtvrťová). Výška tónu je od jeho délky oddělena libovolným počtem bílých znaků (mezera, enter), stejně tak jednotlivé tóny jsou od sebe odděleny libovolným počtem bílých znaků.

Program je „case sensitive“, což znamená, že rozlišuje velká a malá písmena na vstupu. Př. 3-1 ukazuje obsah vstupního souboru pro píseň Kočka leze dírou.

Př. 3-1 Ukázka vstupního souboru (Kočka leze dírou)

```
D major
4
D 4 E 4 F# 4 G 4 A 8 A 8
B 8 B 8 A 16
B 8 B 8 A 16
G 4 G 4 G 4 G 4 F# 8 F# 8
E 8 E 8 A 16
G 4 G 4 G 4 G 4 F# 8 F# 8
E 8 E 8 D 16
```

3.1.2 Výstup

Výstupem mého programu je midi soubor, který je dán parametrem `-o VYSTUPNI_SOUBOR`, program za něj připojí koncovku mid. Midi výstup jsem zvolil proto, že midi formát je výhodný pro další práci s tímto výstupem (např. vytvoření notového zápisu) Další výhodou je to, že soubory `*.mid`, lze snadno přehrát mnohými dostupnými programy pro přehrávání hudby, což umožňuje rychlou sluchovou konfrontaci s tímto výstupem.

Jako hudební nástroj pro výstupní soubor jsem zvolil klavír, protože se jedná o jeden z nejběžnějších doprovodných hudebních nástrojů.

3.2 Princip fungování programu

3.2.1 Načtení vstupu

Po spuštění se program pokusí najít a otevřít soubor daný parametrem `-i`, potom z něj načte melodii a další důležité atributy, kterými jsou stupnice a takt.

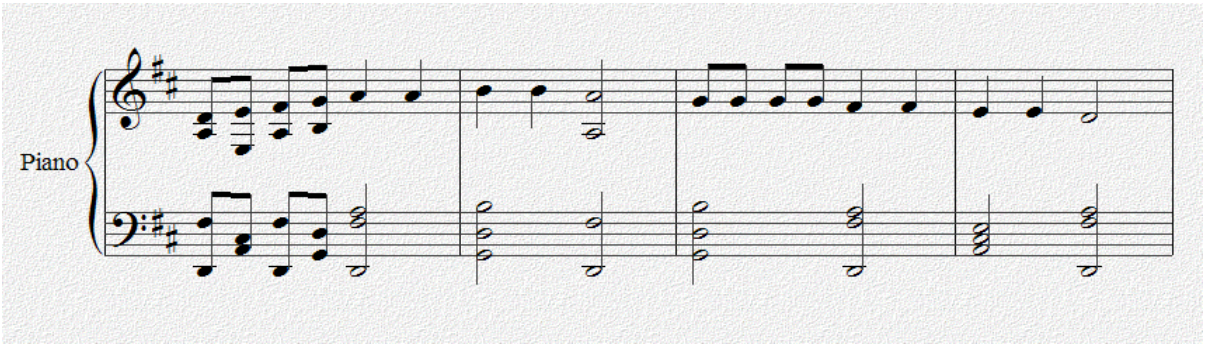
Pokud dojde k nějaké chybě (např. nebyl nalezen vstupní soubor, nebo obsah vstupního souboru je chybný), je volána chybová metoda `error()`, která vypíše chybové hlášení a výzvu ke spuštění programu s parametrem `-h` pro nápovědu. Pokud je program spuštěn s parametrem `-h`, je zobrazena nápověda.

3.2.2 Harmonizace melodie

Pro harmonizaci melodie je využíván algoritmus, který vychází z postupu popsaného v kapitole 1.1 Harmonizace melodie kvintakordy.

Nejprve je vytvořen závěr, poté jsou jednotlivým tónům melodie přiřazeny akordy a nakonec jsou vytvořeny vnitřní hlasy[19]. Před vytvořením vnitřních hlasů se však program zeptá uživatele, který modul harmonizace chce použít. Nyní má uživatel na výběr ze 4 modulů harmonizice.

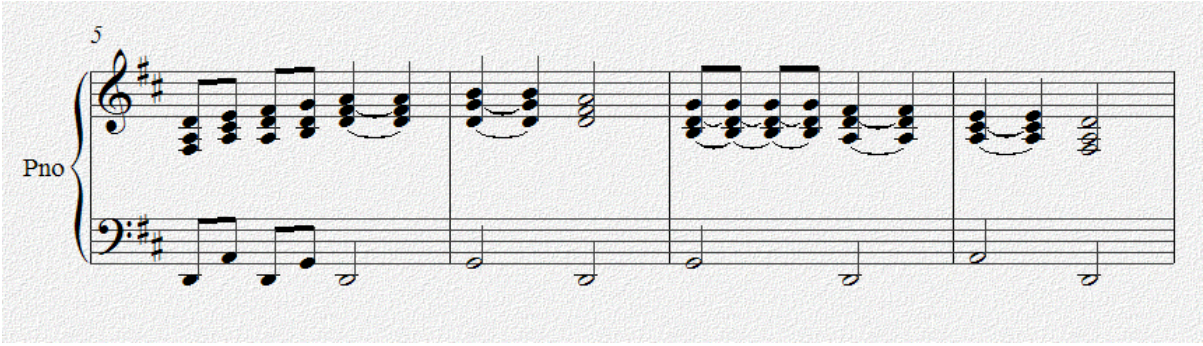
HarmonizerSimple a) vede vnitřní hlasy paralelně s basem (viz Obr. 3.1).



