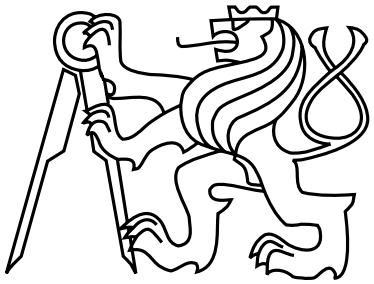
Na tomto místě bude oficiální zadání vaší práce

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studiijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Diplomová práce

iPad aplikace pro výuku irských tanců

Bc. Ondřej Macoszek

Vedoucí práce: Ing. Roman Berka, Ph.D.

Studijní program: Otevřená informatika, strukturovaný, Navazující magisterský

Obor: Softwarové inženýrství

7. května 2015

Poděkování

Rád bych poděkoval panu Ing. Romanu Berkovi, Ph.D; Ing. Janu Pleškovi a Ing. Janu Šedivému, CSc. za rady, trpělivost a podporu při realizaci této diplomové práce. Rovněž bych chtěl poděkovat své přítelkyni a rodině za podporu.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 7. 5. 2015 .....

Abstract

The target of this diploma thesis is to propose, design, implement and test a mobile application for teaching Irish dances. Within the thesis will be done an examination of methods for detection of music speed and musical beats, algorithms for slowing down the music while maintaining the pitch of tone and algorithm for generating suitable dance movements. Selection of algorithms will be implemented for a mobile device. The mobile application will call upcoming dance movements to selected music. Algorithm and application will be tested in a real environment.

Abstrakt

Cílem práce je navrhnout a implementovat iPad aplikaci pro výuku irských tanců. V rámci práce bude provedena rešerše metod pro detekci rychlosti hudby, detekci hudebních dob, pro změnu rychlosti hudby při zachování výšky tónu a pro výběr vhodných tanečních prvků. Vybrané vhodné algoritmy budou implementovaný pro iPad. Aplikace bude automaticky včas hlásit blížící se taneční prvky. Algoritmy i aplikace budou otestovány v reálném provozu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

1.1 Problematika

Tanec je pro mnoho lidí zábavou a relaxací, které se oddávají ve volném čase. Prospěšný je tělu i duchu člověka. Tanečník si procvičí paměť, postřeh, pohyb po prostoru, naučí se ovládat své tělo a pozná nové přátele.

V České republice neobvyklým druhem společenských tanců jsou Irské setové tance. Jde o tradiční lidové tance jejichž původ lze vystopovovat až k francouzským čtverylkám, které se díky vojákům vracejících se z napoleonských válek dostaly do různých koutů světa. V Irsku čtverylky padly na úrodnou půdu a místní lidé si tance přizpůsobili irské tradiční hudbě a zvykům. Podobu tance ovlivňovali také průběžně irští a evropští taneční mistři cestující z kraje do kraje. [1] Život v Irsku byl v některých obdobích obzvlášť krušný a zkoušený hladomorem - proto také mnoho rodin odcestovalo do různých koutů světa. Tradiční irská hudba a tanec cestoval také. Postupem času lidé na tradice zapomínali, ale někde se naopak udržely. Nejvíce původních tanců se dochovalo zejména v západní části Irska. Tanec i hudba se předávaly v rodinách z pokolení na pokolení a lidé vytvářeli také nové variace. Téměř nikdy dříve se tance nezapisovaly, protože každá vesnice měla obvykle dva či tři vlastní oblíbené tance, které tančili při společných setkáních. Tancovačky probíhaly buď ve stodolách nebo na křižovatkách cest, kde bylo vždy dost prostoru a pevná zem. Až v posledních padesáti letech začali lidé tance sbírat a zapisovat. Mnoho zajímavých tanců například zaznamenal na svých cestách tanečník a hudebník Pat Murphy, vytvořil dokonce celou serii knih [2] plnou podrobně popsaných tanců z nejrůznějších míst Irska i světa. Dále lze přepisy tanců zaznamenat třeba na webu Dance Minder [6] nebo případně ve formě videí na YouTube. Vyučují se často formou ukázky na tanečním semináři. Lidé si tedy nyní mohou užít mnohem více tanců než bylo možné dříve.

Irské setové tance se postupně stávají oblíbené také u nás v České republice. Svědčí o tom například rostoucí počet účastníků workshopů pražské Bernard's Summer School, nově vznikající lekce setového tance v několika českých tanečních skupinách a již několikátý ročník pražského setového víkendu. Taneční skupina Sona Sól dokonce irské setové tance vyučuje na ČVUT v rámci tělocviku a jako lekce pro univerzitu třetího věku při ČVUT.

Tančí se v párech. Jednotlivé páry jsou zformovány do čtveřic, čímž tvoří útvar o osmi tanečnících nazývaný "set". V této odrudě irského tance (setové tance) se zřídka používá jiný

útvar a taneční kroky jsou často ušity na míru přesně této osmici lidí. Elektronicky či písemně zaznamenaných tanců bude mezi 150-200. Tance se běžně pojmenovávají podle místa vzniku či původu. Každý tanec se skládá z několika, obvykle 4-5, figur. Každá figura vyžaduje hudbu o určitém počtu taktů a specifickém rytmu (reel, jig, polka...). Figura představuje posloupnost opakujících tanečních prvků. Drtivou většinu dílčích prvků tanečníci dobře znají a umí je zatančit.

Téměř všechny setové tance mají společnou nevýhodu, a tou je skutečnost, že jsou na první pohled poměrně složité, dlouhé a propletené. Tanečníci si běžně pamatují perfektně 3-5 oblíbených tanců. Zbývající tance dovedou zatančit jen s nápovědou. Není v lidských silách si zapamatovat přesně všechny existující tance a vědět vždy, který taneční krok bude ten další. Zde přichází na pomoc hlasatel, neboli anglicky caller, který s pomocí své paměti, případně poznámek/návodu, hlásí nahlas do hudby následující kroky. Hlášení probíhá velmi stručně a rychle, takže se hlásí často pouze název kroku a kdo jej má tančit. Na lekcích a workshopech se tance vysvětlují pomaleji, včetně všech detailů jako je držení rukou, podupávaní nohou a přesný směr pohybu tanečníků.

1.2 Úskalí a problémy

Hlášení kroků vyžaduje od hlasatele vysokou soustředěnost na hudbě a na návod k tanci. V hudbě musí sledovat jednotlivé takty a vědět kdy začít hlásit následující krok. Melodie irské tradiční hudby je strukturovaná do opakujících se celků po osmi taktech, takže tanečník se základním hudebním sluchem se dovede snadno orientovat v počtu uběhlých taktů. Návod k tanci [2] [6] je psán tak, aby detailně aby vystihl všechny zajímavosti tance (vysvětlení kroků a prvků, držení rukou, podupávání, přesný směr pohybu). Takto vyčerpávající popis je perfektní pomůckou pro výuku. Je však příhodnější znát také správně výstižné kratší názvy pohybů a tyto stručné názvy pak napovídat během tancovačky tanečníkům. Dalším problémem je, kdy přesně krok začít hlásit. Například pokud v tanci od 24. taktu začíná prvek "Swing", je vhodné zahlásit prvek dostatečně včas, aby na něj dovedli tanečníci zareagovat. Zároveň by se prvek neměl hlásit příliš brzo, aby to nebylo matoucí. Správné načasování hlášení je výzva zejména pro začínající hlasatele. Nezkušení hlasatelé obvykle nedovedou napovídat stručně, výstižně a vhodně včas. Omyly v hlášení vedou ke zmatku a tanečníci si pak figuru tolik neužijí.

Učitelé tanec obvykle znají nazpaměť a dovedou hlásit kroky perfektně včas. Často však musí sledovat tanečníky, zda se některý z nich neztratil a případně mu přispěchat na pomoc. V tomto momentě může zapomenout zavolat následující krok a tím tak ohrozit ostatní tanečníky. Pro plynulost lekce i pro zapamatování opravy chyby u tanečníka je preferovanější, když jej učitel opraví ihned, nikoli až celý tanec všichni dotančí. Jsou zde tedy kladeny nároky šikovnost učitele, aby zvládl zároveň opravovat chybující tanečníky i hlásit blížící se kroky.

Dalším nezbytností je, aby hlasatel volal nahlas. Pokud dovede pracovat s bránicí, pak je bez větší námahy hlas slyšet i při hlasitě hrající hudbě. S tímto se potýkají opět především začínající hlasatelé, případně ti, kteří nemají větší zkušenosti se zpíváním. Zkušenější hlasatele však může zradit hlas během nemoci. Pokud nejsou nahlas slyšet blížící se kroky, můžou tanečníci začít plést kroky.

Výběr správné hudby může také skýtat drobná úskalí. Figury jednotlivých tanců vyžadují hudbu o přesném počtu taktů a určitém rytmu. Někdy se stává, že hudba nemá dost taktů a přestane hrát dříve než tanečníci stihnou tanec dokončit. Stává se i opačný případ, kdy hudba hraje déle, což tolik nevadí, ovšem je příjemnější když závěrečný poslední dup ladí s posledním tónem v melodii. Existují speciálně připravená alba, u kterých je spočítán počet taktů a je tak snadnější vybrat tu správnou skladbu. U nové hudby, či hudby netanečních kapel je nutné takty spočítat, což je velmi zdlouhavý a únavný proces. Rytmus je u názvu skladby často jasně zmíněn. Pokud rytmus znám není, je nutné aby zkušenější tanečník nebo hudebník rytmus poznal. V irské tradiční hudbě se nejčastěji setkáme s rytmy reel, jig, polka, hornpipe, slide. Mají svá hudební specifika, ale často je lze odlišit pocitem z poslechnuté skladby. Reel má pravidelnější zvuk, jig a polka jsou houpavější, hornpipe je pomalejší a těžké doby lákají k výskoku. Existují také pomocné říkanky, které když promlouváme do hudby, ladí nejlépe jen s určitým rytmem. S určením typu rytmu mají ovšem obtíže lidé bez hudebního sluchu.

Pokročilí tanečníci do tance zapojují takzvaný "battering". Jde o podupávání do rytmu velmi podobné stepu. Vychází ze starého stylu solového irského tance Sean Nós, který je charakteristický speciálním podupáváním bez zvedání nohou příliš vysoko nad zem a volnějším pohybem rukou. Každý tanečník má vlastní osobitý styl a běžně improvizuje po celou dobu tance. Běžně se tančí na velmi malém prostoru, například vysazených dveřích hospody, či na barelu. Vystoupení jsou pro obecenstvo velmi přitažlivé a poutavé. Existuje několik obecně známých kroků pojmenovaných podle místa původu (například Clare, Connemara, Roscommon), které zkušenější tanečníci rádi přidávají podle chuti do jednotlivých částí tance. Není však vždy úplně jednoduché je do tance zakomponovat, tak aby tanečník neohrozil ostatní a průběh tance. Pro vyučování těchto kroků je užitečné tančit na pomalejší hudbu. Bohužel ne vždy je pomalejší hudba k dispozici, proto tanečníci využívají aplikace pro zpomalení hudby.

Většina slov používaných pro napovídání je anglicky. Téměř ve všech zemích se hlásí kroky anglicky. Ovšem v některých zemích či místech jsou tanečníci zvyklí na odlišné názvy tanečních prvků. Mohlo by být tedy rovněž užitečné mít možnost zvolit lokalizaci tance pro určitou oblast a vyjít tak vstříc tanečníkům.

Posledním problémem, který není tolik překážkou při běžných setových tancovačkách, jako spíš lákavou dovedností, je vymýšlení kroků na míru hudbě. Užitek by si taková dovednost našla například při trénování rychlosti postřehu na volání tanečních kroků, ale také při vytváření tanečních choreografií. Taneční choreografie je často líbivá právě proto, že změny kroků se dějí s ohledem na vývoj hudby. Hudba může mít tiché části, hlasité, rychlé, pomalé nebo vzájemně podobné části. Dobrý choreograf dokáže těchto aspektů využít při tvorbě nového tance.

1.3 Možná řešení

Pro hlasatele by bylo řešením delegovat hlášení na někoho jiného, kdo zvládne sledovat takty v hudbě a zároveň hlásit včas blížící se taneční kroky. Mým návrhem je vytvořit aplikaci, která by dokázala hlasatele v případě potřeby zastoupit a zajistit tanečníkům spolehlivé hlášení kroků. Jelikož se setové tance můžou tančit nejen doma v kuchyni, ale i v tanečním sále či restauraci bude ideální, aby taková aplikace byla mobilní. Případně, aby mobilní

zařízení po připojení k projektoru dovedlo také navíc zobrazovat blížící se kroky na větší plátno.

Výběr hudby podle počtu taktů a rytmu by mohla pomoci mobilní aplikace díky automatické detekci hudebních dob. Z hudebních dob lze dopočítat takty ve skladbě. Aplikace by pak uživateli mohla nabídnout inteligentní filtrování skladeb podle požadovaného počtu taktů a rytmu.

Pro výuku batteringu tanečníci často využívají existující mobilní aplikace pro zpomalení hudby. Není úplně prioritou zakomponovat tuto funkcionalitu do výsledné aplikace diplomové práce. Avšak rád bych prozkoumal možnosti dostupných knihoven, které lze pro zpomalení použít. Při návrhu uživatelského rozhrané by bylo dobré na toto myslet, a počítat s prostorem kam tuto funkci s dalším vývojem aplikace integrovat.

Lokalizace nápovědy by přišla vhod některým tanečním komunitám, které jsou zvyklé na odlišné názvy některých tanečních prvků. Zajistit tuto lokalizaci by bylo možné prostřednictvím zapisu tance v aplikaci. Zápis by mohl umožnit definovat odlišné názvy pro různé lokalizace. Termín lokalizace by zde měl být chápan volněji - ne na úrovni jednotlivých států, ale spíš na úrovni jednotlivých míst třeba i v rámci jedné země.

Vymýšlení kroků na míru hudbě by opět mohla zajistit aplikace, která by na vstupu očekávala analyzovanou hudbu rozdělenou do podobných částí a množinu kroků, z nichž má být nová posloupnost kroků k příslušné hudbě vymyšlena. Tato aplikace nemusí být nutně mobilní, protože bude sloužit především jako pomůcka učiteli. Tanec vymyšlený aplikací by choreograf ještě potřeboval doplnil o vlastní nápady.

Kapitola 2

Specifikace cíle

Identifikovaná úskalí problematiky hlasatele v irských setových tancích, určitě nebudou specifická jen pro tento druh tance. Myslím, že by bez větších obtíží mělo být možné nalezená řešení přenést a využít také v jiných odrůdách tance, tedy nejen tance irského. Lidové tance jsou velmi různorodé a věnovat se všem by vyžadovalo všechny podrobněji poznat a pokusit se řešení zobecnit tak aby vyhovovalo většině druhům, což by rozsahem nezbytné práce daleko přesáhlo časové možnosti této diplomové práce. V rámci diplomové práce tedy zaměřím veškeré úsilí zejména na vyřešení problémů konkrétně v irských setových tancích, které osobně dobře znám. Díky mé volnočasové zálibě ve hře irské tradiční hudby na housle se dovedu rovněž snadněji orientovat v melodii skladeb a rytmů typických pro irskou hudbu (reel, jig a další).

2.1 Vymezení požadavků

Role hlasatele v irském setovém tanci je důležitá, ovšem některé problémy jeho práci znesnadňují, proto by ocenil, kdyby v určitých případech mohl delegovat své povinnosti na mobilní aplikaci. Aplikace by tedy měla být schopná zastoupit praktické činnosti hlasatele. Zejména musí zvládnout včasné hlášení blížících se kroků do hudby, právě tento úkol má nejvyšší prioritu, protože na něm závisí spokojenost tanečníků ze správně zatančeného tance (hlavní úloha hlasatele), a je tedy klíčové se mu důkladně věnovat. Zbylé problémy nastíněné v předchozí části nejsou pro hlasatele tolik palčivé, ovšem jejich vyřešení by hlasateli usnadnilo práci - bude zajímavé je také prozkoumat a myslet na ně při návrhu aplikace. Při návrhu uživatelského rozhraní budu především myslet prioritně na potřeby cílové skupiny hlasatele a sekundárně na potřeby tanečníku.

Hlavní a nejdůležitějším cílem práce je prozkoumaní metod pro detekci rychlosti hudby, zjištění přesných míst jednotlivých hudebních dob a dopočítání taktů na základě hudebních dob a vybraného rytmu. Z nalezených metod a algoritmů vyberu případně sestavím ideální algoritmus, který bude vhodně řešit napovídání blížících se kroků do hudby.

Z vedlejších problémů bude užitečné nejprve, aby aplikace učitele informovala u jednotlivých hudebních skladeb kolik taktů obsahují a v jakém jsou rytmu. Dále za druhé se pokusím zjistit možné způsoby zpomalení hudby při zachování výšky tónu, což ocení užitelé při výuce složitějších tanečních kroků (battering). Za třetí budu při návrhu zápisu tance myslet

na možnou lokalizaci v nápovědách hlášené terminilogie. Posledním vedlejším problémem je prozkoumání možností generování tanečních kroků na míru hudbě.

S ohledem na potřeby cílové skupiny navrhnu vhodné uživatelské rozhraní mobilní aplikace. Pomocí low-fidelity prototypů a jejich testování v rámci uživatelského testování se budu snažit doiterovat k ideální podobě rozhraní aplikace, která bude nejlépe vyhovovat cílové skupině uživatelů.

Cílovou aplikaci předem navrhnu s ohledem na zásady a zkušenosti softwarového inženýrství. Identifikuji případy užití a jejich protějšky v modelu aplikace. Dále navrhnu jakým způsobem bude řídící kód aplikace organizován, jak bude pracovat s modelem aplikace a co bude zajišťovat uživatelské rozhraní.

Během realizace se zaměřím na implementaci části aplikace sloužící k včasnému hlášení tanečních kroků do vybrané hudby. Aplikace bude obsahovat několik připravených tanců, které po otevření zobrazí návod k tanci, následně dovolí vybrat hudbu na kterou se bude tancovat. V této hudbě algoritmus automaticky rozpoznaná hudební takty, díky čemuž bude aplikace vědět, kdy hlásit blížící se taneční kroky. Během hlášení bude obrazovka ukazovat textově popis aktuálních a následujících kroků. Rovněž ověřím jaký úspěch bude mít aplikace v reálném provozu.

2.2 Cílová skupina

Cílovou skupinou jsou tanečníci irských setových tanců. V rámci této skupiny je většina znalá základních pojmů a zaběhnutých názvů kroků. Začátečníci obvykle terminologii pochopí během prvních tréninků. S ohledem na tento poznatek, lze při hlášení hlásit skutečné zavedené názvy tanečních kroků. Tanečníci jsou na názvy zvyklí jak z tréninků, tak z tancovaček i z textových popisů tanců. Setovým tancům se přibližně z 80 procent věnují lidé starší nad 50 let, proto při hlášení je nutné volit hlasitost vyšší než je hudba a při vizuálním zobrazení blížících se kroků vybrat větší a tučné písmo na kontrastním pozadí. Mladší tanečníci (20 - 30) jsou většinou spíše v Čechách, případně ještě mladší tanečníci (pod 15 let) tančí setové tance přímo v Irsku.

Primární cílovou skupinou jsou hlasatelé a taneční učitelé, protože jsou to oni koho aplikace zejména zastupuje. Proto při ovládání lze brát ohled na širší znalosti uživatele například v oblasti hudby a nezbytností ke každému tanci. Učitel například dovede rozlišit rytmus skladby, tedy zda jde o reel, jig, polku, slide nebo hornpipe. Učitel také ví, že ke každému tanci je nutné vybrat hudbu o určitém počtu taktů. Uživatelské rozhraní bude tedy dobré tvořit tak aby učiteli co nejvíce ulehčilo jeho práci. Jde o pokročilé tanečníky s dlouhodobějšími zkušenostmi v tanci. Často se věnují vyučování tance pro veřejnost, vedou pravidelné lekce i nárazové workshopy. Sami se v tanci průběžně vzdělávají a rozšiřují si obzory příležitostnými návštěvami jiných lektorů. Mají zkušenosti s vysvětlováním dílčích tanečních prvků, dovedou zastoupit v páru místo pána i dámy, orientují se v taneční terminologii, obvykle si pamatují zpaměti více tanců než většina tanečníků. Při vysvětlování běžně pracují s vytištěným návodem tance, který používají jako oporu pro své vysvětlování a mají v textu často vyznačené připomínky či postřehy k důležitým problematickým částem tance. Učiteli by aplikace měla sloužit jako náhrada za taneční poznámky nebo knížky, proto by bylo dobré kdyby obsahovala i detailnější popis tance užitečný při vysvětlování. Případně

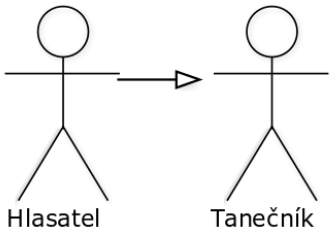
kdyby aplikace dovolila uživateli zaznamenat vlastní postřehy při výuce tance. Nezbytný je pro hlasatele dobrý hudební sluch, díky kterému se dokážou orientovat v melodii hudby, poznat doby, takty i rytmus. Někteří hlasatelé jsou rovněž dobrými muzikanty. Například pan Pat Murphy dříve hrál k tanečním soutěžím na box (irská diatonická harmonika, podobná v čechách známé heligonce). Hlasatelé jsou na tancovačkách žádaní stejnou měrou jako živá kapela (ceilí band).