Obr. 3-1 HarmonizerSimple a)

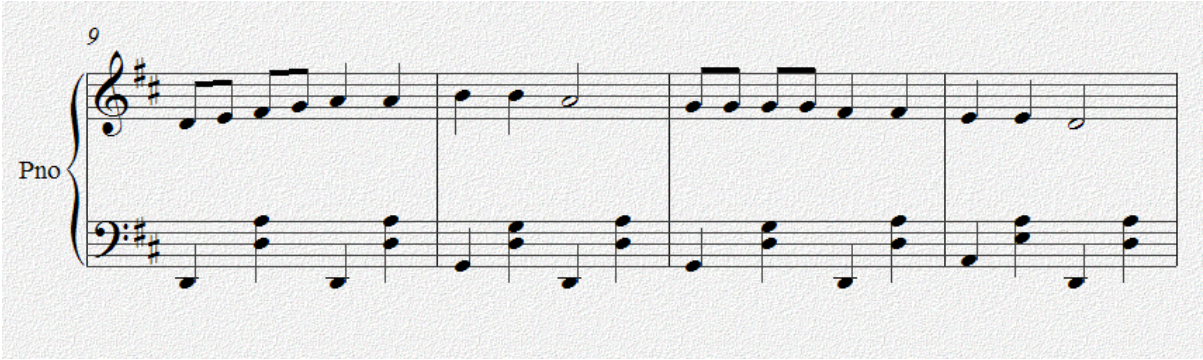
19 vnitřní hlasy = alt a tenor

HarmonizerSimple b) vede vnitřní hlasy paralelně se sopránem (viz Obr. 3.2).



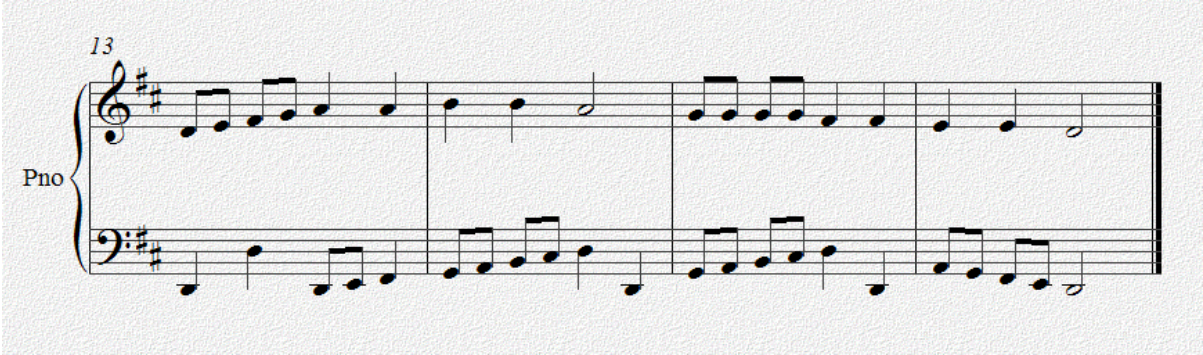
Obr. 3-1 HarmonizerSimple b)

HarmonizeFigure tvoří z basu a vnitřních hlasů doprovodnou figuru (viz Obr. 3.3). Tento harmonizer se vyznačuje tím, že dává uživateli na výběr z několika figur.



Obr. 3-1 HarmonizerFigure

HarmonizerMelodicBas vytváří dvojhlas sopránu s basem, přičemž bas je veden melodicky (viz Obr 3.4).



Obr. 3-1 HarmonizerMelodicBas

Výše zmíněné příklady jsou notovým zápisem výstupu mého programu a jsou uvedeny ve zvukové podobě na přiloženém CD.

3.2.3 Vytvoření midi souboru

Nakonec je z kontejneru not pomocí knihovny jazyka Perl pro midi rozhraní a třídy MidiGenerator vytvořen midi soubor pro výstup.

3.3 Návrh tříd

V této podkapitole se budu věnovat návrhu jednotlivých tříd. Předem je třeba říci, že přestože jsem usiloval o co nejlepší objektový návrh, byl ten původní během implementace částečně pozměněn. Jeho základní jádro a myšlenka však byly naplněny. Změny v návrhu s sebou přinesly mnohé komplikace. Jsou mi však cenným poučením a cennou zkušeností do budoucna. Do této kapitoly uvedu nejprve původní návrh, potom bude následovat finální návrh. Většina změn v původním návrhu byla způsobena zavedením nové třídy `Voice`. Důvody, proč k této změně došlo, se pokusím ozřejmit na závěr.

Jednotlivé třídy, jejich metody a atributy jsou zde pouze nastíněny. Vše bude podrobněji popsáno v kapitole 4 (Implementace). Návrh jsem se snažil koncipovat tak, aby byl program jednoduše rozšiřitelný zejména o další harmonizery, možnosti modulace, změny taktu, polyfonní zpracování apod.

3.3.1 Původní návrh

Původní návrh počítal s následujícími třídami: `NoteHarmonics`, `Staff`, `HarmonizerSimple`, `MidiGenerator` a se třídami pro další harmonizaci. Původně jsem ještě nevěděl, že půjde o `HarmonizerFigure` a `HarmonizerMelodicBas`.

`NoteHarmonics` měla být třída obsahující notu melodie, její délku, takt, ve kterém se daná nota nachází, stupnici, ve které aktuálně probíhá harmonizace, množinu potenciálních akordů, jimiž lze tuto notu melodie doprovodit a další doprovodné hlasy k dané notě.

`Staff` měl reprezentovat kontejner všech not. Původně se mělo jednat o pole objektů třídy `NoteHarmonics`.

`HarmonizerSimple` měl obsahovat základní algoritmus harmonizace, což je v podstatě vytvoření akordického schématu.

`MidiGenerator` měl být třídou jejímž jediným úkolem bylo generovat z objektu třídy `Staff` výstupní midi soubor. Jedná se o jedinou třídu, jejíž původní návrh nebyl během vývoje projektu pozměněn.

3.3.2 Finální návrh

Do finálního návrhu byla přidána třída `Voice`, která obsahuje tón doprovodného hlasu a jeho vlastnosti. Zavedení této třídy ovlivnilo atributy tříd ostatních.

Z `NoteHarmonics` byly odebrány atributy pro doprovodné hlasy. `Staff` byl rozšířen o další tři pole objektů `Voice` pro doprovodné hlasy.

Dále došlo ještě k rozšíření třídy `HarmonizerSimple` o metody, které implementují vytvoření doprovných hlasů vedených paralelně s basem nebo paralelně se sopránem.

3.3.3 Zavedení třídy Voice

Zavedení nové třídy bývá často faktorem, který má významný vliv na celý projekt. Přesto jsem se rozhodl takovou třídu do své práce přidat.

Důvodem byl chybný původní návrh. Ten totiž předpokládal, že každému tónu melodie je přiřazen právě jeden tón (nebo pauza) v každém doprovodném hlase. Tato nota musel být navíc stejné délky jako nota sopránu. Toto je však velmi omezující pravidlo pro doprovod. Zcela vylučuje např. použití doprovodných figur či polyfonii v doprovodu.

Řešením by mohla být změna třídy `NoteHarmonics` tak, že každé sopránové notě by bylo přiděleno pole tónů pro každý doprovodný hlas. Toto pole by mělo nedefinovanou velikost. Toto řešení by však činilo kontejner not značně nepřehledným. Navíc jde ideologicky zcela proti něčemu takovému jako je nezávislost jednotlivých hlasů v polyfonii, o kterou bych svůj projekt v budoucnu rád rozšířil.

Proto jsem se rozhodl zavést třídu `Voice`, která umožňuje jakési vytknutí doprovodných hlasů z třídy `NoteHarmonics`. Vznikla tak třída, která reprezentuje jednotlivé tóny doprovodných hlasů.

4 Implementace

4.1 Výběr programovacího jazyka

Svoji práci jsem původně zamýšlel implementovat v jazyce C++, nakonec jsem se však rozhodl pro Perl. Na následujících několi řádcích vysvětlím, co mě k tomu vedlo.

Perl je interpretovaný programovací jazyk, který roku 1987 vytvořil Larry Wall. Popularita Perlu vzrostla společně s rozvojem internetu, kdy se Perl stal rozšířeným jazykem pro tvorbu CGI skriptů. Při konstrukci jazyka se Larry Wall řídil heslem *„there's more than one way to do it“* (dá se to udělat více způsoby). Perl je vhodným nástrojem k řešení malých i velkých problémů [6]

Díky objektovému rozšíření se dají v Perlu vytvářet prakticky stejné programy jako v C++. Dalo by se spekulovat o tom, že C++ umožňuje narozdíl od Perlu přístup k počítači na nižší úrovni (až na úrovni registrů procesoru), což vede k tvorbě efektivnějšího softwaru. To je sice pravda, ale optimalizace není v našem případě nejzásadnějším aspektem. Naopak nástroje pro práci s midi a rovněž knihovny pro práci s hudbou obecně (`Music::Note, Music::Scales` aj.) jsou v Perlu dostupnější a snáz se používají. Tento faktor byl pro mě při volbě Perlu jakožto programovacího jazyka rozhodující.

Hlavní nevýhodou jazyka Perl je to, že se v něm velmi snadno tvoří nepřehledný kód. Pokud o tomto faktu člověk ví a počítá s ním, je Perl velmi efektivním a pohodlným nástrojem pro vývoj softwaru, který pracuje s hudbou.

4.2 Vývojové prostředí

Perl je jazyk širokého použití, který se dá použít pro vývoj pod mnohými operačními systémy (Linux, Unix, MS Windows, Mac OS...). Měl jsem tedy možnost si vybrat. Pokud je mým úkolem programovat a mám možnost zvolit některou distribuci linuxu, jdu touto cestou. Použil jsem tedy Ubuntu 7.10, protože ho používám na svém notebooku.