Sekundární cílovou skupinou jsou běžní tanečníci, kteří si přišli užít tancovačku nebo zavítali na některou lekci setového tance a chtějí se zdokonalit. Zkušenosti tanečníků se můžou značně lišit. Někteří ovládají své pohyby dokonale, jiní se kroky teprve učí a zdokonalují je. Navštěvují pravidelné lekce, nárazové taneční semináře i tancovačky. Taneční údálosti se konají napříč Evropou i celým světem. Je oblíbené výlety za tancem spojovat s dovolenou a blíže poznat okolí i kulturu místa konání. Věk tanečníků je většinou velmi různý, v průměru je věk spíše vyšší (50-70). Tanec je pro ně volnočasovou aktivitou, kterou si zamilovali, místem setkání s přáteli, příležitostí si procvičit mysl i tělo, nebo dovolenou za poznáním cizích míst.

Rád bych navíc ještě vymezil, kdo cílovou skupinou naopak není. Například kapela či hudebníci nejsou cílovou skupinou, protože během výuky jsou k dispozici pouze v reprodukované podobě z hudebních nahrávek. Dále obecenstvo a přihlížející nejsou prioritou, protože se tance přímo neúčastní a hlasatel s nimi nijak nepracuje.

2.3 Případy užití

Případů užití aplikace bude několik, užitečné různou měrou pro různé tanečníky. Přesto bych je zde rád všechny vyčetl a definoval. Na základě uživatelského testování pak provedu rozhodnutí, které z těchto případů užití jsou skutečně nejpodstatnější pro cílovou skupinu. Nyní totiž není úplně jasné a samozřejmé, které případy mají vyšší prioritu a mělo by jim být dáno v uživatelském rozhraní většího prostoru. S určením priority případů užití pro cílové uživatele mi pomůže uživatelské testování, které provedu v dalších částech této práce.



Obrázek 2.1: Hlasatel přebírá také případy týkající se tanečníka, protože se často nachází i v jeho roli. Mezi těmito actory je tedy vztah dedičnosti

2.3.1 Actors (role uživatelů systému)

Hlasatel reprezentuje primární cílovou skupinu. Problémy řešené diplomovou práci se týkají především hlasatele a jeho potřeb během výuky irských setových tanců. Hlasatel je však zároveň tanečník a budou se ho týkat rovněž případy související s tanečníkem - proto má v diagramu vztah dedičnosti. Předpokládám, že budou aplikaci přímo ovládat prostřednictvím uživatelského rozhraní.

Tanečník reprezentuje sekundární cílovou skupinu uživatelů aplikace. Budou vnímat spíše výstupy, které bude aplikace produkovat (nápověda a hudba). Nepředpokládá se, že budou aplikaci přímo ovládat.

2.3.2 Případy týkající se tance

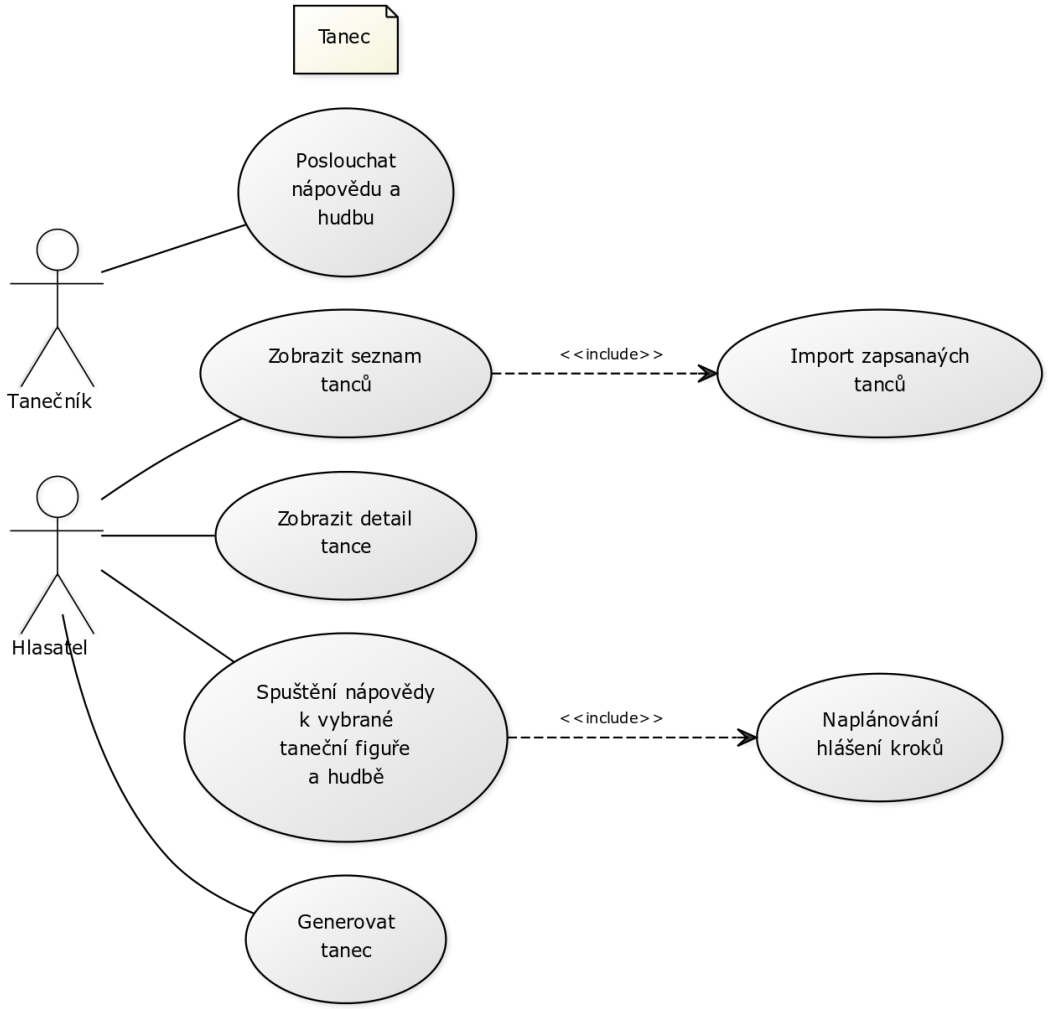
Poslouchat nápovědu a hudbu Tanečník během tance uslyší hudbu, která má správný rytmus a nezbytný počet taktů pro zatančení celé hlasatelem vybrané figury. Do hudby bude aplikace hlásit blížící se taneční kroky. Je důležité, aby tanečník nápovědu slyšel dostatečně včas a stihl na ni zareagovat. Hlas nápovědy by měl být zřetelný a pronikat hrající hudbou.

Zobrazit seznam tanců Hlasatel bude potřebovat zobrazit seznam tanců, které jsou v aplikaci připraveny a popsány. Pro snadnější hledání bude tance seřazeny sestupně podle názvu. Ze seznamu si může hlasatel vybrat tanec, ke kterému si přeje zobrazit detailnější popis a který bude chtít případně tančit. Kliknutím na název tance se zobrazí detail tance.

Import zapsaných tanců Před zobrazením seznamu tanců aplikace automaticky na pozadí zkontroluje, zda nejsou dostupné nové tance a načte nejnovější tance. Při instalaci bude dostupných několik předdefinovaných tanců. Upravit definice tanců uživateli aplikace umožní buď prostřednictvím jednoduché editace textového popisu, případně pomocí stažení definice z webového serveru.

Zobrazit detail tance Detail tance bude obsahovat seznam jednotlivých figur tance. Informuje uživatele o nezbytném počtu taktů v hudbě a rytmu pro každou figuru. Výchozím výběrem bude první figura a aplikace rovnou zobrazí podrobné kroky k této figuře, včetně informace na kolik taktů se mají tančit a kdo je má tančit. V detailu bude možné se přepínat mezi jednotlivými figurami. Dále bude možné vybrat hudbu pro figuru a jakmile bude hudba pro figuru vybrána dovolí aplikace uživateli spustit hudbu včetně volání kroků.

Spuštění nápovědy k vybrané taneční figuře a hudbě Na obrazovce pro volání bude největší prostor vyhrazen zobrazení aktuálního kroku, včetně drobné poznámky kdo jej má tančit a na kolik taktů. V jiné části obrazovky bude ne tolik poutavě zobrazen následující krok. Tyto informace se budou automaticky přepínat v průběhu hudby a tance. Bude možné zde hudbu pozastavit a opětovně spustit.

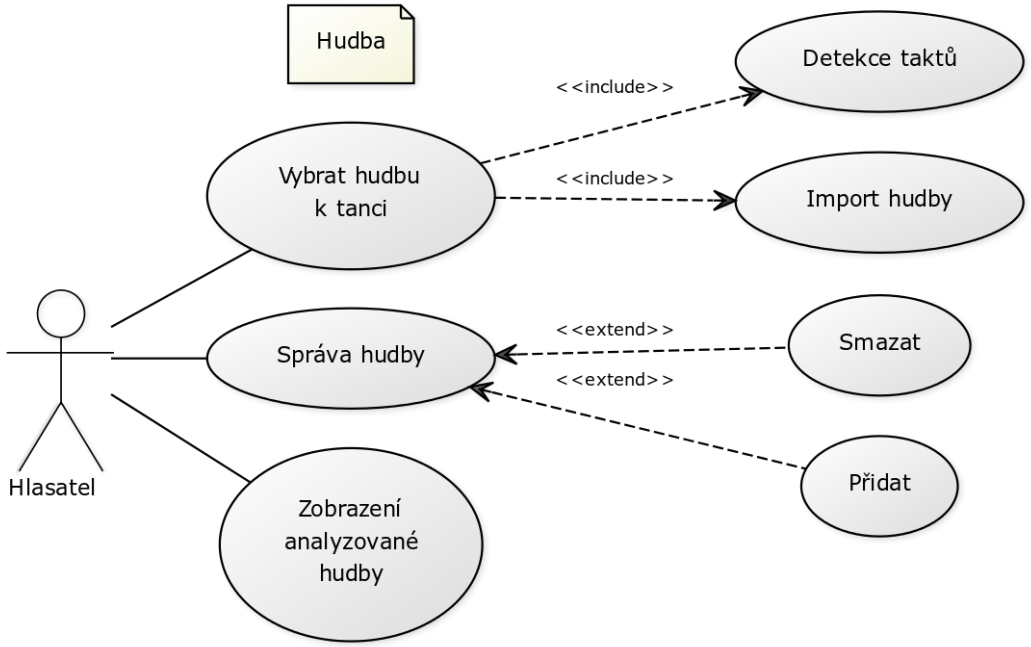


Obrázek 2.2: Use case diagram pro případy týkající se tance

Naplánovat hlášení kroků Před zahájením hlášení nápovědy k taneční figuře aplikace automaticky naplánuje časy hlášení pro jednotlivé kroky. Aplikace musí zajistit dostatečný předstih v hlášení, aby tanečníci byli schopni na hlášení zareagovat. Rovněž by měla hlášení kroků doplnit o vhodné další informace, například kdo má následující kroky tančit pokud se v návodu k tanci píše, že má další kroky tančit jiný pár tanečníků.

Generovat tanec Generátor kroků umožní uživateli vybrat některou z uložených hudebních skladeb a vybrat množinu tanečních kroků, které se použijí při generování. Po spuštění generování se zobrazí náhled posloupnosti kroků podobný jako v detailu figury (viz výše) včetně informace kdo má který krok tančit a na kolik taktů.

2.3.3 Případy týkající se hudby



Obrázek 2.3: Use case diagram pro případy týkající se hudby

Vybrat hudbu k tanci Výběr hudby v detailu tance ukáže novou obrazovku, kde lze vybrat hudbu přítomnou na mobilním zařízení. Tuto vybranou hudbu aplikace uloží a přepošle k analýze pro nalezení hudebních dob a taktů. Pokud takty najde a úspěšně uloží dovolí uživateli tuto hudbu vybrat k tanci.

Detekce taktů Aplikace automaticky na pozadí zahájí analýzu hudební skladby a pokusí se co nejlépe v melodii odhalit hudební doby. Následně podle dob a informace o rytmu

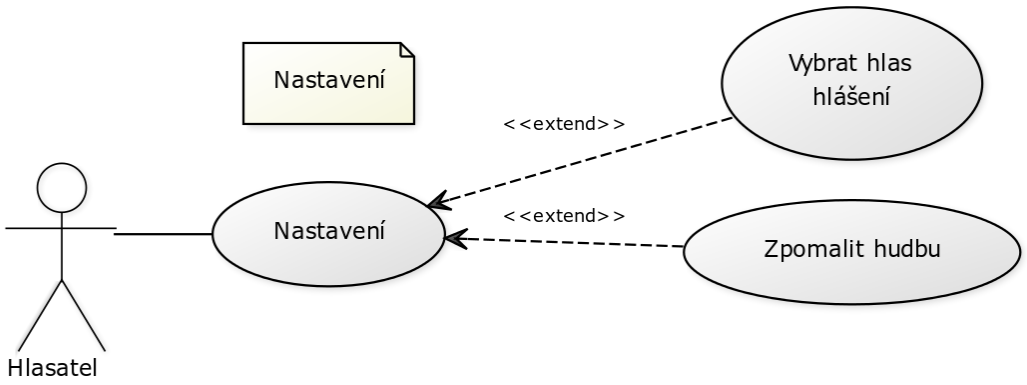
dopočítat takty pro tuto skladbu. Pak je hudba připravena pro použití k nápovědě.

Import hudby Aplikace umožní uživateli prostřednictvím rozhraní přistoupit do databáze hudby v telefonu (iPod) a vybrat pomocí filtru a hledání skladby, které by uživatel rád importoval do aplikace. Vybraná hudba bude zkopírována do adresáře aplikace, aby bylo možné s hudbou dále pracovat.

Správa hudby Obrazovka pro správu hudby umožní vypsat seznam všech skladeb, které byly analyzovány a uloženy prostřednictvím aplikace. Jednotlivé skladby je možné smazat případně zobrazit náhled akustické vlny hudby s vyobrazením nalezených hudebních dob.

Zobrazení analyzované hudby Obrazovka vykreslí akustickou vlnu vybrané skladby. Scrolování obrazovky bude možné posouvat náhled celé vlny. Výraznou značkou bude nastíněno, ve kterém čase vlny se nachází nalezené hudební doby.

2.3.4 Případy týkající se nastavení



Obrázek 2.4: Use case diagram pro případy týkající se nastavení

Vybrat hlas hlášení Uživatel bude moci aplikaci upřesnit informace o lokalizaci hlášení a vybrat hlas používaný pro hlášení nápovědy. Hlas může být generován hlasovým syntetizátorem nebo případně pomocí výběru předem připravených nahrávek používaných slov a vět.

Zpomalit hudbu Uživatel bude moci zvolit rychlost přehrávání hudby. Aplikace hudbu zpomalí podle volby uživatele a přizpůsobí také časování hlášení nápovědy k tanci.

Kapitola 3

Analýza

3.1 Rešerše domény

3.1.1 Hudební teorie

Hudební skladba je tvořena notami, které vnímáme jako melodii. Noty mohou jsou uskupené do taktů, čímž vyjadřují krátký časový úsek skladby, ve kterém se střídají silné a slabé stejně dlouhé doby. Časový úsek se označuje na začátku notové osnovy buď jako dvě čísla oddělená lomítkem, nebo jako dvě čísla pod sebou. Nejčastější používanými jsou časy 4/4 (čtyřčtvrťové), 2/4 (dvoučtvrťové) a 6/8 (šestiosminové). První číslo (nad lomítkem) znamená počet dob v taktu. Druhé číslo (pod lomítkem) popisuje typ doby v taktu - tedy zda je doba v taktu počítána jako čtvrťová či například jako osminová. Každý takt se skládá z určitého počtu těžkých a lehkých dob. Na těžké doby je pro Evropany přirozené podupávat nohou.

3.1.1.1 Irská tradiční hudba

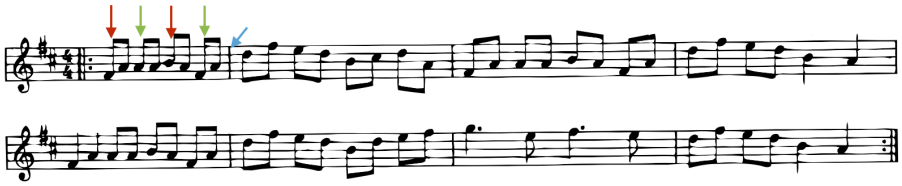
Hudba označovaná jako irská tradiční často zahrnuje nejen skladby původně irské, ale často i oblíbené původem skotské, anglické či jiné. Podoba se v průběhu let vyvíjela a stále vyvíjí. Je pozoruhodné, že irská tradiční hudba nevymizela tolik moc jako tradiční hudba v jiných zemích. Je možné, že zachování tradic pomohla větší izolace země od historických událostí na kontinentu a převážně zemědělský charakter ekonomiky země. Irský tanec, písně a příběhy se tradičně předává zejména ústně. Ne jinak je tomu u hudby, žáci se často učí nové skladby poslechem hraní učitele. Samozřejmě mnoho skladeb bylo přepsáno do notového záznamu a je tedy mnohem snadnější naučit se melodii. Málokdy se však do not zapisuje také způsob frázování a speciální důrazy, je tedy velmi důležité při výuce irské hudby od určité chvíle odprostit se od notového záznamu a trénovat hraní společně se zpomalenou nahrávkou či s učitelem. Muzikant si tak vytříbí cit pro poslech a dovede pak zahrát skladby autenticky irsky. Mezi nejběžnější nástroje pro hraní irské hudby patří housle, knoflíkový akordeon (box), banjo, kytara, bodhran, flétna nebo uilleann pipes (loketní dudy). Muzikanti se rádi setkávají na takzvaných sessions v hospodách nebo u přátel doma, kde si hrají pro potěšení oblíbené skladby. Pro tancovačky jsou velmi žádané kapely (tzv. ceilí band), které mají secvičený repertoár skladeb připravených pro tanečníky. Jde obvykle několik skladeb

sloučených za sebe tak, aby vystačily na celou délku tance. Některé výborné kapely dovedou upravit i délku a přechody mezi jednotlivými skladbami tak aby korespondovaly se změnami kroků v tanci, čímž jen posílí zážitek z tance.

3.1.1.2 Rytmy irské hudbě používané v setových tancích

V irské hudbě rozlišujeme také několik rytmů, které ovlivňují způsob jak se do nich tančí a vyžadují odlišné kroky. Těmito rytmy je reel, polka, hornpipe, jig, slide a občas i některé další jako například fling. Příbuznými co do časového předznamenání jsou reel (4/4), polka (2/4) a hornpipe (4/4). A pak také jig (6/8) a slide (12/8). [14] Existují pomůcky ve formě vět ("říkanek"), které lze opakovaně přeříkávat do hrané hudby a uchem se pak ujistit, zda říkanka ladí s rytmem. V případě upřesnění složitějšího rytmu, lze použít více říkanek a vylučovací metodou si odvodit správný rytmus. Oblíbenost a četnost použítých rytmů závisí na původu tance. Například tance ze severu irska jsou nejčastěji postaveny z reelu nebo jigu. Tance ze západního a jižního irska zase preferují polky a slides.

Reel (4/4) je charakteristický tím, že v jednom taktu má čtyři doby, z nichž jsou dvě doby (první a třetí) těžké a další dvě doby (druhá a čtvrtá) jsou lehké. Pocitově má reel spíše pravidelnější zvuk, působí jako nepřetržený tok not, který nečeká. Obecně se má za to, že reely jsou původem skotské, ovšem staly se velmi oblíbenými také v Irsku. Říkanky použitelné pro tento rytmus jsou například: "Watermelon, watermelon, watermelon...", "Double-decker, double-decker", či "THIS is how the REEL goes, THIS is how the REEL goes, ...".

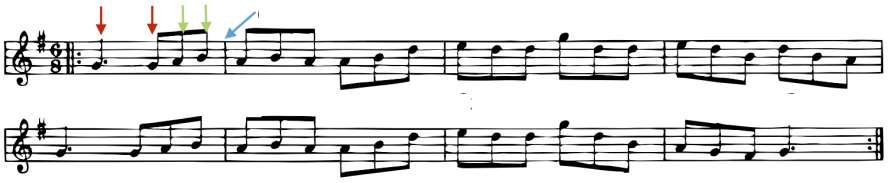


Obrázek 3.1: Reel: červeně těžké doby, zeleně lehké doby, modře takt

Hornpipe (4/4) má podobnou strukturu jako reel, rovněž má stejné časování. Ovšem hraje se pomaleji než reel. V setových tancích se hornpipe nejčastěji používá u posledních figur tance, protože má pomalejší oddychovou melodii ideální pro nabrání sil a rozloučení s tanečními partnery. Do irské hudby přišel hornpipe z Anglie. Říká se o něm, že je to námořnický rytmus - je pro něj docela charakteristické, že vždy první tón v taktu láká tanečníka k výraznému povyskočení na tuto dobu (tento výskok se často využívá v tanečních krocích pro hornpipe).

Polka (2/4) je časováním podobná opět reelu, ale hraje se mnohem rychleji. Pocitově jsou polky houpavější. Přišly do Irska z Německa/Čech a byly přizpůsobeny místní tradici - zní tedy trošku jinak než jsme v Čechách zvyklí.

Jig Pro jig (6/8) je charakteristické, že v jednom taktu má šest dob, z nichž jsou první a čtvrtá doba těžké a další zbývající noty lehké (druhá, třetí, pátá, šestá). Tento rytmus je považován za ryze irský. Lze rozlišit ještě několik druhů jigů (double, single, slip), které se ovšem nepoužívají v setových tancích.



Obrázek 3.2: Jig: červeně těžké doby, zeleně lehké doby, modře takt

Slide (12/8) je strukturou podobný spíše jigu. Ovšem poznat jej z poslechu je občas docela obtížné. Nejčastěji se jedná o slide pokud jsme při poslechu váháme mezi polkou a jigem.

3.1.1.3 Struktura skladeb irské tradiční hudby

Typická irská skladba je obvykle stavěna ze dvou příbuzných částí (označovaných A, B), z nichž každá je nejčastěji o délce osmi taktů. Na konci každé části je většinou repetice, takže se hraje dvakrát než přejdeme k další části. Skladba se tedy hraje například tak, že posloupnost částí plyne AABB. Tato skladba se pak celá běžně opakuje třeba třikrát za sebou.

Muzikanti běžně při hraní propojují dohromady více skladeb stejného rytmu a vytvoří tak delší hudební celek. Například tedy můžou hrát 3x skladbu A, 3x skladbu B, 3x skladbu C.

3.1.1.4 Hudba pro tanec

Při výběru hudby pro tanec hudebníci využívají hojně propojování několika skladeb za sebe. Počet opakování jednotlivých skladeb si určí tak aby celková délka hudby měla počet taktů potřebných pro zatančení celé figury. Takto za sebe poskládaných skladbám se běžně říká "set". Kapely také někdy dovedně pracují s klidností, či chytlavostí jednotlivých skladeb, a tak například na doprostřed setu vloží chytlavější skladbu, která rozproudí naladu tanečníků.

U taneční hudby je důležité, aby perfektně dodržovala stanovené tempo. Jinak hrozí, že se tanečníci v hudbě ztratí a začnou plést délku jednotlivých tanečních kroků. Taneční kapely tedy často používají bicí, nebo basovější nástroje, aby dovedli udržet tempo. Je zde tedy pro kapely menší prostor pro improvizaci s tempem hraní, ovšem dobré kapely dovedou krásně improvizovat a na správných obohatit základní melodii ornamenty.

3.1.2 Irské tance

Irské tance zabírají několik druhů. Světově nejznámější je step dancing, který se tancuje buď ve stepkách (hlasité) nebo v měkkých botách (jsou tiché), může se tancovat sólově i ve skupinách. Běžně jej uvidíte na představeních Lord of the Dance nebo na irských tancovačkách (i v ČR). Zajímavé je, že se téměř nepoužívají ruce při tanci - pouze pro držení s partnerem, takže po většinu času mají tanečníci ruce rovně při těle. Hlavní roli tedy hrají nohy a jejich zvuk nebo pohyb po pódiu. Existuje ještě starý styl Sean Nós, kde se můžou pohybovat volněji a různě zdobit tanec - často se tancuje na malém prostoru, třeba na barelu. Pro irský tance se používá nespočetně mnoho tanečních prvků, z nichž každý lze obvykle provést za jeden až dva takty. Taneční prvky také můžou být různě náročné. Navíc na náročnosti může přidat i určitá specifická posloupnost těchto prvků těsně za sebou. U společenských tanců se termín taneční prvek používá spíše pro označení pohyb páru po prostoru (nejde v první řádě tolik o kroky, které dělají nohy). U solových tanců se jako taneční prvek označuje spíše konkrétní pohyb části těla tanečníka.

3.1.2.1 Druhy irského tance

Step dancing Slovo "step"v tomto druhu irského tance znamená v krok a označuje jak tanec ve stepkách, tak tanec v měkkých botách. Jde převážně o sólový tanec, který je možné vidět na soutěžích či ve světoznámých irských tanečních představeních. Tento druh tance je velmi fyzicky náročný a pro správné provedení většiny prvků je důležité trénovat také ohebnost těla. Ruce se při tanci nepoužívají a zůstávají po celou dobu vždy rovně při těle.

Ceilí a figure dancing Jde o společenské tance a taneční choreografie pro více tanečníků, kteří během tance používají techniku natrénovanou ve Step dancingu. Tanečníci se obvykle formují do řad, zástupů či do kruhů. Není pravidlem, že by v každém ceilí tanci bylo nutné vytvořit pár pána a dámy. Tance obvykle nejsou příliš časově náročné, jsou však fyzicky obtížnější na provedení. Ruce tanečníků opět zůstávají nehybně při těle. Běžně se lze s tímto druhem setkat na vystoupeních či na některých tancovačkách.

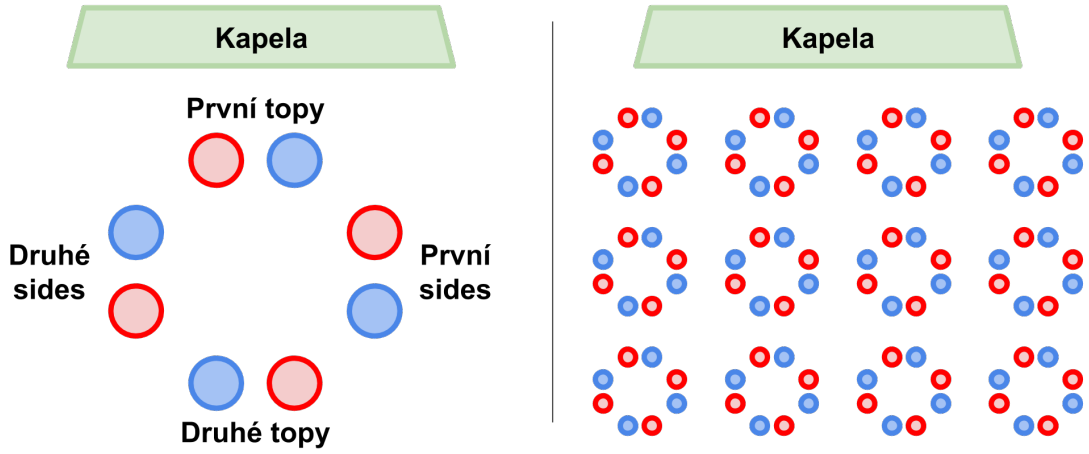
Sean nós Jde o starý styl sólového irské tance, pro který je typické speciální podupávání velmi podobné stepu. Při podupávání se nohy nezvedají příliš vysoko nad zem příliš vysoko nad zem. V tomto druhu tance je dovoleno tanečníkům volněji pohybovat rukama. Každý tanečník má vlastní osobitý styl a běžně improvizuje po celou dobu tance. Tančí se často na velmi malém prostoru, například vysazených dveřích hospody, či na barelu, čímž nabývají u obecenstva na přitažlivosti. S postupem času se ustálilo několik obecně známých kroků, které jsou nesou jméno podle místa svého původu (například Clare, Connemara, Roscommon). Zkušení tanečníci setových tanců rádi podle chuti přidávají tyto kroky na různá místa tance.

Setové tance Jde o společenské tance vycházející z francouzských čtverylek, které byly časem přizpůsobeny irské tradiční hudbě a místnímu tanečnímu stylu. Tanečníci vždy nejprve utvoří páry pána a dámy. Následně pak z párů zformují útvar podobný čtverci o čtyřech párech - tento útvar se nazývá "set"a je tedy složen přesně z osmi lidí. Setové tance jsou často časově docela dlouhé, proto se rozdělují na více částí do takzvaných figur. Figura představuje

posloupnost opakujících tanečních prvků. Drtivou většinu dílčích prvků tanečníci dobře znají a umí je zatančit. Kroky pro pána a dámu jsou vždy téměř stejné, pouze jejich provedení je zrcadlově obrácené.

3.1.3 Struktura irského setového tance

Formace Základem tohoto druhu tance je formace tanečníků nazvaná set. Set je uskupení osmi nebo čtyř tanečníků. Tanečníci tvoří páry, kde pán má dámu po pravé ruce. Rozlišují se dva druhy formace setu. První je "full set", který představuje osm tanečníků tvořících čtyři páry. Druhým je "half set", který představuje čtyři tanečníky tvořících dva páry. Aby hlasatel dovedl napovědět konkrétnímu páru v setu, označují se jednotlivé páry způsobem vyznačeným na obrázku níže. Páry se rozdělí do dvou skupin na takzvané topy a sides. První topový pár je vždy ten, který je narozdíl od ostatních párů blíže k hudbě. Druhý topový pár je naproti prvním topům. Zbylé páry se nazývají sides. Nalevo od prvního topového páru jsou první sides. Opět naproti jsou pak zase druhé sides. U některých tanců je označení sides prohozené, ovšem na tuto skutečnost hlasatelé vždy předem upozorňují.



Obrázek 3.3: Vlevo je znázorněna formace tanečníku v setu. Vpravo je ukázka jak je obvykle umístěno více setů tanečníků na tancovačce. První topy je pár v setu, který je vždy blíže ke kapele.

Průběh tance Irský setový tanec se skládá vždy z několika, obvykle ze čtyř nebo pěti, figur. Vyjímečně se lze setkat dokonce až s devíti figurami (například u Roscommon Lancers). Každá figura má předepsaný požadovaný rytmus (reel, jig, polka a další) a počet taktů, na který je vhodná. Figura představuje posloupnost opakujících tanečních prvků. Drtivou většinu dílčích prvků tanečníci dobře znají a umí je zatančit. Často se figury také nazývají jménem, které vyzdvihuje vyznamný prvek uvnitř figury.

Během tancovačky se obvykle zahlásí, který tanec se bude tančit. Tanečníci utvoří sety a jsou připraveni k tanci. Po dotančení figury je krátká přestávka, během které si tanečníci rychle připomenou průběh následující figury. Po 1-2 minutách se následně se pokračuje další

figurou. U populárních a známých setů se často tančí bez přestávky rovnou další figura - na to musí mít kapela připravený dostatečně dlouhý repertoár skladeb. Například Ballyvourney jig set se nejčastěji tančí bez přestávek.

Příklad: setový tanec Connemara Reel Tento oblíbený tanec obsahuje celkem čtyři figury [15], které se tančí postupně za sebou. Je možné je doplnit krátkými přestávkami. Většina figur je v reelovém rytmu a tančí se poměrně svižně rychle. Poslední figura je rytmem polka, je proti ostatním figurám kratší a netančí se tolik rychle - jde o poslední figuru, kdy se většinou vyměnují dámy v jednotlivých párech a všichni tedy mají možnost se v klidu naposled poznat a rozloučit.

- 1. figura) název: Ladies chain, rytmus: reel, počet taktů: 160
- 2. figura) název: Back to back, rytmus: reel, počet taktů: 192
- 3. figura) název: Swing, rytmus: reel, počet taktů: 184
- 4. figura) název: Maggie in the Woods, rytmus: polka, počet taktů: 96

1	*Everyone* **Lead around & back**	16
2	**Swing**	8
3	*Top Couples* **Back to Back** Top gent & opposite lady do-se-do passing right shoulders then left without turning around.	8
4	**Swing** Top gent & opposite lady swing in the center	8
5	**Adv & retire twice** Both top couples.	8
6	**House**	8
7	**Swing**	8
8	*1st Sides* **Dance 3 to 7**	40
9	*2nd Tops* **Dance 3 to 7**	40
10	*2nd Sides* **Dance 3 to 7**: All couples swing at the end with the 2nd side couples	40

Obrázek 3.4: Ukázka první figury tance Connemara

Příklad první figury tance Connemara lze jsem znázornil na obrázku níže. Můžeme si všimnout, že tanec začíná krokem nazvaným "Lead around", který na 16 taktů tančí všechny páry (everyone). Následuje taneční krok "Swing"na 8 taktů, který opět tančí ještě všechny páry. Pak začíná část tance, kde tančí pouze topové páry (top couples, což jsou ty páry,

které jsou čelem nebo zády k hudbě). Následuje popis, co přesně tančí a kolik taktů mají krokům věnovat. Během této části sides páry stojí na místě. Jakmile topové páry dokončí svou část, jsou na řadě páry sides a tančí stejné kroky, které před chvíli tančily topové páry. Základem každého zápisu tance je tedy jásná posloupnost kroků se zmínkou jak dlouho krok tančit a který pár jej má tančit. Pokud je v tanci nějaký speciální neznámý krok, je návod doplněn o podrobné vysvětlení pohybu. Tyto návody v redukované formě taháku mají někteří tanečníci v kapse, případně během tancovaček jsou tyto stručné nápovědy rovněž promítány projektorem na viditelné místo.

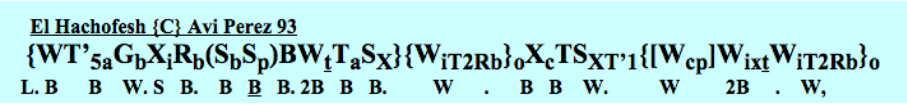
Tance se běžně pojmenovávají podle místa vzniku či původu. Můžeme se tedy setkat s názvy jako Connemara Reel, West Kerry, Ballyvourney nebo Clare Lancers. Pokročilí tanečníci dovedou do tance pocházejícího z určitého kraje doplnit také battering (podupávání na způsob stepu) pocházící z téhož kraje.

3.1.4 Taneční notace

Taneční notace obecně reprezentuje pohyb člověka v tanci. Od dob středověku se vyvinulo mnoho metod a způsobů, které využívají pro popis pohybu grafické symboly, znázorňují trasu tanečníka, dobu trvání kroku, jeho setrvání v určitém momentu. Jiné využívají pro popis prostá slova. Pokud k tanci existuje popis je mnohem snadnější jej šířit. V současné době se čím dál častěji rovněž pořizuje video záznam tance, který je pro výuku ideální. Ovšem pro rychlé připomenutí průběhu tance je textový zápis stále preferovanější.

3.1.4.1 Shorthand Dance Notation

Notace jsou často ušity na míru určitému druhu tance, ve kterém je notace nejvystižnější a nejpopisnější. Velice zajimavě a důmyslně navrženou notací je notace Shorthand Dance Notation, která byla vytvořena speciálně pro izraelské tance. [17] V zápisu používá pismenné nebo číselné značky pro určité kroky izraelského lidového tance. Notace má definované množství výrazů, které dovedou popsat detailně jak pohyb těla tak pohyb po prostoru a v čase hudby. Je tak možné zaznamenat například podrobnosti o přenášení váhy tanečníka, zaznamenat výchozí "nastavení"tanečníka, následný pohyb v tanci, určit části, které se kolikrát opakují. Dokonce je možné zmínit a popsat, kdy má tanečník během tance začít zpívat. Pomocí různých dalších znaků lze vyjádřit na kolik dob je příslušný taneční krok. Výhodou je krátkost zápisu a množství možností, které lze zaznamenat a potenciál pro počítačové zpracování. Nevýhodou je obtížnost pochopení a zapamatování syntaxe pro laiky. Přesto existuje množství tanců takto zaznamenaných, zejména izraelských tradičních tanců.



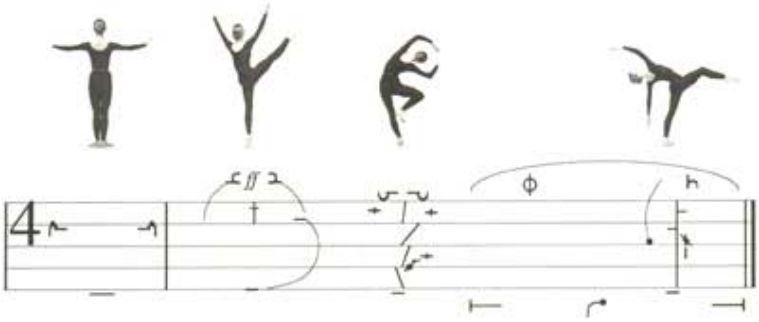
Obrázek 3.5: Shorthand notace pro izraelské tance

3.1.4.2 Benesh Movement Notation

Tuto zajímavou notaci vyvinul britský matematik Rudolf Benesh (jeho otec byl Čech a matka anglo-italského původu) pro svou manželku, která se profesionálně věnovala baletu a tančila v londýnském divadle Sadler's West.

Notace je používána pro zaznamenávání pohybu lidského těla. Dovede poměrně přesně zaznamenat a vystihnout tvar lidského těla v určitých fázích pohybu. Jednotlivé fáze pohybu těla se pak zapisují v posloupnosti za sebou do šablony, které je velmi podobná linkám v notovému záznamu. Linky v záznamu reprezentují části lidského těla (například hlava, ramena, pas, kolena) a pak úplně spodní linka představuje podlahu.

Uplatnění notace nalezla v divadle a baletu při vytváření a popisu choreografií. Dodnes se používá britské Royal Academy of Dance. Podivuhodné je, že se notace stala užitečnou, také v jiných oborech, například v psychoterapii, kde umožňuje lékařům popsat symptomy, které tělo vykazuje v různých situacích či při nemoci. Zkušenější čtenář notace si dokáže velmi rychle představit pohyb ve skutečnosti. Nevýhodou je zaměření spíše na popis tvaru těla v určité části choreografie spíše než na vývoj a průběh tance.



Obrázek 3.6: Benesh notace pro balet

3.1.4.3 Návrh vlastní notace

Existující taneční notace jsou velmi často zaměřené na popis tance či pohybu pro konkrétní druh tance. Je to pochopitelné, protože tak poskytují tvůrci maximální efektivitu při zápisu a při čtení. Využít či převzít tyto notace pro problém setových tanců by bylo příliš komplikované a dostupné možnosti jednotlivých notací by nebyly naplno využity. Obává se, že charakter syntaxe těchto notací by, že by spíše působil značnou zátěž při vyjadřování irských setových tanců. Například není potřeba detailně specifikovat tvar těla při pohybu. Přesto mi poznání těchto notací otevřelo oči a připomnělo některé prvky tance, které bych měl zaznamenat.

Pro irský setový tanec je při popisu prioritou zaznamenat průběh tance v párech. Tedy ujasnit, které páry, který pár či který jednotlivec právě tančí. A kolik taktů provedení kroku vyžaduje. Pro účely lokalizace by bylo dobré zápis připravit tak, aby případnou budoucí lokalizaci usnadnil. Pro potřeby aplikace implementované v rámci této diplomové práce jsem se rozhodl navrhnout vlastní notaci založenou především na potřebách setového tance a nutností

aby byl zápis čitelný počítačem. Měl by také umožnit definovat pohyby znovupoužitelně, aby bylo možné jejich definice využít ke složení složitějších pohybů.

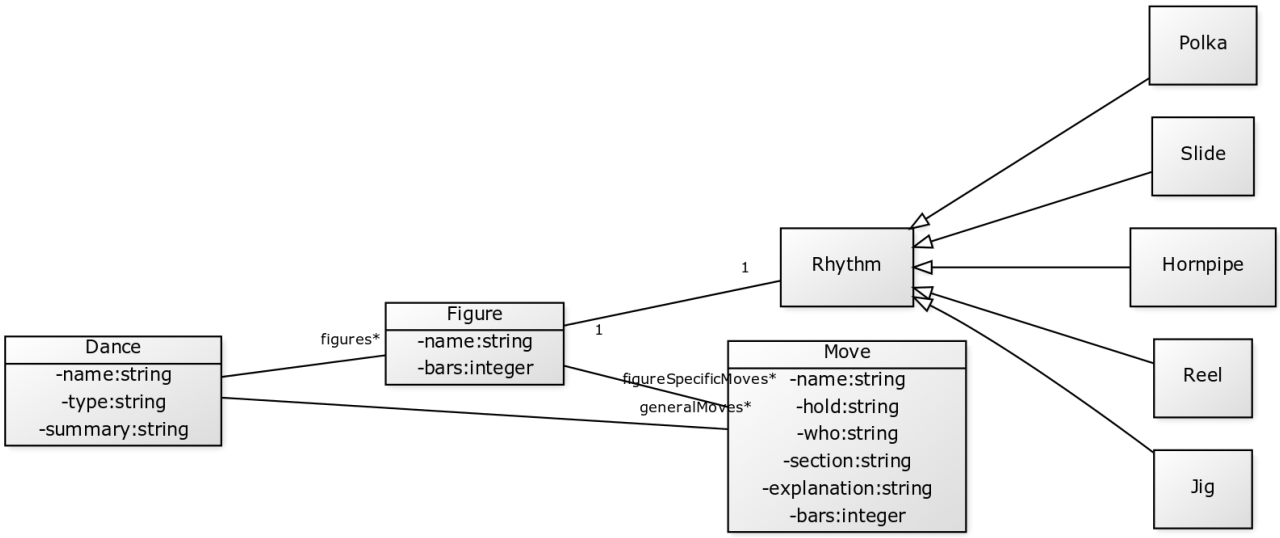
Zápis tance jsem se rozhodl reprezentovat pomocí JavaScript Object Notation (JSON). Tento formát byl v posledních letech velmi rozšířen, zejména se zásadami REST návrhu webových aplikací. Počtem použít se již téměř vyrovnává zápisu pomocí XML. Formát je dostatečně pružný, aby umožnil vyjádřit jak pole, tak hashmapu (slovník) a také běžné primitivní typy. Formálně jej nedefinuje schema jako u XML notace, proto je nutné podobu schematu důkladně zdokumentovat. Většina platforem je připravena tento formát zpracovat a disponují knihovnami, které usnadní práci s JSON formátem. Na cílové platformě (iOS) je připravený nativní parser, který převede JSON zápis do objektů jazyka Swift/Objective-C, se kterými se bude pak v aplikaci lépe pracovat.

Základní stavební jednotka tance Základní kamenem notace je objekt "move", který definuje krok pod určitým jménem. Zaznamenává ke kroku počet taktů nezbytných pro provedení prvku, je možné doplnit taneční držení a podrobnější vysvětlení pohybu či sdělit jiné důležité detaily prvku. Jméno kroku je považováno za unikátní identifikátor napříč celým zápisem, proto je možné odkazovat na krok přímo jeho jménem. Pro účely případného budoucího překladu je v plánu do tohoto objektu přidat hashmapu (slovník), který bude držet pro jednotlivé lokalizace překlady kroku do příslušného jazyka.