Pro psaní kódu jsem zvolil Gedit, což je základní textový editor pro Gnome. Nejedná se o žádné extrémně sofistikované prostředí (je to přece jen textový editor). Ale pro vývoj projektu rozsahu mé práce je to nástroj naprosto dostačující. Navíc pro učení se práce s nějakým jazykem je podle mého názoru mnohdy lepší dát přednost textovému editoru před vývojovým prostředím, které sice dělá spoustu práce za programátora, ale občas hrozí to, že programátor nepochopí některé zákonitosti daného jazyka.

Vývoj mi velmi usnadnila možnost použít volě dostupné knihovny pro práci s hudbou: `Music` (`Music::Note, Music::Scales`) a `MIDI`.

4.3 Implementace tříd a metod

4.3.1 Main.pl

Metody všech tříd jsou volány ze souboru `Main.pl`. V tomto souboru je ještě navíc zpracována kontrola načtení vstupu včetně ošetření případných chyb.

Načtení vstupu má na starosti metoda `readInput()`, která se nejprve pokusí otevřít vstupní soubor, pokud dojde k potížím, vypíše chybové hlášení a ukončí program. Pokud proběhne otevření souboru v pořádku, je vstupní soubor načítán po řádcích. První řádek musí obsahovat název stupnice. Pokud je tento název zadán chybně, což zjistí metoda `checkTime()`, je volána metoda `errors()`, která vypíše chybové hlášení a ukončí program. Druhý řádek vstupního souboru je určen pro takt, ten musí být zadán jako celé číslo v rozmezí 1-9, a vyjadřuje počet dob na takt (uvažuji čtvrťovou notu jako jmenovatele), ošetření správnosti taktu má na starost metoda `checkTime()`. Na zbylých řádcích vstupního souboru jsou zadány jednotlivé tóny melodie a jejich délka. Kontrolu regulérnosti tónů a jejich délky je v kompetenci metody `checkNote()`, která volá pomocnou metodu `isNote()` pro zjištění, zda je zadána nota v rozsahu c - C3. Pokud je vstup vpořádku, je vytvořen kontejner not pomocí `Staff->new()`.

Dále jsou volány následující metody daných tříd, které mají za úkol postupně upravit obsah kontejneru not: `HarmonizerSimple->filterScale()`, `HarmonizerSimple->end()`, `HarmonizerSimple->harmonize()`. Tím je vytvořeno jednoduché harmonické schéma, které je uloženo do kontejneru not (`$staff`).

Následuje výběr jednoho ze čtyř harmonizerů (viz kapitola 3.2.2). Pokud je zadána špatná volba, je uživatel vyzván k opětovnému zadání své volby, dokud nezadá 1, 2, 3 nebo 4. Poté je volán přísušný harmonizér, který z harmonického schématu vytvoří příslušný doprovod.

Nakonec je volána metoda `MidiGenerator->generate()`, které je předán kontejner not `($staff)`. Metoda `MidiGenerator->generate()` potom vytvoří midi soubor pro výstup a celý program je ukončen.

Pokud je program spuštěn s argumentem `-h`, je volána metoda `help()`, která vypíše nápovědu k programu. Potom je program ukončen.

4.3.2 NoteHarmonics.pm

`NoteHarmonics.pm` je souborem atributů daného tónu melodie. Atributy třídy NoteHarmonics jsou následující: `"scale"` = stupnice pro danou notu (rozhodl jsem se každé notě přiřadit vlastní stupnici, protože počítám s rozšířením tohoto projektu o zahrnutí modulací), `"time"` = délka taktu (opět přiřazen každé notě zvlášť, a to ze stejného důvodu jako stupnice), `"beat"` = doba v taktu, na které nota začíná, `"index"` = index přiřazeného akordu z pole akordů, `"sopran"` = sopránová nota,

"notetmp" = pomocná nota pro bas, "length" = délka sopránové noty, "chords" = pole akordů, které mohou být přiřazeny dané notě.

Třída NoteHarmonics má dvě metody: new(), což je konstruktor objektu této třídy. Tato metoda volá metodu getChords(), která vytvoří pole akordů, které mohou být přiřazeny dané notě. Těmito akordy se myslí všechny kvintakordy, které danou notu obsahují.

4.3.3 Voice.pm

Voice.pm je souborem atributů daného doprovodného hlasu. Obsahuje notu daného hlasu a její délku. Týká se pouze basu, tenoru a altu (soprán je obsažen v NoteHarmonics). Jedinou metodou této třídy je konstruktor new(), který přiřadí každé notě její atributy (výšku a délku) a vytvoří ji.

4.3.4 Staff.pm

Staff.pm je kontejnerem všech not. Obsahuje pole objektů třídy NoteHarmonics, dále 3 pole objektů třídy Voice (noty doprovodných hlasů), posledním atributem této třídy je "count", což je počet not zadané melodie.

Třída Staff má dvě metody: kontruktor new() a beat(). Metoda beat() slouží k určení doby v taktu, na které začíná sopránová nota.