Přetěžování definice kroku Díky unikátnímu pojmenování kroku je možné zavést do notace vlastnost přetěžování, kterou bude možné využít k znovupoužitelnosti definic dílčích kroků tance. Hlouběji ve struktuře zápisu tance bude možné se odkázat pomocí jména na, dříve či výše ve struktuře, definovaný krok. Tento odkazovaný krok pak bude možné novým zápisem přetížit a definovat tak specifičtější popis kroku, který bude platný jen v kontextu (úrovni struktury zápisu) ve kterém krok přetěžuji. Jde o podobný princip jakým jsou definovány proměnné na příklad v kódu jazyka Java. Například pokud mám krok nazvaný "Swing"v obecné definici, pak redefinice ve struktuře tance bude platit pouze ve struktuře daného tance a nebude mít vedlejší efekt v jiných tancích. Dále pokud krok redefinuji ještě hlouběji ve figuře bude mít platnost pouze ve figuře. Tuto logiku musí implementovat parser v cílové aplikaci. Formát samotný toto nevynucuje, pouze doporučuje. Cílem této znovupoužitelnosti kroku je předejít chybám při zápisu opakovaných kroků. Po přetížení dřívější definice jsou u kroku k dispozici všechny dříve definované informace, které nebyly přetížením upraveny. Hlouběji ve struktuře je tedy možné psát pouze název kroku a všechny dříve definované informace se na toto místo automaticky kopírují.

Vztahy mezi klíčovými entitami V doméně irských setových tanců jsem nejprve identifikoval jednotlivé entity, ze kterých se tanec skládá. Jde o entity reprezentující krok (Move), tanec (Dance), figuru (Figure) a rytmus (Rhythm). Následující diagram popisuje vztahy mezi entitami. Klíčovým principem je, že krok Move je možné přetěžovat v hloubější struktuře stromu.

- Move ... definice dílčího kroku
  - name ... název kroku



Obrázek 3.7: Popis vztahů mezi entititami reprezentovanými v zápisu irských setových tanců

  - bars ... kolik taktů krok vyžaduje
  - hold ... popis držení rukou
  - section ... označuje skupinu souvisejících kroků, pro snadnější orientaci u složitých figur
  - who ... který tanečník či pár má tento krok tančit
  - explanation ... podrobnější vysvětlení kroku

- Dance ... definice určitého zaznamenaného tance
  - name ... název tance
  - type ... typ tance (set dance, ceilí)
  - summary ... popis tance pro případ potřeby
  - moves ... definice kroků specifických pro tanec
  - figures ... seznam figur

- Figure ... definice figury
  - name ... název kroku
  - rhythm ... rytmus figury
  - bars ... na kolik taktů se figura tančí
  - moves ... definice kroků specifických pro figuru
  - flow ... posloupnost jednotlivých kroků

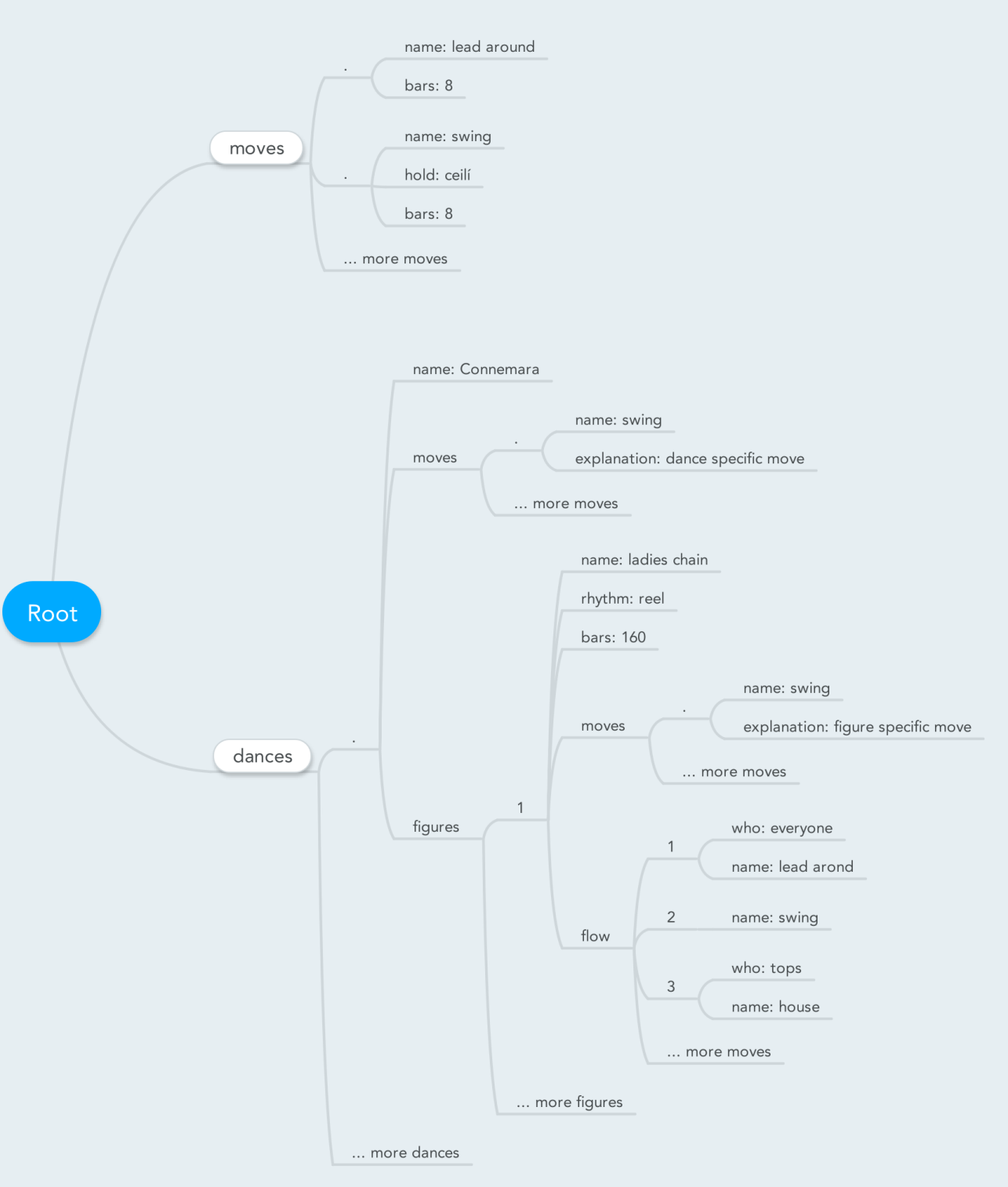
Stavba zápisu Zápis si lze představit jako stromovou strukturu, kde z kořene budou vycházet dvě větve: jedna popisující obecné definice kroků, druhá zahrnující pod sebou všechny definice tanců. Definice kroků budou popisovat základní náležitosti kroku jako je název kroku, držení rukou, počet taktů pro provedení. Uzel s definicí tanců se bude dále hlouběji větvit k definicím jednotlivých zapsaných tanců. Každý tanec bude obsahovat jednak popis sebe sama a bude se větvit na kroky specifické pro tanec a na figury. Uzel pro každou figuru tance pak bude disponovat opět popisem sebe sama (počet nezbytných taktů a rytmu v hudbě) a bude se větvit dál k popisům kroků specifickým pro aktuální figuru a obsahovat především důležitý uzel (nazvaný flow), kde bude popsána posloupnost tanečních kroků pro aktuální figuru, jediný tento uzel bude uspořádaný (půjde tedy o pole, nikoli o množinu). Uzel flow má ještě jednu důležitou vlastnost pro kroky, které budou zapsány o úroveň pod ním - kroky postupně jak plynou za sebou pokud není specifikováno na tomto místě jinak budou brát informaci o položce "who", tedy kdo tančí z kroku který byl definován v poli flow na dřívějším místě. Tato vlastnost umožní při zápisu přehledně zazamenat, že například od prvního kroku dál začínají tančit všechny páry, a třeba o chvíli později (po uplynutí třeba 24 taktů) budou od této chvíle zase tančit jen například topové páry. S touto vlastností se můžeme setkat ve většině zápisů irských setových tanců a přispívá k lepší popisnosti, proto jsem se rozhodl ji zachovat.

Příklad konkrétního zápisu Takto navržený formát je snadno čitelný pro člověka. Rovněž je pro člověka snadné zápsat tanec nový, protože jména struktur vyjadřují přesně a jasně to co znamenají. Snažil jsem se o maximálně srozumetilné a nematoucí pojmenování struktury.

3.1.5 Shrnutí

Irský setový tanec je velmi specifickou odrudou tance a vyžaduje irskou hudbu v určitém rytmu a o určitém počtu taktů. Při výuce i na tancovačkách má důležitou roli, takzvaný hlasatel, který dovede včas a stručně napovídat tanečníkům jak blížící se kroky, tak i který pár v setu je má tančit. Pro tanec se využívá irská hudba o několika specifických rytmech (reel, jig, polka, hornpipe, slide...), které mají vliv na strukturu (těžkých) dob v taktu. Běžně používaná taneční hudba má však značnou výhodu v tom, že muzikanti ji nahrávali s ohledem na důsledné přesné dodržování tempa a je v ní rovněž výrazná basová linka. Tyto faktory se bude snažit využít algoritmus pro detekci hudebních dob.

Struktura tance je podobná stromu. Tanec se skládá z několika uspořádaných figur. Jednotlivé figury pod sebou združují uspořádanou posloupnost tanečních kroků. Stejné kroky se v průběhu tance často opakují, pouze je vždy odlišné pořadí a případně který pár krok právě tančí. Na tancovačkách i na lekcích se po dotančení každé figury obvykle koná menší pauza pro připomenutí kroků v další figuře a následně se figura tančí na hudbu. Pro účely diplomové práce je nutné tance zaznamenat v elektronické podobě. Pro tyto účely jsem vyvořil speciální notaci ušitou na míru potřebám irských setových tanců. Notace má přehlednou stromovou strukturu a srozumitelné jasné pojmenování atributů entit struktury. Je snadno čitelná člověkem i strojem. Povedlo se mi do notace zakomponovat také vlastnost přetěžování, která dovoluje znovupoužitelně definovat jednotlivé taneční kroky a pomáhá tak čitelnosti i



Obrázek 3.8: Stromová reprezentace notace pro zápis irských setových tanců

```
{
   "moves":[
      {
         "name":"Advance and retire twice",
         "hold":"cross hands",
         "explanation":"basic description"
      },
      {
         "name":"Swing",
         "hold":"ceili",
         "bars":8
      }
   ],
   "dances":[
      {
         "name":"Connemara Reel Set",
         "type":"set dance",
         "summary":"1st sides are right of 1st tops",
         "moves":[
            {
               "name":"Advance and retire twice",
               "explanation":"dance specific description..."
            }
         ],
         "figures":[
            {
               "name":"Ladies Chain",
               "rhythm":"reel",
               "bars":160,
               "moves":[
                  {
                     "name":"Advance and retire twice",
                     "explanation":"figure specific description..."
                  }
               ],
               "flow":[
                  {
                     "section":"intro",
                     "who":"Everybody",
                     "name":"Lead around and back",
                     "bars":16
                  },
                  {
                     "name":"Swing",
                     "bars":8
                  },
                  {
                     "section":"A",
                     "who":"Tops",
                     "name":"Advance and retire twice"
                  },
                  {
                     "name":"House"
                  }
               ]
            }
         ]
      }
   ]
}
```


Obrázek 3.9: Přiklad JSON notace pro irské setové tance

stručnosti zápisu. Formát zápisu staví na notaci JSON (běžně používaná při komunikaci s webovými servery).

3.2 Algoritmus pro detekci hudebních dob

Hlavním cílem diplomové práce je umožnit hlasatelům irských setových tanců delegovat hlášení blížících se tanečních kroků na mobilní aplikaci, která dovede včas hlásit taneční kroky do přehráváné vybrané hudby. Aby byla aplikace vůbec schopná kroky hlásit, potřebuje na vstupu dvě nezbytnosti. První je zápis tance, který jsem navrhl v předchozí části a pomocí kterého bude aplikace znát veškeré potřebné informace o tance a může je využít k hlášení blížících se kroků. Druhou nezbytností pro aplikaci je, aby ke každé hudbě znala přesné nebo alespoň přiližné časy taktů. Jedině potom aplikace dovede správně naplánovat hlášení tanečních kroků.

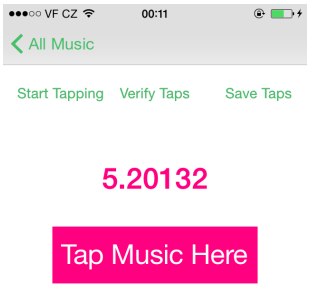
3.2.1 Testovací množina

Před samotným hledáním a tvořením vhodného algoritmu pro detekci hudebních dob bylo nutné sestavit množinu hudebních nahrávek, která bude sloužit k testování správnosti algoritmu. Pro tento účel jsem vybral skladby tradičních irských tanečních kapel (ceilí band) používaných během tréninků a lekcí irských setových tanců. Taneční hudba má však značnou výhodu v tom, že muzikanti ji nahrávali s ohledem na důsledné přesné dodržování tempa a je v ní rovněž výrazná basová linka. Tyto faktory by mohly ulehčit práci detektoru hudebních dob.

Do testovací množiny jsem vybral skladby, tak aby zastupovaly všechny nejpoužívanější rytmy pro setové tance. Tedy reel, jig, polka a slide. Zařadil jsem také několik složitějších skladeb s ne příliš zřetelnou basovou linkou, aby byly výsledky testování objektivnější a robustnější. Celkem tedy 11 skladeb v plné délce. Průměrná délka skladeb je 3 minuty. Jedna skladba je vyjímečně dlouhá 14 minut a vyzkouší také paměťovou šetrnost algoritmu.

- reel ... 5x (každá skladba o průměrné délce 3 minuty, jedna 14 minutová)
- polka ... 3x (každá skladba o průměrné délce 3 minuty)
- jig ... 1x (skladba o průměrné délce 3 minuty)
- slide ... 2x (každá skladba o průměrné délce 2.5 minuty, jedna 6 minutová)

Každou z těchto vybraných skladeb jsem osobně podrobil důkladnému poslechu a ručnímu označení všech hudebních dob. Pro usnadnění tohoto procesu jsem navrhl a naprogramoval jednoduchou jednoobrazovkovou mobilní aplikaci, ve které jsem postupně analyzoval jednotlivé testovací skladby. Aplikace disponovala několika jednoduchými funkcemi pro načtení skladby a přehrání skladby. Na obrazovce aplikace jsem vykreslil tlačítko, které po stisknutí zaznamenalo do databáze čas stisknutí. Během poslechu jsem pak mačkal tlačítko vždy kdy jsem slyšel takt. Tímto procesem jsem vytvořil pro každou skladbu anotaci s časovými značkami pro každou hudební dobu a takt. Měření jsem několikrát opakoval a více naměřených časů u každé skladby zprůměroval takže jsem ke skladbě získal dobře přesné údaje o místech kde se vyskytují hudbní doby.



Obrázek 3.10: Ukázka obrazovky vedlejší aplikace pro anotaci testovací hudby

3.2.2 Metrika pro ověření správnosti nalezených dob

Takto připravenou testovací množinu jsem pak použil pro průběžné ověřování výsledků a chování vyvíjeného algoritmu pro detekci dob. Pro testování jsem na cílové platformě nasal unit test, který spustil algoritmus pro detekování dob s určitým nastavení nad všemi testovacími skladbami. Po skončení unit test vypsal zjištěné odchylky algoritmem nalezených dob od časových značek dob, které jsem anotoval ručně.

Jako metriku jsem nejprve zkoušel zvolit Pearsonův korelační koeficient, který vypadal ze začátku jako vhodný kandidát, který by měřil jak vzdálené jsou od sebe naměřená data a algoritmem zjištěná data. Tato metrika byla dostačující v prvotních fázích vývoje algoritmu, kdy mi pomohla odhalit fatální chyby v algoritmu - například špatné přičtení průběžného času skladby, omyly v převodu z informace o počtu vzorků na časový údaj nalezené doby, případně problémy s posouváním okna zpracovávající analyzovanou skladbu. Tyto chyby často vracely velmi nevypočitatelné časové údaje o hudebních dobách proto porovnání pomocí Pearsonova korelačního koeficientu vracelo nízké hodnoty a motivovalo mě tak k podrobnějším ladění a hledání chyby v aktuálně vyvíjené části algoritmu.

Bohužel v momentě, kdy jsem algoritmus pro detekci hudebních dob dokončený včetně všech fází zpracování začal jsem narážet na chyby poněkud odlišnějšího charakteru. Tedy například kdy hudební doby byly nalezeny ve většině částí skladby, ovšem v některých úsecích skladby byl algoritmus popletený. V určitých melodicky bohatých a složitějších úsecích totiž algoritmus sice správně identifikoval tempo úseku, avšak nedovedl zaměřit správnou fázi hřebenového filtru a posunul tak čas všech nalezených dob přesně o půl doby. Tato chyba je nešikovná, protože může způsobit nesprávné odvození taktů ve skladbě a nakonec pak vést k posunutému napovídání tanečních kroků. Pearsonův korelační koeficient nedokázal tento posun odhalit a vykazoval vždy vysokou hodnutu a vytvářel tak zdání, že algoritmus funguje výborně. Algoritmem nalezená data totiž vykazovala velmi podobně lineární vývoj jako testovací data, který koeficient hodnotí dobře. Rozdíl byl v koeficientu vidět jen jako

přibližně 0.0004 rozdíl proti datům kdy algoritmus doby určil správně, takže jej bylo velmi snadné přehlédnout. Problém jsem odhalil až při vykreslení dob a zvukové vlny na obrazovku aplikace a při také zvukovém testu nalezených dob.

Bylo tedy nutné definovat novou metriku, která by lépe kontrolovala potřebnou kvalitu nalezených dob. Pro další zpracování a cíle diplomové práce, tedy pro napovídání kroků do hudby, jsou nalezené doby užitečné jen pokud jsou detekovány opravdu naprosto přesně - jedině tak lze přesně spočítat takty podle, kterých se následně kroky napovídají. Vytvořil jsem tedy funkci, která přijímá dvě pole. První pole A obsahuje referenční správně ručně anotované doby. Druhé pole B obsahuje algoritmem nalezené doby. Funkce má nastavenou určitou toleranci v hodnotě 0.3 vteřinové odchylky. Postupně prochází pole A a hledá v poli B nejbližší čas, navíc zohledňuje a toleruje nastavenou odchylku. Pokud k době z pole A nenajde žádnou dostatečně blízkou dobu z pole B, pak si funkce poznamená penalizaci za špatně identifikovanou hudební dobu. Funkce vrací procentem vyjádřenou úspěšnost správného odhalení hudebních dob. Tato, postupem sice primitivnější, metoda však daleko přesněji popisuje, kolik dob bylo nalezeno algoritmem správně.

3.2.3 Rešerše a prototypování algoritmu

Analýza hudby byla pro mě osobně novinkou, protože jsem se během studia specializoval zejména na obor softwarového inženýrství. Bohužel jsem dříve neměl příliš štěstí na detailnější seznámení se metodami používanými pro zpracování signálu. Přesto jsem během rešerše poznal a vyzkoušel mnoho nezbytných postupů, které jsem později velmi vděčně využil při analýze hudby. Při prvotních experimentech a zkouškách mi byl velice užitečným nástroj Matlab pomocí, kterého jsem dovedl relativně rychle načrtnout a vyzkoušet fáze potřebných algoritmů. Moc užitečné bylo průběžné vykreslování zpracovávaných dat do diagramu, které jsem využil při poznávání chování zvukové vlny, když se převádí z časové do frekvenční oblasti a naopak, dále při konvolucích s filtry pro zvýraznění změn v hudbě a pro sledování odezvy comb filteru během hledání tempa a fáze hudebních dob.

Existují dva hlavní přístupy k detekci dob v hudbě. První statistický přístup funguje na principu postupného hledání výrazných výkyvů energie signálu v lokáním právě analyzovaném okolí (onset detekce). Druhý přístup je založen na porovnávání šablony periodického signálu (pravidelně rozložené špičky v signálu podobné hřebenu) s hudbou a hledá takovou šablonu, která ve spojení s hudbou vrací nejvyšší energii (bank of resonators).

Skladbu je dobré analyzovat postupně, protože může mít některé části tišší, jiné naopak hlasitější. Často například začne hrát jen jeden nástroj a další (hlasitější) se přidávají později. Případně může kolísat tempo dob. Tyto odlišnosti by mohly algoritmy vychýlit pokud by se zpracovala celá skladba najednou. Rovněž je paměťově šetrnější zpracovávat takto veliký objem nekomprimovaných dat po menších úsecích, které nebudou ohrožovat stabilitu aplikace. Zejména ve značně restriktivním a přísném prostředí mobilních aplikací je dobrá péče o paměťovu náročnost velmi důležitá a cenná.

3.2.3.1 Naivní onset detekce: Hledání výkývů energie v lokálním okolí

První metodou, na kterou jsem během rešerše narazil a kterou jsem zkoušel použít pro detekci byla onset detekce. Jde o relativně jednoduchý princip fugnování, který postupně

hledá nadprůměrně výrazné výkyvy energie hudby v lokálním okolí. Algoritmus zvukový signál postupně rozdělí a nejmenší možné rytmické jednotky [19]. Základem je tedy rozkouskování skladby do několik drobných úseků, které se nazývají basket (koš). Pro každý koš algoritmus spočítá energii signálu a porovná zda je výrazně větší než průměrná, podobně spočtená, energie v okolích koších kolem právě zpracovávaného koše. Pokud je energie v aktuálně analyzovaném koši výrazně vyšší než v jeho okolí, považuje se tento koš (bod) za dobu a zaznamená se čas doby.

Prvním krokem je volba velikosti koše, který představuje nejmenší možnou velikost rytmické jednotky. S ohledem na vzorkovací frekvenci skladby můžou být jiné velikosti koše vhodnější, avšak já osobně jsem během prototypování zpracovával nahrávky vzorkované pomocí nejběžnější 44100 Hz vzorkovací frekvence. Převzal jsem zdrojem doporučenou velikost koše [18], tedy velikost v hodnotě 1024 vzorků. Hodnotu koše e lze spočítat vzorcem 3.1, kde proměnná a je pole vzorků, i_0 je index prvního vzorku právě zpracovávaného koše v poli všech vzorků signálu, k je index identifikující vzorky, které se sčítají do hodnoty koše. Umocnění ve vzorci slouží k usměrnění signálu kladným směrem.

$$e = \sum_{k=i_0}^{i_0+1024} a[k]^2 \tag{3.1}$$

Druhým krokem je volba velikosti lokálního okolí, zde jsem zvolil velikost na hodnotu 44. Hodnotu průměrné energie v lokálním okolí značeného $< E >$ lze spočítat pomocí vzorce 3.2, kde B představuje pole, kam se ukládají energie jednotlivých košů (spočtené vzorce 3.1).

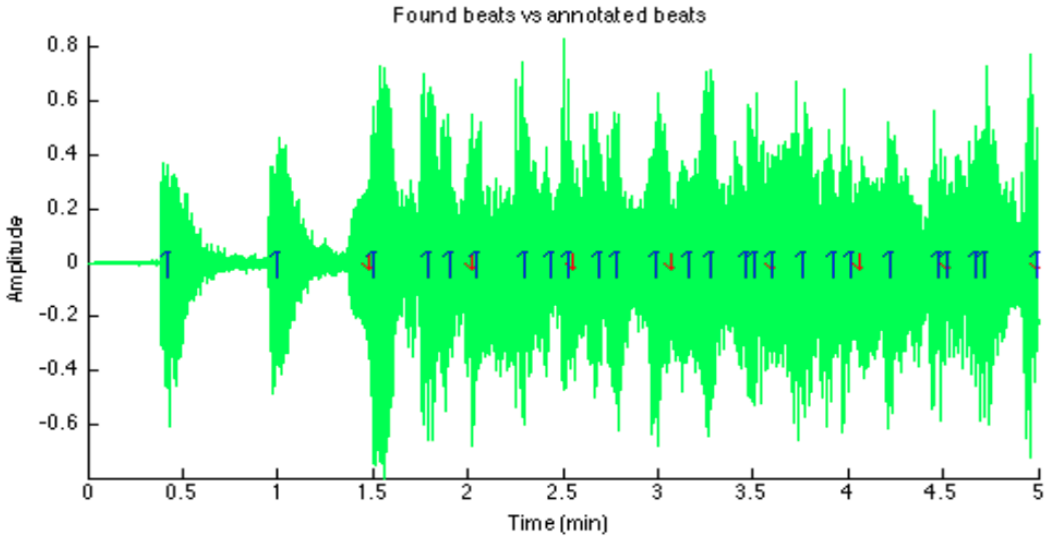
$$< E >= \frac{1}{44} * \sum_{i=0}^{44} B[i]^2 \tag{3.2}$$

Třetím krokem je porovnání energie v aktuálně zpracovávaném koši značené proměnnou e vůči proměnné $< E >$ představující lokální okolí kolem aktuálního koše. Konkrétně se porovnává e vůči $C* < E >$, kde C je konstanta určující citlivost algoritmu na výkyvy energie v signálu hudby. Pokud je hodnota e vyšší než $C* < E >$ pak byla nalezena hudební doba.

Obrázek 3.11 ukazuje, kde byly ve zkušební skladbě nalezeny algoritmem hudební doby (modré značky) a kde jsou skutečné žádané hudební doby (červené značky). Lze pozorovat, že metoda postihne všechny výrazné výkyvy v hudbě a zahrne tedy i potřebné hudební doby, avšak nalezne také množství dalších výrazných míst v melodii.

Pro algoritmus existuje několik možných dalších vylepšení:

- Upravit konstantu C v závislosti na proměnlivosti signálu v okolí. V každé iteraci je potřeba spočítat proměnlivost dle vzorce uvedeného v článku. Toto vylepšení může pomoci skladbám, které obsahují mnoho podobně hlasitých zvuků.
- Rozdělit zkoumanou část signálu do několika pásem podle frekvencí, které obsahují. Tím bude možné snáze určit beaty zvlášť pro hlubší nástroje (bicí, basa) a dát jim větší



Obrázek 3.11: Ukázka nalezených dob v irské hudbě, modře jsou vyznačeny rozpoznané doby, červeně ručně vybrané správné doby

váhu. Obvykle hudební doby ve vyšších pásmech jen potvrzují ty nalezené v nižších. Počet pásem by neměl být vyšší než cca 32, protože pak bude obtížné detekovat doby ve více různých skladbách.

Zhodnocení metody Algoritmus nachází výrazné místa v hudbě velmi rychle a přesně pro každé pásmo. Nevýhodou, je že kromě skutečných dob nalezne ještě mnoho dalších nedůležitých bodů. Pro účely práce potřebujeme pouze doby v pravidelném tempu. Pravidelné tempo dob lze získat jen u skladeb s velmi výraznými basy (hip hop, rock). Vzhledem k charakteru nalezených dob se domnívám, že bude tato metoda nevhodná pro účely cíle diplomové práce, protože nalezené hudební doby nevykazují pravidelnost hledaných těžkých dob. Onset detekce nalezne spoustu zajímavých bodů, kde energie signálu hudby je skutečně vysoká či nějakým způsobem vyjímečná. Těchto míst je ovšem příliš mnoho a nelze v nich po takovémto zpracování nalézt hledané pravidelné tempo hudby a odvodit přesné hudební doby melodie. Odhalené nadbytečné body budou vadit dalšímu zpracování, kde podle nalezených dob bude aplikace odvozovat časy taktů a na základě taktů napovídat. Napovídání založené na této metodě by prakticky vypadalo tak, že by aplikace hlásila blížící se kroky v podivně nepravidelných intervalech, což by bylo pro tanečníky matoucí a nepřineslo by hlasateli mnoho užitku. Proto jsem algoritmus založený čistě na tomto způsobu detekce zavrhl a hledal jiné způsoby. Bohužel metodu nelze využít ani jako první krok předzpracování hudby před jinými metodami, protože z výstupu algoritmu již není možné získat podrobnější informace o frekvenční oblasti. Metoda onset detekce je vhodná spíše pro jiné účely - myslím, že například pro aplikaci, která by měnila barevné vizualizace podle výrazných změn v hudbě, či pro generátor video alba fotografií, kdy se fotografie mění s ohledem na změny v hudbě. Jak dále ukážu existuje naštěstí ještě další metoda, která je založená na hledání pravidelností v hudbě.

3.2.4 Lepší způsob hledání dob pomocí hřebenového filtru

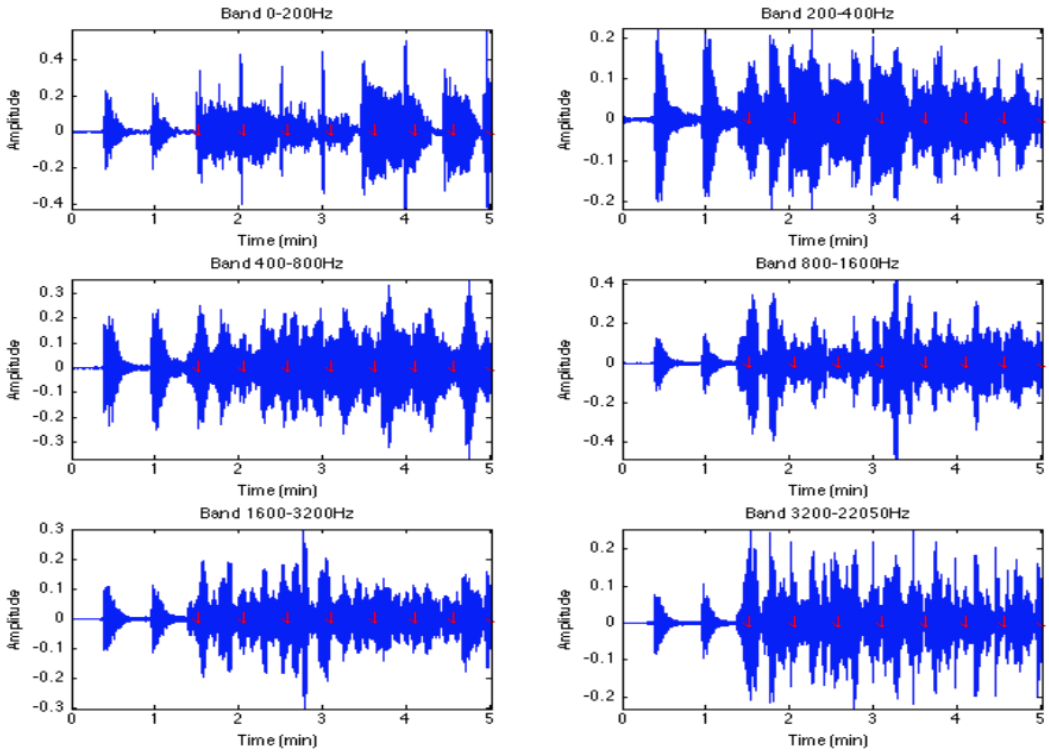
Základem je úvaha, že pro dva signály $x(t)$ a $y(t)$ můžeme pomocí cross-korelační funkce stanovit, jak moc jsou tyto signály podobné. Jedním signálem je analyzovaná hudba. Druhým porovnáváným signálem je vždy jeden z několika periodických signálů, které rezonují v určitém tempu. Algoritmus čerpá zejména z práce Erica D. Scheirera [7].

3.2.4.1 Princip fungování

Signál hudební nahrávky na vstupu bude zpracováván po částech pomocí posuvného okna. Každé okno vrátí určitý počet nalezených hudebních dob, z tohoto počtu vždy zachová 70% dob od začátku okna, zbývající doby ke konci okna budou vypuštěny. Okno se posune o 70% své velikosti dál k novým datům hudebního signálu. Akceptací pouze zlomku nalezených hudebních dob umožňujeme algoritmu doopravit případné anomálie na okraji okna způsobené převody pomocí Fourierovy transformace z časové do frekvenční oblasti a zpět.

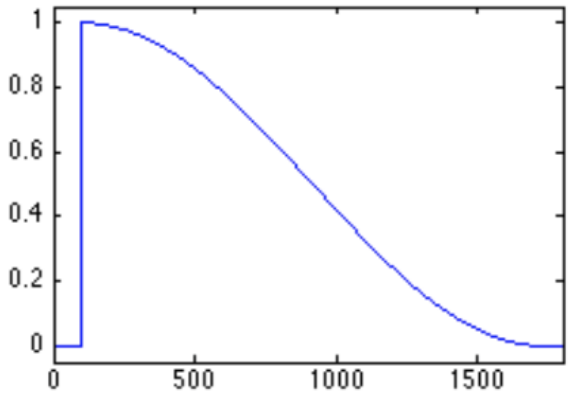
1. krok) Filterbank / Rozdělení do pásem podle frekvencí Jde o výborný způsob, jak můžeme algoritmu nabídnout pomoc a dovolit mu ujistit se o existenci zajímavého vzoru v signálu z více zdrojů. Dalším krokům algoritmu pak mohou brát ohled na specifický vývoj melodie v různých hladinách zvuku. V běžném hudebním signálu tak jak přichází na vstupu jsou po zaznamenání do digitální podoby smíchány dohromady informace o hlubokých a vysokých tónech. Separovat do sebe hluboké a vysoké tóny je možné, tak že nejprve převedeme signál z časové oblasti do frekvenční. Ve frekvenční oblasti vybereme postupně koše náležející dovnitř určitého pásma. Hraniční frekvence pro rozdělení do pásem se v příkladech[7] doporučují volit logaritmickou řadou. Jakmile jsou koše rozděleny do pásem, provede se pro každé pásmo zpětný převod z frekvenční oblasti do časové, čímž získáme zvukovou vlnu obsahující pouze tóny o frekvenci dané hranicí pásma. Během implementace jsem pro uchovávání této struktury vytvořil objekt podobný matici, kde v řádcích jsou uložena jednotlivá pásma a ve sloupcích samotné vzorky příslušných pásem. Tuto strukturu pak předávám dalším krokům algoritmu. Tento krok je nastaven tak, aby signál rozdělil do následucích pásem:

- low pass do 200Hz
- band pass
  - 200 Hz až 400 Hz
  - 400 Hz až 800 Hz
  - 800 Hz až 1600 Hz
  - 1600 Hz až 3200 Hz
  - 3200 Hz až 4096 Hz
- high pass od 4096 Hz



Obrázek 3.12: 1. krok: Ukázka pásem ve časové oblasti, vývoj melodie je nyní v každém pásmu zřetelnější než byl u původních vzorků kde byly zaznamenány dohromady všechny frekvence.

2. krok) Envelope / Obálka signálu zajistí, aby každé pásmo mělo jemnější a hladší signál bez příliš extrémních výkyvů. Každé pásmo je nejprve nutné dvoucestně usměrnit (full-wave rectification). Pro vytvoření obálky provedeme konvoluci s polovičním Hanning oknem, které sice zjemňuje pomalu měnící se signál signál, ale zároveň zachovává náhlý růst. Náhlý růst reprezentuje určitou zajímavou část melodie a téměř vždy je mezi těmito body vždy i bod který představuje hledané pravidelné hudební doby. Tím, že poloviční Hanningovo okno zjemní plynulé pro nás nezajímavé části signálu, ale zachová místa s náhlým růstem můžeme tato místa v dalších krocích využít a detekovat.



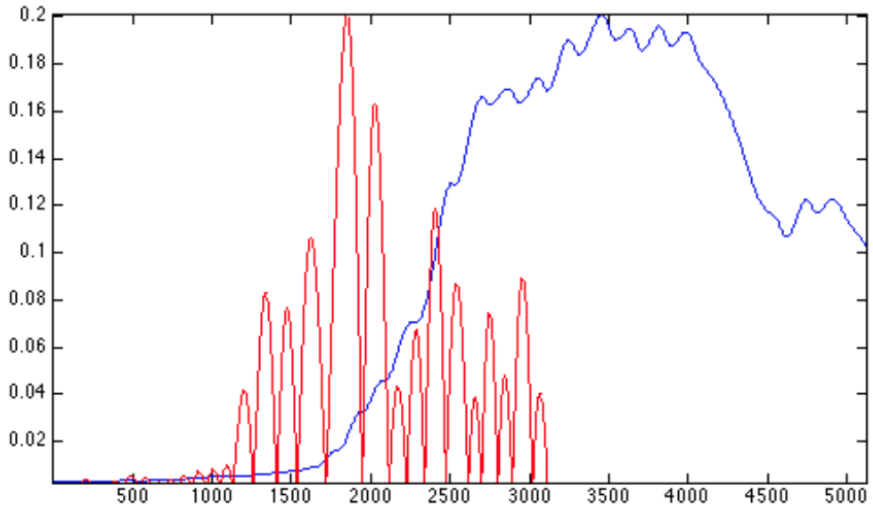
Obrázek 3.13: 2. krok: Poloviční Hanning okno

3. krok) Zvýraznění náhlých změn signálu Jak jsem upozornil dříve, místa s náhlým růstem jsou pro nás zajímavá, protože se mezi nimi často nachází také body představující pravidelné hudební doby. V tomto kroku se algoritmus pokusí tato místa zvýraznit, aby je bylo snažší identifikovat v dalších krocích. Pro zvýraznění využije derivaci prvního řádu, kterou aplikuje na vzorky každého pásma. Konkrétně aplikuje aproximovanou podobu derivace prvního řádu (viz 4.1), kde x je pole se vzorky a i je proměnná označující aktuálně zpracovávaný vzorek.

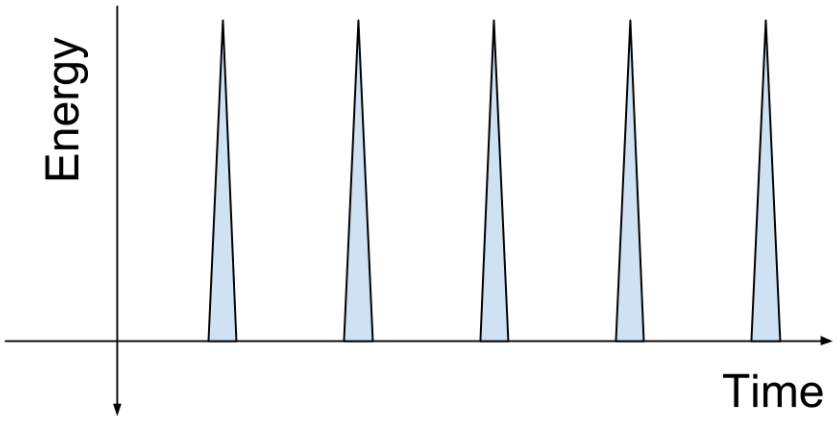
$$x[i-1] - x[i] \tag{3.3}$$

4. krok) Zjištění tempa je jedním z podstatných kroků algoritmu. Tempo lze popsat jako frekvenci výskytu periodického pulzu v signálu. Jakmile zjistíme v signálu dominatní tempo máme k dispozici první důležitou informaci, která nám pomůže identifikovat přesná místa hudebních dob.

Pro hledání tempa využíváme pomocný synteticky vytvořený signál, který má podobu hřebenu - proto se této metodě říká hřebenový (comb) filtr. Jednotlivé vrcholky hřeben jsou od sebe vzdáleny v určitém tempu. Hledání probíhá od určité spodní hranice tempa až po horní hranici tempa.



Obrázek 3.14: 3. krok: Derivace části obálky



Obrázek 3.15: 4. krok: Příklad hřebeného syntetického signálu používáneho k hledání tempa

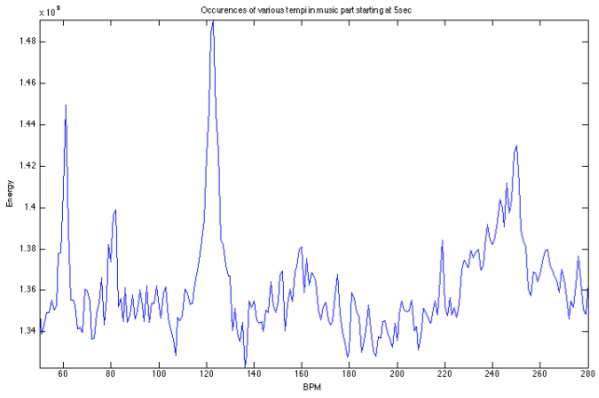
Frekvenci impulzů v hřebenovém filtru i dob v hudbě udává míra BPM (beats per minute, doby za minutu). Počet vzorků Ti mezi každým impulzem lze spočítat vzorcem, kde fs je vzorkovací frekvence záznamu (typicky 44100), BPM je míra dob za minutu.

$$Ti = \frac{60}{BPM} * fs \tag{3.4}$$

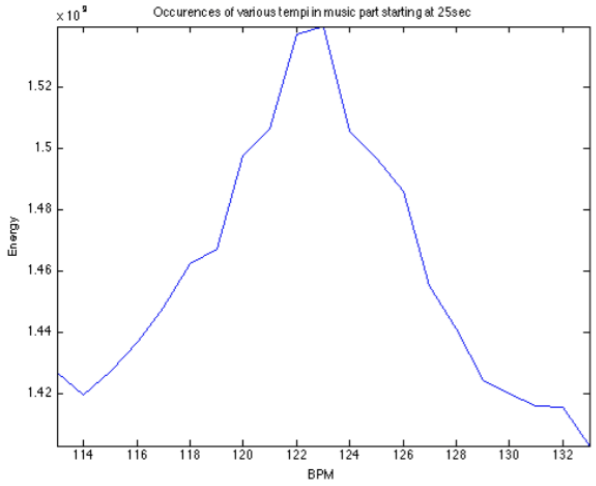
Tempo irské hudby se běžně pohybuje od 60 BPM do 160 BPM. Pro první úseky skladby je nastaven široký rozsah spodní a horní hranice (60 až 220 BPM, beats per minute, dob za minutu). V dalších úsecích skladby je možné hledaný rozsah zužit, aby algoritmus nezmátly některé neobvyklé variace v melodii uprostřed skladby. V každé iteraci hledání je známo hledané tempo, například 73 BPM. Pro toto hledané tempo je vytvořen hřebenový filtr, který má vrcholky od sebe vzdálené přesně v tomto tempu. Následně je hřebenový signál postupně zleva do prava přikládán k vzorkům analyzované části a při každém přiložení se spočítá míra rezonance hřebenového signálu se vzorky hudby. Aby bylo detekování tempa robustnější počítá se míra rezonance ke každému frekvenčnímu pásmu. Jakmile jsou vyzkoušeny všecna hledaná tempa vybere se tempo, které nejlépe rezonovalo s hudebním signálem.