4.3.5 HarmonizerSimple.pm

HarmonizerSimple je programovým jádrem pro harmonizaci. Obsahuje nejen metody, které vytvoří základní harmonické schéma, ale je zároveň základním harmonizerem, který dává na výběr ze dvou odlišných možností. Vnitřní hlasy totiž mohou postupovat buďto paralelně s basem nebo paralelně se sporánem.

Třída HarmonizerSimple obsahuje tyto metody: filterScale(), filterScaleTmp(), end(), harmonize(), selectChord(), scale(), paralel(), lookForward(), basParalel() a sopranParalel().

Metoda filterScale() prochází polem "nh" (pole objektů třídy NoteHarmonics) třídy Staff. Pro každý objekt třídy NoteHarmonics potom volá pomocnou metodu filterScaleTmp(), která z pole akordů pro danou sopránovou notu vyřadí ty, které nejsou doškálními akordy stupnice určené atributem "scale". Metoda filterScale() potom upraví atribut "chords" u každého objektu pole NoteHarmonics.

Metoda end() vytvoří závěr pro danou melodii. Podle pravidel popsaných v kapitole 1.1.1. Vybraný akord je nastaven pomocí změny hodnoty atributu "index" třídy NoteHarmonics.

ostatniHlasyPo zpracování kontejneru not metodami `filterScale()` a `end()` je volana metoda harmonize(). Tato metoda je ústředním harmonizerem, který postupně prochází kontejner not a na každou notu melodie aplikuje algoritmus v následujícím příkladu (Př. 4-1)

Př. 4-1 Algoritmu metody harmonize()

```
1) vyber akord dle priority a paralely (pomocné metody selectChord(),
   scale() a paralel())
2) podívej se dopředu, zda nenastal konflikt kvůli závěru (pomocná
   metoda lookForward())
   - pokud je vše v pořádku, přejdi k další notě
   - pokud nastala chyba, vyřaď vybraný akord a vrať se k bodu 1)
```

Metoda `basParalel()` vytvoří ostatní hlasy (bas, tenor a alt) k sopránu dle vybraného akordu, tak, že vnitřní hlasy jsou vedeny paralelně s basem.

Metoda `sopranParalel()` vytvoří ostatní hlasy (bas, tenor a alt) k sopránu dle vybraného akordu, tak, že vnitřní hlasy jsou vedeny paralelně se sopránem.

4.3.6 HarmonizerFigure.pm

`HarmonizerFigure` přebírá kontejner not, který již byl upraven pomocí metod třídy `HarmonizerSimple` tak, že obsahuje harmonické schéma.

Úlohou `HarmonizeruFigure` je potom stylizovat doprovod pomocí jedné ze čtyř figur, které dává uživateli na výběr. Třída `HarmonizeFigure` obsahuje tyto metody: `harmonize(), beat(), figure1(), figure2(), figure3()` a `figure4()`.

Metoda `harmonize()` je první volanou metodou této třídy. Její úlohou je vyzvat uživatele k výběru jedné ze čtyř nabízených figur. Dále už jen volá metodu pro příslušnou figuru. Metoda `beat()` je pouze pomocnou metodou pro určení doby v taktu. Metody `figure1(), figure2(), figure3()` a `figure4()` slouží k vytvoření jednotlivých doprovodných figur. Při tvorbě figur je bráno v potaz právě to, o jakou dobu v taktu se jedná.

4.3.7 HarmonizerMelodicBas.pm

Třída `HarmonizerMelodicBas` stejně jako `HarmonizerFigure` přebírá kontejner not, který byl již upraven metodami třídy `HarmonizerSimple`.

Úlohou `HarmonizerMelodicBas` je potom vytvořit dvojhlas, kdy je bas veden pokud možno melodicky. Vzniká tak vlastně jakási velmi jednoduchá polyfonní struktura. Melodičnosti basu je docíleno především vkládáním melodických tónů mezi těžké doby[20].

Třída `HarmonizerMelodicBas` obsahuje tyto metody: `harmonize(), insertNotes()` a `Scale()`.

20 těžká doba = doba, na které je v daném taktě přízvuk

Metoda `harmonize()` simuluje běh času v nejmenších časových úsecích (dvaatřicetiny). Pokud nastává těžká doba, je vytvořena basová nota jakožto základní tón akordu, který doprovází příslušnou sopránovou notu. Mezi basové tóny jsou potom postupně vloženy melodické tóny pomocí metody `insertNotes()`. `Scale()` je pouze pomocná metoda pro vytvoření tonálního terénu pro daný modus.

4.3.8 MidiGenerator.pm

Třída `MidiGenerator` slouží k vytvoření midi výstupu. Z kontejneru not vytvoří soubor `vystup.mid`. Během své práce využívá knihovny pro práci s hudbou (`Music`) a s midi (`MIDI`).

Metody třídy `MidiGenerator: generate(), note2number()` a `staffLength()`.

Metoda `generate()` je vlastním generátorem midi. Ve svém hlavním cyklu prochází kontejner not po dvaatřicetinách, jakožto nejmenším přípustném časovém úseku, a postupně pomocí midi událostí spíná a vypíná jednotlivé noty v jednotlivých hlasech.