Formálně lze hledání tempa popsat jako hledání nejlepší hodnoty cross-korelační funkce signálem hudby a mezi několika synteticky vytvořenými signály (comb), které rezonují v určitém tempu. Při implementaci provádím dílčí výpočty cross-korelační funkce ve frekvenční oblasti, kde je lze spočítat značně rychleji než v časové oblasti jako součin obou porovnávaných signálů. Hodnota E_y reprezentuje míru rezonance syntetického hřebenového filtru X s frekvenčním pásmem hudby Y. Ve vzorci konstanta N popisuje počet vzorků, k je index právě počítaného vzorku.

$$E_y = \sum_{k=0}^{N} |X[k] * Y]|^2 \tag{3.5}$$

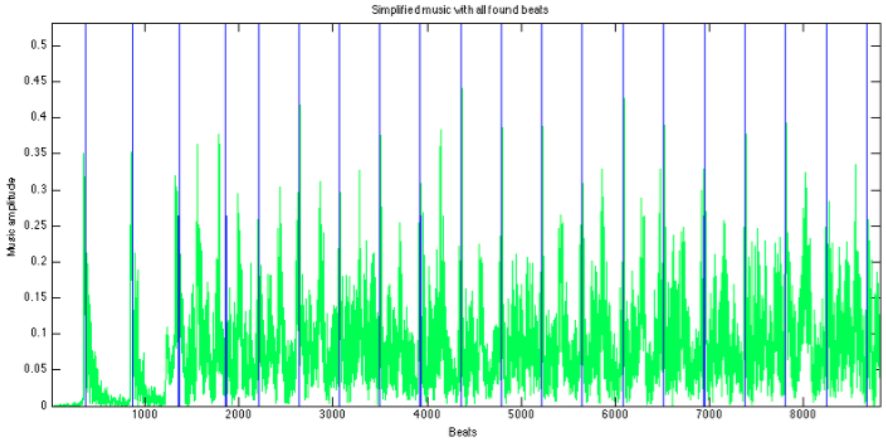


Obrázek 3.16: 4. krok: Výsledky shody comb signálů s hudbou ukazují, že dominantním tempem je cca 120 BPM. Další vysoké vrcholky na 60 a 240 představují násobky dominatního tempa.



Obrázek 3.17: 4. krok: Zužením lze hledání zrychlit a zároveň zachovat pružnost pokrývající drobné odchylky v tempu hudby.

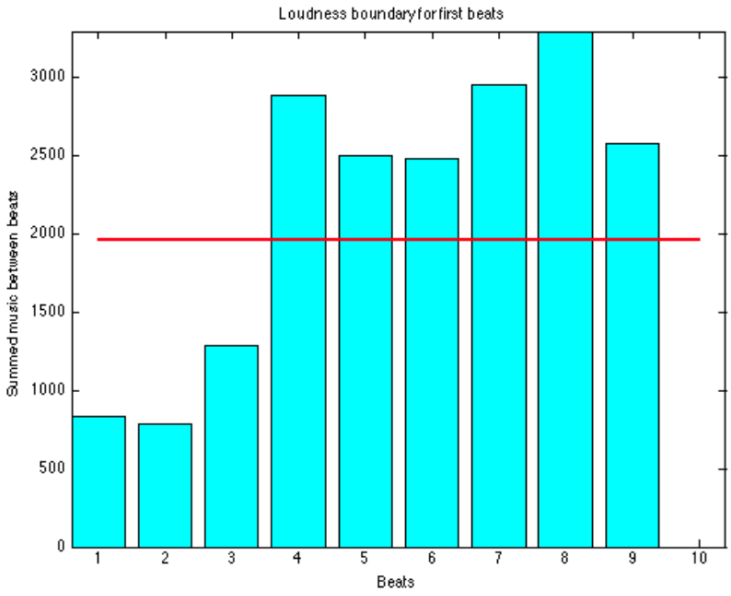
5. krok) Určení fázového posunu Tato fáze je důležitá zejména proto, že nám umožní lokalizovat přesné časy, kde se hudební doby nacházejí. Pro zjištění fáze je nutné opět použít comb signál odpovídající dominatnímu tempu, které jsme nalezli v předchozím kroku. Comb signál se v časové oblasti posouvá po jednotlivých vzorcích intervalu tempa a hledá se nejlepší shoda se signálem hudby. Místo nejlepší shody určuje fázi. Výsledkem jsou místa, kde se mají objevovat doby.



Obrázek 3.18: 5. krok: Posunutí hřebenu na pozice kde vykazuje nejvyšší sumy v konvoluci se signálem

6. krok) Vyfiltrování nezajímavých dob V některých skladbách irské taneční hudby jsou první doby pouze zahajovací, kdy kapela dvěma tóny naznačí tempo a ve skutečnosti

hudba začne hrát později. Případně skladba obsahuje předtaktí. Případné doby označující tato místa nás pro účely napovídání kroků nezajímají. Dokonce by mohly mít negativní vliv (hlášení kroků příliš brzy), proto je důležité je důkladně vyfiltrovat. Společnou vlastností těchto dob je jejich nízká celková hlasitost proti dobám ve zbytku skladby. Proto doby na začátku skladby, které mají podprůměrnou hlasitost jsou vypuštěny.



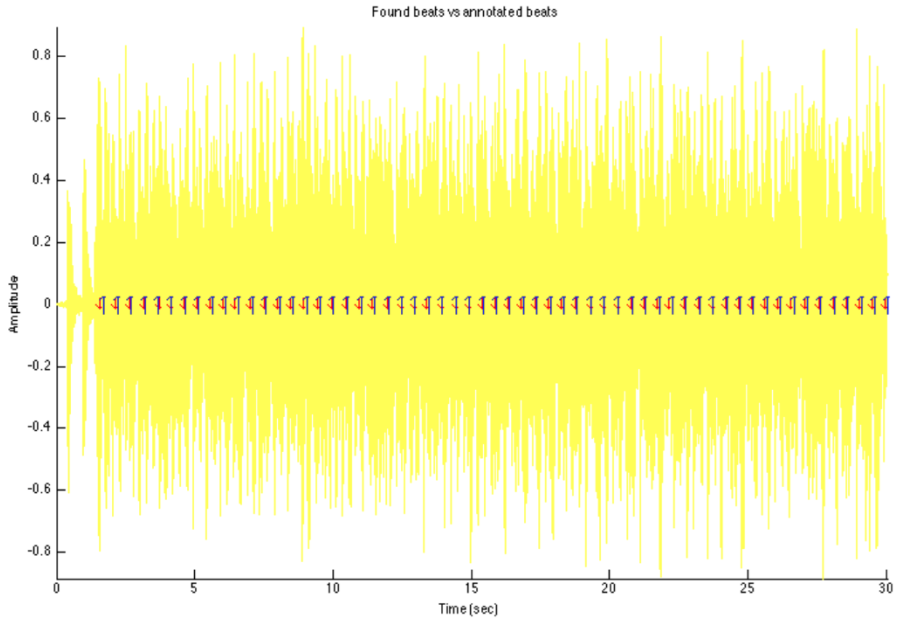
Obrázek 3.19: 6. krok: Odfiltrování nezajímavých dob s podprůměrnou hlasitostí

3.2.5 Testování algoritmu

Výhodou principu fungování algoritmu je, že respektuje tempo dob a nalezne správný počet dob. Ovšem během testování a průběžného ladění algoritmu jsem narážel na zajímavé překážky u některých skladeb, které měly v určitých částech zvláštní důrazy v melodii a mátly tak algoritmus pro detekci správných dob.

Během ladění a testování algoritmu jsem vytvořil pro účel kontroly kvality opravy unit test, který mi automatizoval rutinní práce spuštění aktuálního detekčního algoritmu nad testovacími daty. Unit test pro každou skladbu vždy spočítal aktuální metriku určující míru úspěšně určených hudebních dob. První stabilní verze algoritmu detekovala hudební doby s průměrnou úspěšností 54.3%, což nebylo příliš pozitivní proto jsem hledal možné příčiny chybovosti.

Krátké okno Prvním problémem byla skutečnost, že některé úseky skladeb obsahovaly melodii, ve které algoritmus rozpoznal jiné tempo než běžně průměrně rozpoznával do té doby u ostatních částí skladby. Taneční skladby v drtivé většině drží vždy velmi podobné tempo až do konce. Bylo tedy nutné algoritmus donutit, aby pokud již nějaké tempo rozpozná na začátku skladby, aby toto tempo dodržoval s postupem času, a povolil jen drobné



Obrázek 3.20: Výsledek: Nalezené doby se shodují s ručně anotovanými dobami. Modře nalezené doby, červeně ručně anotované

dochylky v rozsahu 10% od průměrného tempa. Tento problém se mi podařilo vyřešit experimentováním s délkou zpracovávaného okna. Původní verze používala okno o velikosti 5 vteřin. Při zvyšování se začala zlepšovat také míra úspěšnosti detekce. Jako optimální se ukázala velikost okna 30 vteřin, která dosáhla průměrná úspěšnost detekce na 77.8%. Větší okna zbytečně zatěžovala procesor zařízení a nepřinesla větší úspěch.

Posunutá fáze Druhý problémem, který se projevil téměř současně s předchozí nepříjemností, byla skutečnost, že opět některé části skladby měly velmi podobně pravidelnou melodii třeba o půl doby posutou a algoritmus tak nalezel, jako silněji rezonující právě tuto posunutou linku, která ovšem nepředstavovala správné hudební doby. Tento problém se povedlo odstranit pokusy s omezením frekvenčních pásem s vyšším tóny, kdy úspěšnost detekce zlepšila při vypuštění frekvenčních pásem nad 1600 Hz. Dále ke zlepšení pomohla pojistka k hledání umístění fáze comb filteru, která hledání omezila určitou drobnou hranicí nepřesahující polovinu vzdálenosti mezi dobami comb filteru. Tato vylepšení posunuly průměrnou úspěšnost detekce až na 91%.

3.2.6 Shrnutí

Práce na hledání, implementaci a následném ladění algoritmu pro detekci hudebních dob, byla jednou z nejobtížnějších částí mé diplomové práce.

Jsem rád, že jsem ještě před samotnou implementací věnoval dostatečný čas přípravě testovací množiny hudebních nahrávek kapel hrajících irskou tradiční hudbu používanou k tanci. Zejména úsilí věnované anotování správných dob jsem ocenil během ladění algoritmu.

Velikost okna	Průměrná úspěšnost detekce	Časová náročnost detekce
5 vteřin	54.30%	15.4 vteřin
10 vteřin	57.15%	16.7 vteřin
20 vteřin	60.23%	18.2 vteřin
30 vteřin	77.80%	21.7 vteřin
60 vteřin	49.18%	98.6 vteřin

Obrázek 3.21: Experimenty s velikostí okna zpracování

Mohl jsem tak důvěřovat dříve správně nalezeným dobám a soustředit se na hledání chyb v implementaci algoritmu. Hlavně během finální optimalizace bylo moc užitečné experimenty s konfigurací algoritmu ověřovat pomocí unit testu, který mi automatizoval ověřování.

Algoritmus se ukázal být spolehlivým na testovací množině dat a dovede tedy aplikaci dodat potřebně přesné informace o časovém umístění hudebních dob. Aplikace pak dovede podle hudebních dob jednoduše dopočítat časy taktů a naplánovat včasné hlášení blížících se tanečních kroků.

3.3 Rešerše vedlejších problémů

3.3.1 Změna rychlosti hudby při zachování výšky tónů

3.3.1.1 Resampling

Základním způsobem jak změnit rychlost hudby je "resampling". Záznam se prodlouží poměrně k míře zpomalení hudby. Stávající vzorky se pak rovnoměrně rozprostřou po délce nového záznamu. Dále se chybějící vzorky dopočítají pomocí interpolace mezi okolními vzorky. Nevýhodou resamplingu je změna výšky tónů, která je způsobena tím, že původní frekvence tónů byly naměřeny pro původní vzorkovací frekvenci.

3.3.1.2 Phase vocoder

Výhodou použití phase vocoderu [20] je, že při prodlužování nebo případně zkracování délky záznamu se nemění výška tónů. Algoritmus postupuje po blocích signálů, které navíc zjemňuje přes filtrační okno.

Pro každý blok provádí následující kroky:

- Převede blok do frekvenční oblasti
- Úpravou fáze frekvencí změní výšku tónů
- Převede blok zpět do časové oblasti
- Uloží blok do nového signálu metodou OLA (overlap and add), která zajistí jemný přechod mezi bloky a hlavně zajistí zrychlení/zpomalení hudby

3.3.1.3 SOLA

Představuje zkratku pro Synchronous Overlap and Add a je alternativou k phase vocoderu. Funguje na principu rozkouskování signálu do několika drobných částí a následně spojení těchto částí dohromady, s tím, že některé části jsou zopakovány (pro zpomalení), nebo vypuštěny (pro zrychlení).

Cílem je nalézt periody (základní frekvence) vybrané části signálu pomocí algoritmu detekci výšky tónu (provádí se obvykle výběrem maxima v autokorelaci signálu). Princip se nazývá "time-domain harmonic scaling"[22]

Pro zajištění lepší návazností bloků algoritmus vybírá další přidávanou část s ohledem na to jak dobře zapadá za poslední tóny aktuálního bloku. Zjištění podobnosti se provádí cross-korelací mezi koncem aktuálního bloku a několika možnými posuny následujícího bloku.

Nevýhodou algoritmu proti Phase Vocoderu jsou občasné zvukové artefakty (ozvěny) způsobené opakováním/vypouštěním vzorků.

3.3.1.4 Výběr knihovny pro iOS aplikaci

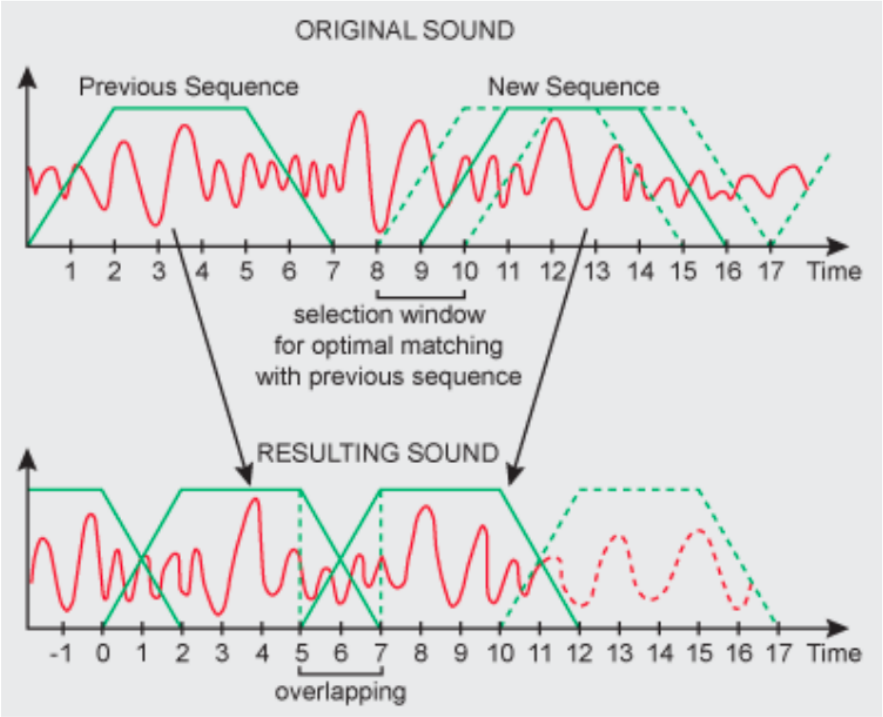
Na základě poznání fungování jednotlivých metod jsem hledal knihovny pro zpracování audio signálu, takové které by zpomalení zvuku prováděly pomocí algoritmů založených na metodě SOLA.

Knihovny pro iOS poskytují většinou buď vlastní

Výběr jsem zužil na dvě komerční knihovny Dirac Mobile a Ruber Band Library, které se zaměřují specificky na změnu tempa hudby s ohledem na výšku tónu. Podporují jak jednorázové zpracování tak zpracování proud signálu například při streamování.

Nakonec jsem zvolil jako ideální však knihovnu Sound Touch [21], která pod open source licencí nabízí srovnatelné funkce, včetně zpracování živě streamovaného signálu. Je psána multiplatformně a obsahuje řadu příkladů pro zprovoznění pod OSX. Zvládá upravovat tempo hudby při zachování výšky tónu, i bez jejího zachování. Využívá modifikovanou verzi SOLA založené na porovnávání kvality jednotlivých možností umístění v novém prodlouženém záznamu.

(viz obrázek 3.17)



Obrázek 3.22: Fungování algoritmu SOLA (synchronous overlap and add

3.3.2 Výběr vhodných tanečních prvků

Jedním z problémů, jehož řešení jsem chtěl prozkoumat je nalezení způsobu jak automaticky vybrat taneční kroky tak aby vhodně souzněly s hudbou.

V této části práce se zaměřím vyjímečně na irské sólové tance, které se narozdíl od setových společenských tanců, tančí samostatně. Má sice proti setovým tancům kratší choreografie, avšak solové taneční kroky vyžadují často mnohem méně taktů (1,2,3,4 i 5 taktů) k provedení než společenské tance. Kratší taneční kroky však příjdou vhod, protože umožní lépe využít melodie hudby. Avšak metodu je možné snadno přenést také do setových tanců, pouze bude generování snadnější protože kroky obvykle nejsou kratší než čtyři takty.

Algoritmus který by dokázal vygenerovat taneční kroky na míru hudbě by byl velmi přínosné například pro účely tvorby choreografií nebo pro trénování pozornosti při volání nových kroků. Kroky tance jsou totiž často poutavější a líbívější (pro diváka i pro tanečníka samotného) právě proto, že změny kroků se dějí s ohledem na vývoj melodie hudby. Pokud se například v hudbě vyskytuje podobná melodie je vhodné zařadit podobné kroky pro části obsahující tuto melodii.

Tanec se skládá z tanečních prvků, z nichž každý lze obvykle provést za jeden až dva takty. Taneční prvky také můžou být různě náročné. Navíc na náročnosti může přidat i určitá specifická posloupnost těchto prvků těsně za sebou. Například:

- Náročnost samotných prvků - škála od 1 (nejjednodušší) do 5 (nejobtížnější)
  - promenáda (2)
  - sedmičky (2)
  - jelen (4)
  - rocky (5)
- Náročnost posloupností
  - promenáda + promenáda ... obtížnost 0 navíc
  - promenáda + jelen ... 1 navíc
  - jelen + rocky ... 3 navíc
  - sedmičky + jelen ... 2 navíc

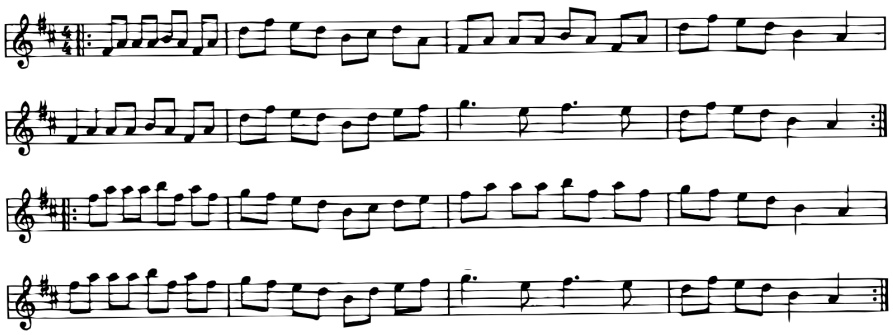
Irská hudba je obvykle stavěna po 32 taktech melodie, která je v některých částech podobná. Uvedu na příkladu skladby Silver Spear vypadá takto:

(viz obrázek 3.18)

Tanec tedy by měl vypadat podobně: na prvních 16 taktů tančit, a na dalších 16 tančit něco jiného ale podobného. Může například zachovávat v těchto 32 taktech nějaký společný taneční prvek, třeba “rocky”.

Nejlépe to je vidět na tradičních tancích nazývaných set, který se stepuje a je navržen přesně do hudby, dokonce se netancuje na jinou než tuto hudbu. Například tanec Blackbird [23].

Pro tanečníky je jednoduché tancovat tanec, kde se obtížné prvky nebo obtížné posloupnosti neobjevují příliš často. Na druhou stranu může tanec vypadat nezajímavě. Proto by algoritmus měl mít monžnost volit míru obtížnosti výsledného tance.



Obrázek 3.23: 8 taktů * 2 (repetice) + 8 taktů * 2 (repetice) = 16 + 16 = 32 taktů

3.3.2.1 Požadavky algoritmu

Cílem algoritmu bude vygenerovat a najít kombinaci tanečních prvků takovou aby obsahovala vybrané taneční prvky a celková náročnost prvká na provedení ve 4 taktech hudby aby nepřesáhla určitou zvolenou mez. Budeme se snažit maximalizovat krásu tance. Krásný tanec je definován jako takový, ve kterém taneční kroky mají co největší odlišnost vůči okolním krokům a to hlavně v místech, kde se melodie písničky mění na jinou. Tuto odlišnost se budu snažit maximalizovat.

Vstupem algoritmu bude:

- zjednodušený zápis hudby, kde budou znázorněny pouze intervaly not a takty
  - případně by se mohl v zápisu objevit příznak pro výraznou změnu melodie (v irské hudbě se obvykle mění po 16 taktech), to lze využít pro nějaký výrazný nový prvek
  - nebo taky brát ohled na to jestli melodie zrovna stoupá nebo klesá
- mez náročnosti vygenerovaných prvků ve 4 taktech
- výběr několika tanečních prvků, které by se měly ve vygenerovaném tanci určitě objevit alespoň jedenkrát

Výstupem algoritmu bude vygenerovaný tanec, který bude nejlépe sedět do hudby, bude obsahovat požadované prvky a splňovat požadovanou náročnost ve 4 taktech hudby.

Předpoklady algoritmus bude následující znalosti:

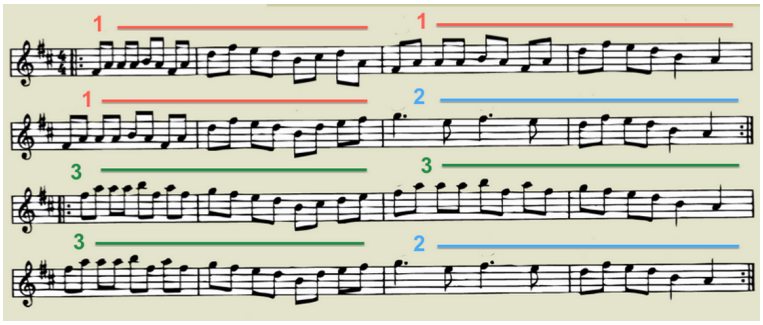
- ohodnocení náročnosti některých kombinací tanečních prvků (samotný prvek může být jednoduchý ale když se zatančí hned po jiném, může to být náročnější)
- seznamu tanečních prvků včetně zaznamenaných intervalů jejich kroků
- případně taky seznamu ručně zadaných pěkných posloupností prvků

Kategorizace algoritmu Půjde o rozvrhovací problém, kde se algoritmus bude snažit rozmístit taneční prvky do hudby tak aby co nejlépe odpovídaly kroky intervalům hudby. Zároveň budou kombinace tanečních prvků vždy splňovat výše uvedená kritéria.

3.3.2.2 Stavba hudby a tanečních kroků

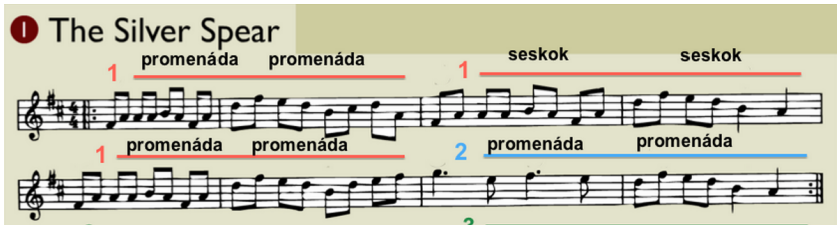
Nejprve ještě zrekapituluji jak je obvykle stavěna hudba a jak kroky.

Irská hudba je tvořena z několika různých melodií, které se různě opakují. Názorně předvedu na skladbě Silver Spear. Barevně jsou označeny (očíslovány) stejné nebo velmi podobné melodie, které se opakují - někdy se opakují těsně za sebou, někdy až později ve skladbě. Nahrávka [24]



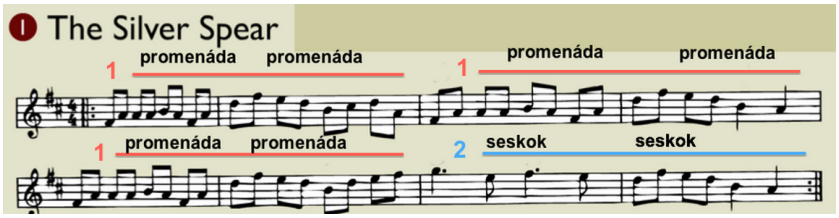
Obrázek 3.24: Ukázka stavby irské skladby. Barevně jsou značeny podobné části melodie.

Kroky irského tance lze zatancovat na jakoukoli melodií. Ovšem pokud se kroky použijí tam kde s melodií souzní, vypadá tanec nejpěkněji. Například je škoda pokud tanec například vypadá takto: v místě kdy se melodie 1 mění na 2, se tančí stále stejný krok



Obrázek 3.25: Ukázka špatného rozložení kroků. Pokud se v místě změny melodie vygeneruje stále stejný krok je to škoda protože generátor dostatečně nevyužívá informací o podobnosti melodií a jejich vývoji

Přitom by mohl vypadat lépe třeba takto: využívám podobné melodie a dávám tam podobné kroky, až když se výrazně změní melodie změním výrazně i kroky. Promenáda a seskok jsou dost odlišné (promenáda je víc pohybová, seskok je na místě)



Obrázek 3.26: Ukázka správného rozložení kroků, kde vždy se změnou hudby se mění také taneční kroky

Dále je vhodné zachovat stejné kroky alespoň na dvou taktech, jinak je tanec pro tanečníky extrémně náročný - bez ohledu na náročnost jednotlivých kroků či jejich kombinací.

3.3.2.3 Jak poznat pěkný tanec

- změní-li se melodie, pak se změní i kroky (nebo směr pohybu), snaha mít co nejodlišnější kroky v těchto změnách (před a po)
- při změně melodie na jinou, nesmí být v nové melodii použity kroky z předchozí melodie
- některé prvky se nesmí opakovat těsně za sebou příliš mnohokrát (např rocky)
- dodržet nějakou celkovou hranici obtížnosti tance

3.3.2.4 Fungování navrženého algoritmu

Pro implementaci rozvrhovacího algoritmu jsem se rozhodl zvolit jako základ genetický algoritmus, který mi dovolí přehledněji a pružněji zformulovat požadovaná omezení při generování než bylo nutné v Integer-Linear Programming.

Implementaci naleznete v přiloženém CD ve složce "java-generovani-tance"v souboru SearchForBeautifulDance.java

Vytvoření počáteční populace

- Spočítám kolik je v hudbě podobných melodií.
- Takto vytvořím několik instancí objektu reprezentující Tanec
  - Pro každou z podobných melodií (jsou identifikovány pořadovým číslem, jako v ilustraci nahoře) náhodně vyberu taneční krok
  - Tento krok se bude opakovat v každém taktu právě vytvářeném tanci, kde se podobná melodie vyskytuje

Evoluce

- Pomocí algoritmu Large Roulette Wheel [26] vyberu část populace určenou k přežití do další generace. Vyšší pravděpodobnost na přežití mají tance s vyšší fitness hodnotou.
- Doplním populaci na původní počet
  - Vytvořím nový tanec křížením dvou nahodných přeživších tanců
  - Tento vytvořený tanec ještě nechám zmutovat
  - Vypočítám a uložím fitness hodnotu tohoto nového tance

Výpočet fitness hodnoty

- Tanec je v rozdělen do částí, jejichž počet a délka (měřená v taktech) je shodná s počtem a délkou melodií ve skladbě.
- Procházím každou část tance
  - sčítám náročnosti kroků v jednotlivých částech
  - sčítám míru odlišnosti mezi jednotlivými kroky při změně melodie
  - počítám zda se krok mění na jiný než původní při změně melodie
  - počítám kolikrát bylo přesáhnut limit na opakování kroku těsně za sebou
  - počítám kolikrát bylo přesáhnut limit na opakování kroku v celém tanci tance
- Z přechozího iterování získám míry, které nyní ještě normalizuji aby vyjadřovaly procenta. Což mi umožní lépe ovládat váhy ve výpočtu výsledné fitness hodnoty
  - normalizedDifficultyScore = difficultyScore / difficultyScoreMax
  - normalizedDistinctionScore = distinctionScore / distinctionScoreMax
  - normalizedStepChangesScore = stepChangesScore / stepChangesScoreMax
- Abych odřízl tance s vysokou náročností využiji extra vysoké penalizace pokud normalizedDifficultyScore přesahuje stanovený limit náročnosti

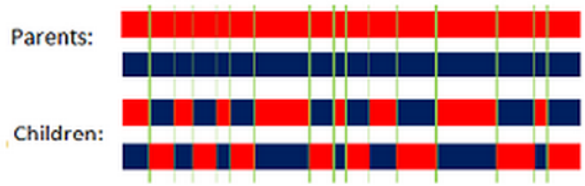
Výsledná fitness hodnota se počítá následovně:
fitnessValue =

- 10000 * normalizedStepChangesScore
  (omezení) posune nahoru tance, které se mění tehdy kdy se mění melodie (ideál je 100
- + 100 * normalizedDistinctionScore
  (maximalizace) posune nahoru tance, které se mají ve změnách melodie co nejodlišnější kroky ideální je 100
- - 1000 * stepRepeatingOverLimitInNeighborhoodScore
  (omezení) posune níže tance, kde bylo překročeno stanovené omezení na opakování kroků těsně za sebou ideální je 0 (nepřesahuje limit)

- - 100 * fm.stepRepeatingOverLimitInWholeDanceScore
  (omezení) posune níže tance, kde bylo překročeno stanovené omezení na opakování kroků v rámci celého tance ideální je 0 (nepřesahuje limit), má nižší váhu než okolí protože překročení zase tolik moc nevadí
- - 1000 * difficultyPenaly
  (omezení) posune níže tance, které nesplnily limit obtížnosti ideální je 0 (nepřesahuje limit)

Operace křížení

- Nastavím "přepínač tanců"náhodně na jeden ze dvou vstupních tanců
  - určuje ze kterého tance se mají kopírovat kroky do nově vytvářeného tance - na způsob železniční vyhýbky
- Iteruji přes všechny části melodie
  - pokud se melodie v této části změnila proti původní části náhodně přepnu "přepínač tanců"některý ze dvou vstupních tanců
  - kopíruji kroky pro aktuální část melodie do nového tance
- Výsledkem je nový tanec zkřížený metodou “Uniform Crossover”



Obrázek 3.27: Náhodně jsou vybrány místa křížení v obou tancích o stejných velikostech. [27]

Operace mutace

- Zajišťuje přísun náhodných kroků, i těch které se v populaci zatím nevyskytují
- Vyberu náhodně několik podobných melodií a přiřadím jim nový náhodný krok
- Příklad:
  - mám části melodií označené identifikátorem podobnosti (např viz skladba Silver Spear nahoře)
  - vyberu identifikátory #1 a #2
  - všem částem tance, které patří pod melodii #1 přiřadí nový náhodný krok
  - všem částem tance, které patří pod melodii #2 přiřadí nový náhodný krok

3.3.2.5 Příklad algoritmem generovaných špatných a dobrých tanců

Špatný tanec 1 Důvod: Příliš mnoho těsně za sebou opakovaných "dup dup". Zároveň překročena náročnost (limit byl 40%). Přitom tanec jinak správně dodržuje odlišnost kroků a jejich výměnu při změně melodie.

```
|^ dup dup  |  dup dup  |^ dup dup  |  dup dup  |^ dup dup  |  dup dup  |^ rockrock |  rockrock
|^ dup dup  |  dup dup  |^ dup dup  |  dup dup  |^ dup dup  |  dup dup  |^ rockrock |  rockrock
|^  seskok  |   seskok  |^  seskok  |   seskok  |^  seskok  |   seskok  |^ rockrock |  rockrock
|^  seskok  |   seskok  |^  seskok  |   seskok  |^  seskok  |   seskok  |^ rockrock |  rockrock
step changes: 100% (7.0)
distinction: 79% (5.5)
step repeating over limit (neighbor): 6.0
step repeating over limit (overall): 0.0
difficulty: 63%
Fitness: 3853.5714285714294
```

Špatný tanec 2 Důvod: Tanec má sice nízkou obtížnost, ale jinak obsahuje velmi málo odlišné kroky a navíc nemění kroky když se mění melodie (v prvním řádku nakonci by v posledních dvou taktech měl být jiný krok protože se tam mění melodie - viz Silver Spear nahoře). Také překročil počet promenád v celém tanci (maximum bylo 12, přitom jich tam je 20).

```
|^ promenad |  promenad |^ promenad |  promenad |^ promenad |  promenad |^ promenad |  promenad
|^ promenad |  promenad |^ promenad |  promenad |^ promenad |  promenad |^ promenad |  promenad
|^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^ promenad |  promenad
|^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^ promenad |  promenad
step changes: 57% (4.0)
distinction: 29% (2.0)
step repeating over limit (neighbor): 1.0
step repeating over limit (overall): 8.0
difficulty: 33%
Fitness: 3942.857142857142
```

Dobrý tanec Důvod: Při změně melodie tanec perfektně dodržuje změnu kroků a snaží se tam o jejich maximální odlišnost. Rovněž není příliš obtížný.

```
|^ promenad |  promenad |^ promenad |  promenad |^ promenad |  promenad |^  seskok  |   seskok
|^ promenad |  promenad |^ promenad |  promenad |^ promenad |  promenad |^  seskok  |   seskok
|^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^  seskok  |   seskok
|^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^ sedmicky |  sedmicky |^  seskok  |   seskok
step changes: 100% (7.0)
distinction: 87% (6.1)
step repeating over limit (neighbor): 0.0
step repeating over limit (overall): 0.0
difficulty: 33%
Fitness: 10087.142857142857
```

3.3.2.6 Experimenty s parametry algoritmu

Hudba Pro experimenty jsem použil skladby Silver Spear a Glasgow Reel. První má jednodušší strukturu a melodi, druhá je naopak složitější.

Základní nastavení

- Počítač: OSX 10.8.2, proc: 2.4 GHz, ram: 4GB DD3
- Prostředí: NetBeans 7.2
- Výchozí parametry hledání:
  - počet evolucí = 30
  - obtížnost = 0.4
  - populace = 100
  - selekce = 0.3

Nastavení kroků:

Krok	Obtížnost [3 ... nejtěžší]	Max. těsné opakování	Max. opakování v celém tanci
Promenáda	1	8	12
Sedmičky	1	8	12
Jelen	3	-	-
Seskok	1	4	-
Dup dup dup	2	2	-
Rock rock rock	3	2	-

Obrázek 3.28: Tabulka popisuje nastavení parametrů kroků pro generování tanců během testování. Uživatel může například snad upřesnit algoritmu, které kroky jsou pro něj obtížné. Algoritmus pak bude vědět které kroky preferovat a kterým se vyhnout. Případně lze využít detailnější specifikování obtížnosti pomocí sloupečků definující maximální těsné opakování případně maximální opakování kroku v celém tanci

Průběh testů Každé nastavení experimentu je spuštěno desetkrát a měřené hodnoty jsou zprůměrovány.

Legenda ke skladbám:

```
^ ... označuje začátek podobné části
číslo ... je identifikátor podobných částí
| ... označuje takt
```

Test jednoduché skladby Silver Spear

```
|^ 1 |  1 |^ 1 |  1 |^ 1 |  1 |^ 2 |  2
|^ 1 |  1 |^ 1 |  1 |^ 1 |  1 |^ 2 |  2
|^ 3 |  3 |^ 3 |  3 |^ 3 |  3 |^ 2 |  2
|^ 3 |  3 |^ 3 |  3 |^ 3 |  3 |^ 2 |  2
```

Experiment	Avg. Elapsed Time [vteřiny]	Avg. Fitness [fitness index]	Avg. Consumed Memory
populace = 5	0.03	10056	0.1 MB
populace = 10	0.03	10066	0.3 MB
populace = 15	0.04	10086	0.5 MB
Výchozí parametry (populace = 30)	0.05	10087	2.3 MB
populace = 300	0.11	10087	3.0 MB
populace = 700	0.36	10087	7.0 MB
populace = 1000	0.49	10087	9.5 MB
populace = 2000	0.81	10087	7.3 MB

Obrázek 3.29: Tabulka popisuje naměřené výsledky získané během testování jednoduché skladby Silver Spear. Tabulka napovídá, že jako nejlepší velikostí populace se jeví populace o velikosti 30. Od této pulace se již nezlepšuje fitness index algoritmu. Proto pro jednoduché skladby bude nejvhodnější velikost populace o třiceti tancích

Test složitější skladby Glasgow Reel

```
|^ 1 |  1 |  1 |^ 2 |^ 1 |  1 |  1 |^ 2
|^ 1 |  1 |  1 |^ 2 |^ 1 |  1 |  1 |^ 2

|^ 3 |  3 |^ 4 |  4 |^ 3 |  3 |^ 5 |  5
|^ 6 |  6 |  6 |  6 |^ 3 |  3 |^ 5 |  5

|^ 1 |  1 |  1 |^ 2 |^ 1 |  1 |  1 |^ 2
|^ 1 |  1 |  1 |^ 2 |^ 1 |  1 |  1 |^ 2

|^ 3 |  3 |^ 4 |  4 |^ 3 |  3 |^ 5 |  5
|^ 6 |  6 |  6 |  6 |^ 3 |  3 |^ 5 |  5
```

Experiment	Avg. Elapsed Time [vteřiny]	Avg. Fitness [fitness index]	Avg. Consumed Memory
Výchozí parametry (populace = 30)	0.04	10088	1.6 MB
populace = 300	0.17	10098	4.8 MB
populace = 700	0.55	10099	9.6 MB
populace = 1000	0.84	10099	14.2 MB
populace = 2000	-	-	out of memory

Obrázek 3.30: Tabulka popisuje naměřené výsledky získané během testování složitější skladby Glasgow Reel. Tabulka napovídá, že jako nejlepší velikostí populace vychází opět populace o velikosti 30. Protože od této poulace výše se fitness index sice zlepšuje pouze do populace 300, avšak paměťová náročnost algoritmu stoupá do vyšších čísel. Proto rovněž pro složitější skladby bude vhodné volit populaci o velikosti třiceti tanců

Experiment	Avg. Elapsed Time [vteřiny]	Avg. Fitness [fitness index]
Výchozí parametry populace = 1000 počet evolucí = 30	0.84	10099
počet evolucí = 60	1.9	10100
počet evolucí = 100	2.7	10100

Obrázek 3.31: Tabulka popisuje naměřené výsledky získané během testování dodatečného experimentu zda lze dostáhnout vyšší fitness funkce jiným způsobem. Test zkusil vybrat populaci kdy bylo dosaženo nejvyššího fitnes indexu. Následně jsem zkoušel zvyšovat počet evolucí abych zjistil zda se nepovede vygenerovat tanec s ještě vyšším fitness indexem, bohužel se fitnes index již příliš mnoho závratně nezvyšoval, proto opět vychází jako nejlepší konfigurace 30 evolucí při velikosti populace 30 tanců

Ukázka dvou nejlepších vygenerovaných tanců .

```
- Best:
 |^  seskok  |   seskok  |   seskok  |^ dup dup  |^  seskok  |   seskok  |   seskok  |^ dup dup
 |^  seskok  |   seskok  |   seskok  |^ dup dup  |^  seskok  |   seskok  |   seskok  |^ dup dup
 |^ sedmicky |  sedmicky |^ promenad |  promenad |^ sedmicky |  sedmicky |^  jelen   |   jelen
 |^ promenad |  promenad |  promenad |  promenad |^ sedmicky |  sedmicky |^  jelen   |   jelen
 |^  seskok  |   seskok  |   seskok  |^ dup dup  |^  seskok  |   seskok  |   seskok  |^ dup dup
 |^  seskok  |   seskok  |   seskok  |^ dup dup  |^  seskok  |   seskok  |   seskok  |^ dup dup
 |^ sedmicky |  sedmicky |^ promenad |  promenad |^ sedmicky |  sedmicky |^  jelen   |   jelen
 |^ promenad |  promenad |  promenad |  promenad |^ sedmicky |  sedmicky |^  jelen   |   jelen
step changes: 100% (29.0) distinction: 83% (24.1) step repeating over limit (neighbor): 0.0
step repeating over limit (overall): 0.0 difficulty: 46%
Fitness: 10024.770114942528

- Best:
 |^ sedmicky |  sedmicky |  sedmicky |^  seskok  |^ sedmicky |  sedmicky |  sedmicky |^  seskok
 |^ sedmicky |  sedmicky |  sedmicky |^  seskok  |^ sedmicky |  sedmicky |  sedmicky |^  seskok
 |^ dup dup  |  dup dup  |^  seskok  |   seskok  |^ dup dup  |  dup dup  |^  seskok  |   seskok
 |^ promenad |  promenad |  promenad |  promenad |^ dup dup  |  dup dup  |^  seskok  |   seskok
 |^ sedmicky |  sedmicky |  sedmicky |^  seskok  |^ sedmicky |  sedmicky |  sedmicky |^  seskok
 |^ sedmicky |  sedmicky |  sedmicky |^  seskok  |^ sedmicky |  sedmicky |  sedmicky |^  seskok
 |^ dup dup  |  dup dup  |^  seskok  |   seskok  |^ dup dup  |  dup dup  |^  seskok  |   seskok
 |^ promenad |  promenad |  promenad |  promenad |^ dup dup  |  dup dup  |^  seskok  |   seskok
step changes: 100% (29.0) distinction: 98% (28.4) step repeating over limit (neighbor): 0.0
step repeating over limit (overall): 0.0 difficulty: 40%
Fitness: 10097.931034482759
```

3.3.3 Shrnutí

Pro detekci hudebních dob jsem zvolil jako nejvhodnější metodu detekce pomocí comb filteru, protože vykazuje nejstálejší výsledky na testovaných skladbách. Rovněž by implementace měla být možná s pomocí standardní vestavěné vDSP knihovny na iOS.

Během rešerše vedlejších problémů jsem se pokusil prozkoumat možnosti jakými by bylo možné tyto problémy řešit. Zkusit vybrat ideální knihovnu použitelnou pro mobilní zařízení, případně zjistit ideální konfiguraci vhodnou pro přenosné zařízení, které mají omezenější paměťové a výpočetní možnosti.

Při hledání možností pro změnu rychlosti hudby při zachování výšky tónů jsem narazil na několik možností. Bylo možné algoritmy pokusit se implementovat sám pomocí dostupné na mobilních zařízeních vestavěné knihovny vDSP Accelerate. Ovšem našel jsem rovněž množství knihoven, které problém zpomalení dovedou řešit pomocí pokročilejších metod mnohem kvalitněji. Napsání vlastního kódu řešící problém alespoň srovnatelně kvalitně by vyžadovalo nejvhodnější knihovny založené na metodě SOLA. Proto jsem vybral knihovnu SoundTouch. Knihovna je k dispozi pro více platforem a navíc je možné studovat použitý kód, díky open source licenci.

Algoritmus pro výběr vhodných tanečních prvků do anotované hudby se mi povedlo sestavit pomocí genetického algoritmu, který umožnil vhodně formulovat potřebná omezení. Metody křížení a mutace jsem jsem naprogramoval tak aby vhodně vytvářely nové netestované variace tance. Definování fitness funkce bylo poměrně náročné, ale podařilo se mi do definice zakomponovat většinu požadovaných omezení na vhodně složitý tanec, na počet opakování určitých kroků či omezení na opakování posloupností velmi obtížných tanečních kroků. Testy ukázaly správné a očekávané chování algoritmu.