Metoda `note2number()` je pomocná metoda, která převádí výšku noty na číslo určené výškou tónu. Výška tónu je určena oktávou, ve které se nachází, tónem (C, D, E, F, G, A, B) a případnou posuvkou. Tato metoda je používána rovněž třídou `HarmonizerSimple` při určování paralelních postupů mezi hlasy v metodě `paralel()`.

4.4 Postřehy a doporučení

Výběr programovacího jazyku, operačního systému i způsob vývoje programu byl zvolen tak, že se dal projekt bez závažnějších problémů naimplementovat. Přesto bych chtěl upozornit na několik málo postřehů a zkušeností, kterých se mi dostalo při implementaci mé bakalářské práce.

Perl je programovací jazyk, který má syntakticky velmi blízko k jazyku C, což je jazyk, který je domovinou mnohých programátorů. Kód v Perlu však může být méně přehledný. Jedná se možná z velké části i o to, že jsem s Perlem začal pracovat až v souvislosti s mojí bakalářskou prací. Proto jsem používal mnohem častěji komentáře, což se mi několikrát vyplatilo. Také je třeba klást větší důraz na volbu názvů identifikátorů tak, aby co nejpřesněji vystihovaly podstatu problému, ke kterému se vztahují. Tento fakt je dán tím, že v Perlu se neuvádí typ identifikátorů.

Další záludností je uvedení špatného jména identifikátoru při přiřazování. Případný překlep může snadno vytvořit novou proměnnou nebo atribut třídy a interpret na tuto skutečnost neupozorní. Potom zoufalý programátor marně hledá chybu v kódu.

Přesto se mi s Perlem pracovalo dobře. Výhodou je například dynamická práce s pamětí, pokročilé datové typy (např. asociativní pole), dostupnost mnohých knihoven pro vývoj nebo třeba možnost využití regulární výrazů.

5 Budoucí vývoj

V této kapitole se pokusím nastínit, jak by mohl pokračovat vývoj tohoto projektu v budoucnosti. Vzhledem k tomu, že se jedná o poměrně rozsáhlou problematiku, je zde mnoho možností, jak dále tento program vyvíjet. Tato skutečnost byla zřejmá už od počátku vývoje, proto jsem se snažil program navrhnout tak, aby další vývoj byl možný.

5.1 Další možnosti harmonizeru

Třídu `HarmonizerSimple` bych rád rozšířil o možnost analyzovat stupnici, což by vedlo k tomu, že by bylo možno harmonizovat i modulující melodie. S touto okolností jsem počítal již na počátku vývoje, proto je u každého objektu `NoteHarmonics` atribut `"scale"`, který značí určitou stupnici v daném tónu.

Rovněž bude třeba implementovat možnost změny taktu. Z jistého úhlu pohledu nemusí mít změna taktu na harmonizaci příliš významný vliv. Akordy jsou přiřazovány většinou bez ohledu na dobu tónu v taktu. Při tvorbě figurovaného doprovodu se však jedná o poměrně zásadní záležitost. Není příliš vhodné použít například figuru ve valčíkovém rytmu pro čtyřdobý takt. Rovněž v polyfonii je pozice tónu v taktu velmi významná. Zde se berou v úvahu hlavně lehké a těžké doby. Na lehkých dobách jsou potom často dizonantní souzvuky, ty jsou však zpravidla rozvedeny do konzonancí na dobách těžkých.

Ke zdokonalení výstupu harmonizeru bude třeba přidat možnost použití dalších akordů. Současná verze počítá pouze s kvintakordy. Do harmonizace je však třeba zahrnout i jejich obraty (sextakord, kvartsextakord), dále také septakordy i s jejich obraty, uvažuji rovněž o akordech jiné než terciové stavby.

5.2 Použití CSP

V případě dalšího rozšiřování harmonizeru dojde nutně ke zvýšení počtu omezujících podmínek, které bude třeba brát při harmonizaci v potaz. O problému harmonizace obecně pak lze mluvit jako o CSP (úlohy s omezujícími podmínkami)

Perl je jazykem, který není navržen pro řešení těchto problémů. Proto by se mohlo stát, že budeme mít tolik omezujících podmínek, že řešení v Perlu by bylo nepřehledné a neefektivní (v krajmín případě dokonce takřka nemožné).

Výhodou jazyka Perl je však možnost využít rozhraní pro práci s jazyky, které jsou specializovány právě na řešení problémů spjatých s umělou inteligencí (jako třeba Lisp nebo Prolog). Toto rozhraní budu při nárustu počtu podmínek využívat.

5.3 Zavedení kontrapunktu

Kontrapunkt ve své nejjednodušší podobě už program obsahuje ve třídě HarmonizerMelodicBas, kdy je basový hlas veden do jisté míry nezávisle na melodii. Jedná se však o mnohem rozsáhlejší problém než pouze o snahu rozpohybovat doprovodný hlas různými melodickými tóny, čímž vznikne jistá nezávislost doprovodného hlasu na melodii. [7]

Vždyť kontrapunkt ve své pokročilé podobě vede k takovým postupům, které z několika mála motivů vytváří některé z vyšších polyfonních hudebních forem. Těmito formami jsou například passacaglia, ciaccona, chorálová předehra nebo fuga. To už by potom vznikl kompoziční program a mým primárním cílem je vytvořit harmonizer melodie. [8]

Přesto bych polyfonii v nějaké jednodušší formě do své práce rád zahrnul. Bude však třeba věnovat se rovněž hudební teorii, protože v tomto oboru nejsem příliš zběhlý.

5.4 Alternativní přístupy

O možnosti principů přístupu, který pracuje s převodem horizontály do vertikály jsem mluvil již v kapitole 1 (Principy harmonizace). Tuto možnost bych chtěl do svého projektu zahrnout, protože na jejím výstupu jsou často velmi zajímavé akordické postupy. Další výhodou je univerzálnost jejího použití. Nezávisí totiž na modu ani taktu.

Další alternativou jsou potom metody pracující s náhodou, dodekafonií, tvorbou vlastních stupnic apod.

5.5 Uživatelské rozhraní

5.5.1 Vstup

Další možností, na které je možné pracovat je vstup. Program připouští vstup pouze v textové formě, což může být pro uživatele nepohodlné. Jedním z možných řešení tohoto problému by mohlo být zavedení midi vstupu.

5.5.2 Grafické rozhraní

Program nemá žádné grafické rozhraní. Zatím jsem neuvažoval o jeho zavedení do projektu, protože se domnívám, že přednější záležitostí je funkčnost programu jako takového. Proto jsem se zaměřil raději na vývoj algoritmů, které mají na starosti harmonizaci apod.

V případě, že by měl být program masově používán, nebylo by zavedení grafického rozhraní problémem.

6 Závěr

Navrhl jsem a vytvořil program, který harmonizuje melodii. Program dává uživateli možnost výběru z několika harmonizačních postupů. Všechny tyto postupy vycházejí ze společného akordického schématu, které je tvořeno pomocí třídy `HarmonizerSimple`.

Snažil jsem se o to, aby program nekladl velké nároky na uživatele. To se mi podařilo částečně splnit. Uživatel má za úkol pouze zadat několik voleb pro výběr příslušného harmonizeru. Pro větší pohodlnost však chybí možnost midi vstupu.

Původní návrh zůstal z větší části zachován. Přesto však muselo dojít k jistým změnám, z nichž nejvýznamnější bylo zavedení nové třídy `Voice`. Původní návrh totiž nepočítal s tím, že by jednomu tónu melodie bylo přiřazeno více tónů v některém doprovodném hlase (melodické tóny apod.).
Pro implementaci jsem zvolil jazyk Perl. Zpočátku jsem měl z použití tohoto jazyka strach pro jeho pověst jazyka, ve kterém se tvoři nepřehledný kód. Nakonec jsem jej přesto zvolil na doporučení vedoucího své práce Ing. Michala Fapša. Důvodem byla dostpnost knihoven pro práci s hudbou a midi. Na Perl jsem si rychle zvykl a mé obavy se ukázaly jako neopodstatněné. Stačí dodržovat několik málo zásad štábní kultury a Perl je potom vhodným nástrojem pro Rapid Application Development (rychlý vývoj softwaru).

Program byl vyvíjen a testován v operačním systému Ubuntu 7.10. Během testování se potvrdilo, že program bez problémů zvládne zharmonizovat jednoduchou melodii. Výstupy a vstupy testů jsou ve složce `priklady` na přiloženém CD.

6.1 Přínos práce

Největším přínosem mé práce je to, že jsem vytvořil program, který je schopen vytvořit jednoduchou harmonizaci melodie. Midi výstup se dá jednoduše převést do notového zápisu a zahrát na klavír, což znamená, že program mohou použít s výhodou lidé, kteří sice doprovázejí na klavír, ale neumí vytvořit doprovodný part z melodie.

Výhodou mého programu je jednoduchost z hlediska uživatele. Ovládání programu je popsáno v nápovědě, kterou lze vypsat spuštěním programu s parametrem `-h`. Práci s programem se tak lze naučit poměrně rychle a během několika málo minut je možno mít výsledek práce programu před sebou. Program je teoreticky nezávislý na operačním systému, protože je vyvíjen v programovacím jazyce Perl. Prozatím jsem ho však testoval pouze v linuxu (Ubuntu 7.10) a ve Windows XP.

Program je navržen tak, aby bylo možno jej dále vyvíjet a tím docílit toho, aby se stal kromě obyčejného jednoduchého harmonizeru rovněž pomocníkem skladatele při komponování.

6.2 Budoucí vývoj projektu

Budoucímu vývoji projektu jsem věnoval celou pátou kapitolu. Během své práce jsem si totiž uvědomil, že se jedná o velmi rozsáhlou a spletitou problematiku. Každou melodii lze totiž zharmonizovat více způsoby, úloha nikdy nemá jediné řešení.

Po přečtené kapitoly 5 (Budoucí vývoj) se může zdát, že jsem tedy na začátku jednoho velkého projektu. Toto zdání je pravdivé. Rád bych totiž ve své práci pokračoval i v budoucnosti na své případné diplomové práci.

Zaměřit bych se chtěl zejména na řešení dané problematiky pomocí CSP. Tento způsob totiž umožňuje uvažovat při harmonizaci více omezeujících podmínek. Zajímavým ale velmi složitým přístupem je pohled na harmonizaci z hlediska kontrapunktu.

Literatura

[1] Silná, Ingrid: *Nauka o harmonii*. Konzervatoř P. J. Vejvanovského Kroměříž, 1992

[2] Zenkl, Luděk: *ABC hudební nauky*. Praha, Editio Bärenreiter, 2003

[3] Anders Torsen biography, http://cmr.soc.plymouth.ac.uk/tanders/index.htm

[4] Programování CSP, http://kti.mff.cuni.cz/~bartak/podminky

[5] Strasheela for musicians, http://cmr.soc.plymouth.ac.uk/tanders/writing.htm

[6] Perl - Wikipedie, otevřená encyklopedie, http://cs.wikipedia.org/wiki/Perl

[7] Silná, Ingrid: *Nauka o kontrapunktu*. Konzervatoř P. J. Vejvanovského Kroměříž, 1998

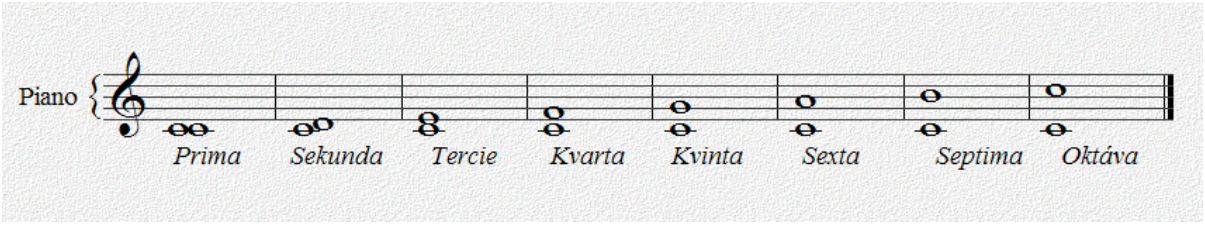
[8] Silná, Ingrid: *Nauka o hudebních formách*. Konzervatoř P. J. Vejvanovského Kroměříž 1999

Seznam příloh

Příloha 1. Slovníček pojmů

Na následujících řádcích jsem se pokusil vysvětlit pojmy související s hudbou, které jsem ve své práci použil. K jejich vysvětlení nepoužívám přesné definice uvedené v odborných publikacích, což může někdy vést k menším nepřesnostem (například bas nemusí být nutně harmonickým hlasem). Tyto nepřesnosti jsou však z hlediska mé práce zanedbatelné a jsou nutnou daní za stručnost, jednoduchost a srozumitelnost:

- alt: harmonický hlas nacházející se pod sopránem
- bas: nejspodnější harmonický hlas
- dizonance: souzvuk dvou nebo několika tónů, který působí drsně až nepříjemně
- dodekafonie: zrovnoprávnění všech dvanácti tónů temperované chromatiky v evropské hudbě
- dominanta: doškální kvintakord vystavěný na pátém stupni daného modu
- doškální akord: akord skládající se pouze z tónů daného modu
- doškální tón: tón vyskytující se v danému modu
- interval: výšková vzdálenost mezi dvěma tóny (viz Obr. P1-1)



Obr. P1-1 Základní intervaly

- kontrapunkt: nauka o polyfonním skladebném způsobu, znamená tedy spojení dvou nebo více samostatných hlasů v hudební celek
- konzonance: souzvuk dvou nebo několika tónů, který na náš sluch působí příjemně
- kvartsextakord: 2. obrat kvintakordu, basovým tónem je kvinta původního akordu
- kvintakord: trojzvuk složený ze dvou tercií
- lehká doba: doba, na které není v daném taktě přízvuk
- mikrointerval: interval menšího rozsahu než půltón
- modulace: přechod z jedné tóniny do druhé
- modus = stupnice
- polyfonie: způsob skladby založený na dvou nebo více samostatných, současně znějících, hlasech
- polytonalita: vztah k několika tonálním centrům současně
- sextakord: 1. obrat kvintakordu, basový tón se přesune o oktávu výš, novým basovým tónem je potom tercie původního akordu, která s novým sopránem tvoří charakteristickou sextu
- soprán: vrchní melodický hlas

- stupnice: stoupající nebo klesající uspořádaná řada tónů (často v rozmezí jedné oktávy)
- subdominanta: doškální kvintakord vystavěný na čtvrtém stupni daného modu
- tenor: harmonický hlas nacházející se nad basem
- těžká doba: doba, na které je v daném taktě přízvuk
- tónika: doškální kvintakord vystavěný na prvním stupni daného modu
- vnitřní hlasy: alt a tenor

Příloha 1. Obsah přiloženého CD

Součásti bakalářské práce v elektronické podobě:

- text práce ve formátu pdf
- text práce ve formátu odt
- programová dokumentace
- zdrojové soubory programu
- příklady vstupů a výstupů programu