3.4 Rešerše implementační platformy

3.4.1 Mobilní zařízení

Mobilní zařízení svým charakterem klade určitá omezení jak uživateli, tak také vývojáři. Velice omezený prostor nutí vývojáře navrhovat uživatelské rozhraní tak, aby obsahovalo pouze nejnutnější věci pro splnění daného případu užití. Případy užití by měly být vždy před implementací a návrhem revidovány a redukovany na množiu, kterou je vhodné řešit pouze pro platformu. Pokud se aplikace pokouší obsáhnout široké spektrum funkcí, je zde velmi vysoké riziko nízkého přijetí mezi koncovými uživateli. Je tedy velmi podstatné důkladně vybírat a zjišťovat co uživatele zajímá a co je pro ně prioritou. Výpočetní výkon je u mobilních zařízení často nižší než na desktopových stanicích, proto musí být výpočetně náročnější procesy testovány a optimalizovány, aby neznepříjemňovaly používání aplikace na zařízeních uživatele. Ve světě mobilních operačních systémů je běžné, že v jednom okamžiku je spuštěna obvykle pouze jedna aplikace. Tím, že aplikace běží téměř samostatně dovede zařízené lépe organizovat své zdroje a aplikace se tak jeví jako rychlejší, než by mohla být na desktopové stanici, kde současně s ní musí počítač odbavovat také řadu dalších aplikací.

Apple pro tvůrce aplikací připravil vyčerpávající příručku Human Interface Guidelines [11]. Platforma iOS má několik zvláštností, které je potřeba brát při návrhu uživatelského rozhraní v potaz. Myslím, že většina poznámek je přínostná i pro jiné mobilní platformy.

Základní principy Snažit se návrh zakládat na tom jak lidé myslí a pracují, spíše než na možnostech zařízení a všech jeho funkcích.

- Používat metafory (přepínač, připínáček na mapě,...)
- Přímou manipulaci s objekty (dotyková gesta pro zvětšování fotografie, nebo její rotaci - takto bych otáčel i skutečnou fotografii).
- Zpětná vazba po nějaké akci/kontaktu musí být okamžitá.
- Akce pro ovládání aplikace by měla přicházet od uživatele, nikoli od aplikace samotné.
- Aplikace má být krásná a esteticky celistvá (jak dobře se vzhled propojuje s funkcionalitou).
- Všechny ovládací prvky použitelné v aktuálním kontextu by měly jít vidět.

Jednoduchost používání Při návrhu aplikace se doporučuje řídit se heslem "when on doubt, keep it simple"(tedy nejsem-li si jistý, raději vyškrtám nejasné věci a zachovám jen to nejjednodušší). Snaha ořezávat seznam funkcí na minimum a najít klíčové činnosti. Pokud přenáším nápady z desktopu na mobil použiji pravidlo 80-20 (největší procento lidí používá často jen několik velmi málo funkcí).

Nejdůležitější a několikrát opakovaná zásada je ilustrována následující situací:

- představte si, že člověk jde po městě a vytáhne iPhone z kapsy, rychle něco zjistí a pak iPhone zase schová

- uživatelé často dělají několik věcí současně když používají iPhone (mluví, drží se v tramvaji, jedou pozdě autem, hrajou na kytaru, vaří...)
- aplikace by neměla zdržovat, měla by se soustředit pouze na vyřešení určité momentální věci a nerozptylovat člověka ne tolik důležitými věcmi

Dělat věci tak aby byly uživatelům zřejmé a na první pohled patrné. Například pokud bych chtěl zapnout televizi přes mobil, stačí dát do středu větší zapínací tlačítko, ostatní je v tu chvíli zbytečné, proto ostatní prvky uživatelského rozhraní skrýt.

Minimalizovat vstup textu požadovaného po uživateli. Uživatelé neradi píšou, je to pro ně vzhledem ke kontextu používání příliš zdlouhavé a obtížné. Pokud lze něco vybrat v seznamu, nebo automaticky odhadnout, je téměř vždy nejlepší volbou udělat to za uživatele. Uživatel tuto iniciativu velmi ocení a bude za ni vděčný, protože vstup textu mu trvá vždy nejdéle při používání mobilního zařízení. Poslední dobou je čím dál snadnější používat pro vstup hlas, který automat převed do textové podoby.

High-level informace k určité činnost nebo situaci by měla být vždy umístěna v horní části uživatelského rozhraní aplikace. Nezávislé na tom kam lidé tapají, hořejšek je prostě nejviditelnější.

Další důležitosti Všechny prvky by měly být minimálně 44x44 pixelů veliké aby je šlo snadno stisknout.

Zaměřit se na hlavní úkol. Nabídnout uživateli v rozhraní jen ty informace a ovládací prvky, které v aktuálním kontextu právě potřebuje. Určit co je kdy nejdůležitější.

Omezit technický žargon, zvolit slovník blízký cílové skupině.

Používat gesta na vhodných místech a pro vhodné akce.

Start aplikace Měla by nastartovat okamžitě bez zbytečného zdržování. Nepoužívat úvodní obrázky, nebo načítací animace - nejlépe použít jako úvodní obrázek kostru rozhraní bez tlačítek, textu a obsahu. Zohledňovat orientaci displeje při startu.

Ukončení aplikace Nikdy by se aplikace neměla ukončit zevnitř. Vždy se toto má nechat na uživateli, má od toho kulaté tlačítko pod displejem. Doporučuje se uložit stav uživatele před vypnutím. Aplikace by měla být vždy připravena, že může být kdykoli ukončena.

Nastavení Nejlépe pokud aplikace nevyžaduje vůbec žádné nastavení ze strany uživatele. Doporučuje se zaměřit na potřeby 80% uživatelů. Získávat maximum informací z nejrůznějších zdrojů (GPS, počasí, dřívější používání...).

Zvuky Uživatel by si měl sám zvolit hlasitost. Pokud má uživatel sluchátka, chce být nejspíše nenápadný (nepouštět alarm z hlavních reproduktorů ale do sluchátek). Brát ohled ztišený režim.

Omezení Paměť je omezená, pokud paměť přestává stačit systém se snaží najít další a pokud ani pak nestačí je aplikace ukončena.

Aplikace nemají explicitně přístup k ostatním nainstalovaným aplikacím a systému. Určitě ne přímo k jejich filesystému. Přistupují k nim pouze omezeně přes speciální knihovny.

Jediná právě používaná aplikace. Systém neumožňuje zároveň používat více aplikací. Nepoužívané aplikace jsou pozastaveny na pozadí. Ve vyjímečných případech můžou vykazovat známky života i při používání jiné aplikace, ale i tehdy jejich činnost putuje přes nějakou oficiální Apple aplikaci (například hudba puštěná z aplikace A, se pustí přes oficiální aplikaci iPod a pak lze hudbu poslouchat ve kterékoliv jiné aplikaci).

Jedna právě používaná obrazovka. Aplikace v určitém okamžiku řeší právě jednu věc. Neměly by zobrazovat více informací než je nutné. Pomůžou tak člověku se rychleji rozhodout a ušetří mu čas. Tvůrce aplikace musí promyslet všechny volby za uživatele a dát mu na výběr jen ty relevantní a podstatné.

Minimální podpora nápovědy v aplikaci. Ideální aplikace nepotřebuje nápovědu, protože všechno v ní dává smysl. Uživatel platformy je zvyklý a naučený z oficiálních aplikací i ze spousty soukromých na určité chování jednotlivých UI komponent, proto je zbytečné předvádět jak se ovládají. Naopak toho lze využít a začlenit do aplikace nějaký prvek (možná neobvyklý na jiných platformách) a uživatel obvykle hned ví jak ovládat. Například “swipe to delete” je funkce kdy při swipe gestu přes položku v tabulce je možné zobrazit tlačítko pro smazání této položky. Funkce je známá ze všech oficiálních aplikací, proto uživatelé pokud něco budou chtít v naší aplikaci smazat, budou nejspíš nejprve zkoušet toto gesto zda funguje i zde. Můžeme toho využít a zajistit tím tak například nějakou po aplikaci požadovanou funkcionalitu.

Samozřejmě i chování aplikace by mělo být předvídatelné. Například v aplikaci pro zasílání SMS v obrazovce s vypsanou poslední konverzací po tapnutí na tlačítko pro napsání zprávy budu jako uživatel očekávat že budu psát přímo člověku v této konverzaci a že nebudu muset znova vybírat příjemce zprávy.

3.4.2 Promises

Kvalitní aplikace, nejen mobilní, se vyznačuje rychlou responzivitou na interakci uživatele a dovede průběžně uživatele informovat o své činnosti. Toho lze dosáhnout vhodnou architekturou aplikace.

Obvykle v mobilní iOS aplikaci běží dvě vlákna. První nazývané "hlavní vlákno"obsluhuje vykreslování a události v uživatelském rozhraní. Druhé a další vlákna obstarávají různé další buď systémové služby (audio, video, dsp zpracování...) nebo vlastní služby aplikace.

Ideální aplikace pokud obdrží ke zpracování událost interakce uživatele, pak by měla práci delegovat na pozadí a umožnit tak hlavnímu vláknu vyřídit další interakce. Pokud se to v aplikaci neděje, obvykle dochází ke zpomalování animací, zamrzávání rozhraní, což ubírá na popularitě a použitelnosti aplikace.

iOS ekosystém nabízí možnost nechat běžet kód na vlákně v pozadí s nižší prioritou, což velmi odlehčí hlavnímu vláknu. Ovšem ve složitějších aplikacích, kdy po činnosti A se má vykonat B nebo C, a až se obě splní vykonat ještě práci D - v těchto aplikacích se pak stává zdrojový kód velmi nečitelným a špatně udržovatelným. Typickým příznakem je tzv. callback

hell [28] - kdy se například volání anonymních funkcí vnořuje stále hlouběji a hlouběji. Do takového kódu je velmi obtížné přidávat další logiku.

Moderním návrhovým vzorem řešícím problém callback hell jsou Promises. Hlavní myšlenkou promises je uzavřít kód do objektu Promise, který našemu kódu poskytne dvě funkce. První funkci "fulfill"zavoláme pokud náš kód skončí úspěšně. Druhou funkci "reject"zavláme pokud náš kód skončí chybou, případně je někdy možné rovnou vyhodit vyjímku. Objekt Promise poskytuje navíc další metody, které nám dovolí pracovat s vývojem naši rozhodovací byznys logiky.

Metoda then Pokud na objektu promise zavoláme "then", zajistíme, že v případě úspěchu kódu uvnitř Promise, se nám zavolá jiný kód, který vložíme jako argument metodě "then".

Metoda catch Blok metody catch se provede v případě selhání promise. Pokud je zřetězeno více promises pak se provede kód v nejblíšší catch metodě. Promises mezi chybovou a mezi catch metodou jsou přeskočeny a neprovedou se vůbec.

Metoda when Další užitečnou metodou je "when", která nám dovolí zavolat kód až poté co jsou splěny všechny promises které této metodě předáme jako pole.

Metoda finally zajistí aby byl blok proveden v každém případě po skočení promise, bez ohledu na to zda promise skončila úspěšně či nikoli

Mezi knihovny nabízející podporu Promises je v prostředí iOS knihovna PromiseKit. Je k dispozici ve verzi pro Swift i pro Objective-C. [29]

```
PMKPromise *promise1 = [NSURLConnection GET:@"http://placekitten.com/100/100"];
PMKPromise *promise2 = [UIView promiseWithDuration:0.3 animations:^{
    self.frame = CGRectOffset(self.frame, 0, 200);
}];

[PMKPromise when:@[promise1, promise2]].then(^(NSArray *results){
    UIImage *kittenImage = results[0];
    NSNumber *animationCompleted = results[1];
});
```

Obrázek 3.32: Ukázka kódu zapsaného pomocí promises. Po splnění promise1 a promise2 bude vykonán kód uvnitř metody then

3.4.3 Shrnutí

V cílové aplikaci bude vhodné využít návrhového vzoru Promises a nejlépe prostřednictvím knihovny PromiseKit pro iOS. Pomůže mi předejít vnořování anonymních funkcí do hloubky (callback hell). Zároveň tím bude podpořena myšlenka kvalitní aplikace, tedy, že je aplikace responzivní a časově náročný kód se provádí ve vlákně na pozadí.

Kapitola 4

Návrh řešení

Cílem diplomové práce je vytvoření aplikace pro výuku irského tance. Při návrhu vhodně řešené podoby aplikace jsem postupoval postupovat nejprve od načrtnutí hrubé podoby uživatelského rozhraní a toku obrazovek. Tyto ranné náčrtky jsem podrobil, krátkým testům srozumitelnosti na mých kamarádech a blízkých rodinných příslušnících. Důkladnější testování jsem provedl později ještě na několika reprezentantech cílové skupiny aplikace.

Po ujasnění podoby uživatelského rozhraní budu uvažovat nad způsobem jak prvky rozhraní oživit pomocí řídícího kódu. Pokusím se identifikovat modelové třídy a místa, kde by se kód velmi pravděpodobně opakoval. Průběžně budu uvažovat nad celkovou architekturou aplikace, zda podporuje přehlednost a modulárnost dílčích součástí.

Zařazení návrhu uživatelského rozhraní na začátek procesu návrhu aplikace má svůj význam. Nakreslení rozhraní je relativně levné, a průběžné testování s uživatelem nám odhalí, které části jsou jak pro uživatele důležité. Kreslení dovolí více experimentovat a objevovat nové způsoby jak uživateli srozumitelně sdělit potřebné informace. Díky iteracím testování lze experimenty korigovat a utvářet tak rozhraní, které obsahuje jen ty nejdůležitější prvky. Tyto poznatky nám pak dovolí lépe prioritizovat programování, které je časově náročnější a citlivější na změny. [30]

4.1 Uživatelské rozhraní

4.1.1 Persony

Z cílové skupiny vychází dvě základní skupiny lidí, kterých se vyvíjená aplikace bude týkat. Některé vlastnosti mají vzájemně společné, protože vycházejí z prostředí s podobnou motivací - láskou k tanci. Ovšem jejich další povinnosti je pak už dosti odlišují.

Persona 1 - učitel

- návyky
  - před tréninkem či tancovačkou si připravují tance, které budou hlásit a vyučovat
  - hlásí kroky natolik včas aby je stihli tanečníci pochopit a zatančit

  - opravuje tanečníka pokud u něj vidí chybu
  - vyučuje jednotlivé kroky i celé tance
- potřeby
  - mít spolehlivý a slyšitelný vlastní hlas
  - mít přehled v taktech hudby
  - dovést se zorientovat v tanci
  - hledat chyby u tanečníků a opravovat je
- současná praxe
  - musí se soustředit zároveň na volání kroků do hudby a zároveň na opravování studentů
  - pokud má tichý hlas studenti neslyší volání a dělají chyby
- dovednosti
  - dovede ovládat mobilní telefon

Persona 2 - tanečník

- návyky
  - poslouchat volání tanečních kroků
  - poslouchat hudbu a její doby
  - tančit do hudby
- potřeby
  - uvědomit si taneční krok před tím než jej má zatančit
- dovednosti
  - občas ovládat mobilní telefon, s potížemi

4.1.2 Testovací scénář

Obecně při průběhu každého testování jsem se snažil dávat pozor abych otázky a poznámky nestavěl takovým způsobem abych testerům nepodsouval mé myšlenky. Minimálně jsem napovídal význam prvků na obrazovce, jejich význam a průchod aplikací.

Před každým testovacím sezením připravím a seřadím papírové náčrtky za sebe, tak aby nahoře byla vidět jen jediná obrazovka. Cílem je nerozptylovat testera dalšími vjemy.

První kolo testování bude provedeno v klidném prostředí a prototyp bude ovládat sedíc u stolu. Uživatel bude mít na rozmyšlení dostatek času.

1) Popíšu testerovi situaci, aby si představil, že je učitel na lekci irských setových tanců, a že by rád svým studentům pustil nápovědu k druhé figuře tance Connemara.

2) Opět seřadím náčrtky do výchozího pořadí. Popíšu testerovi situaci, aby si představil, že by rád vygeneroval tanec.

3) Opět seřadím náčrtky do výchozího pořadí. Testerovi povím, že nyní nemá žádný speciální úkol, nechám průchod aplikací čistě na jeho zvídavosti. Tento způsoby by mohl pomoci poznat jak lidé můžou uvažovat při ovládání.

Druhé kolo testování bude testovat chování v zátěžových podmínkách. Testera umístím do rušnějšího a stísněného prostředí plné hospody a ovládání prototypu bude probíhat ve stoje. Rovněž požádám testera aby před začátkem testování vyrazil doběhnout na konec ulice a zpět, čímž jsem chtěl docílit podobného pocitu, který má učitel pokud zrovna dotančil a vybírá další figuru k tanci.

1) Popíšu testerovi situaci, aby si představil, že je učitel na lekci irských setových tanců, a že by rád svým studentům pustil nápovědu k druhé figuře tance Connemara.

2) Opět seřadím náčrtky do výchozího pořadí. Popíšu testerovi situaci, aby si představil, že by rád vygeneroval tanec.

3) Opět seřadím náčrtky do výchozího pořadí. Testerovi povím, že nyní nemá žádný speciální úkol, nechám průchod aplikací čistě na jeho zvídavosti. Tento způsoby by mohl pomoci poznat jak lidé můžou uvažovat při ovládání.

4.1.3 Testování s uživateli

Papírové náčrtky uživatelského rozhraní jsem dříve testoval pouze na dvou reprezentantech (učitel a tanečník). Zpětně jsem si uvědomil, že pro objektivnější testování rozhrání by bylo vhodnější zopakovat dodatečně testování na širší okruhu uživatelů. Kontaktoval jsem tedy tedy několik českých tanečních skupin a oslovil celkem 3 další učitele a 8 dalších tanečníků.

Během uplynulého letního semestru jsem rovněž požádal o zpětnou vazbu k již hotové aplikaci učitele z německa, anglie, irska a švýcarska. Opakovaně jsem zkoušel různé hlasové varianty nápovědy na irských setových tanečnících univerzity třetího věku při ČVUT.

4.1.4 Low fidelity prototyp a výsledky testování

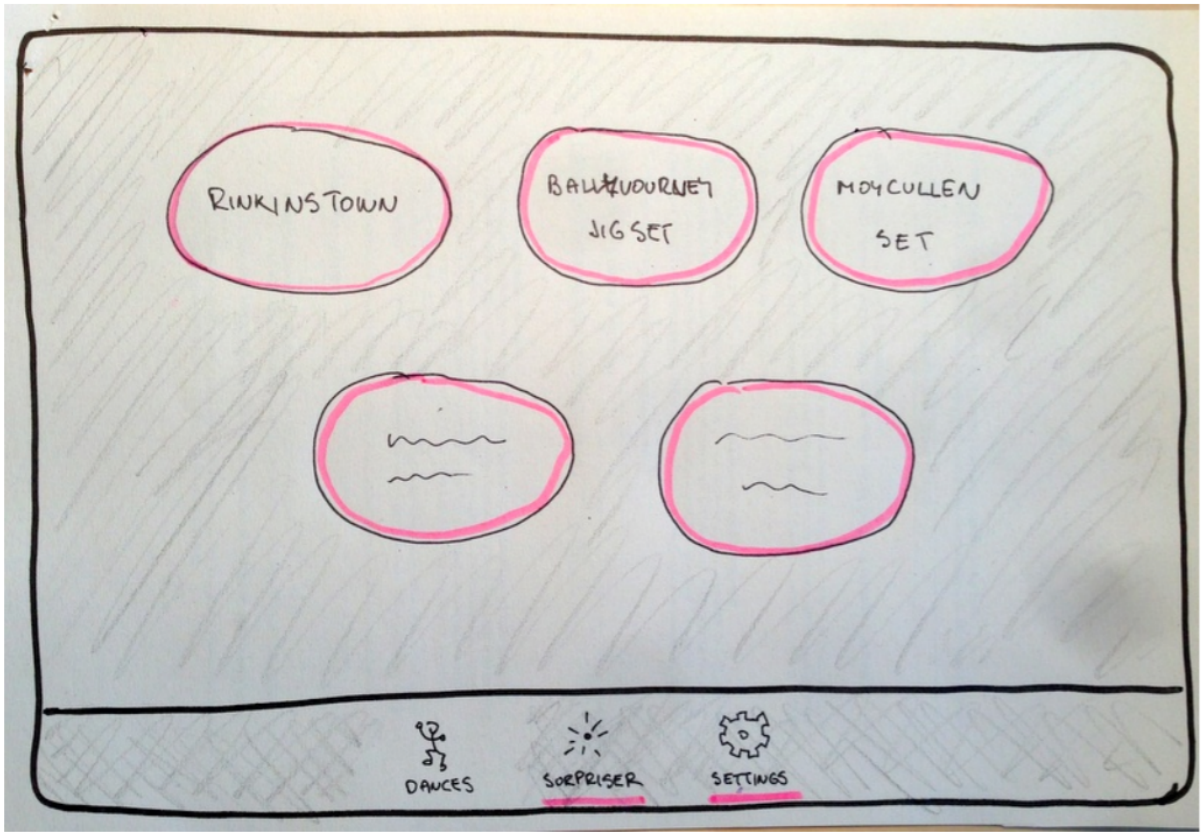
Low fidelity prototyp je tvořen několik náčrtky na papíře. Prvotní náčrtky vznikly syntézí případů užití do uživatelského rozhraní s ohledem na omezení daná platformou iOS.

4.1.4.1 Výběr tanců

Popis Obrazovka zobrazuje výpis tanců uložených v aplikaci. Po kliknutí na některý z tanců se zobrazí detail tance. Pokud je tanců více lze scrollováním zobrazit další tance. Menu ve spodní části umožňuje přepínat se mezi sekcemi aplikace (generování kroků, nastavení)

Zpětná vazba testování Seznam byl pochopen správně a připadal testerům přehledný. Přepínání mezi sekcemi aplikace bylo jasné také, nejspíš protože jde běžný princip používaný napříč více iOS aplikacemi. Uživatelům nebylo příliš jasné co bude v nastavení a laikům (persona 2) nebyl jasný význam generování kroků a tudíž je nelákalo se tam přepnout.

.

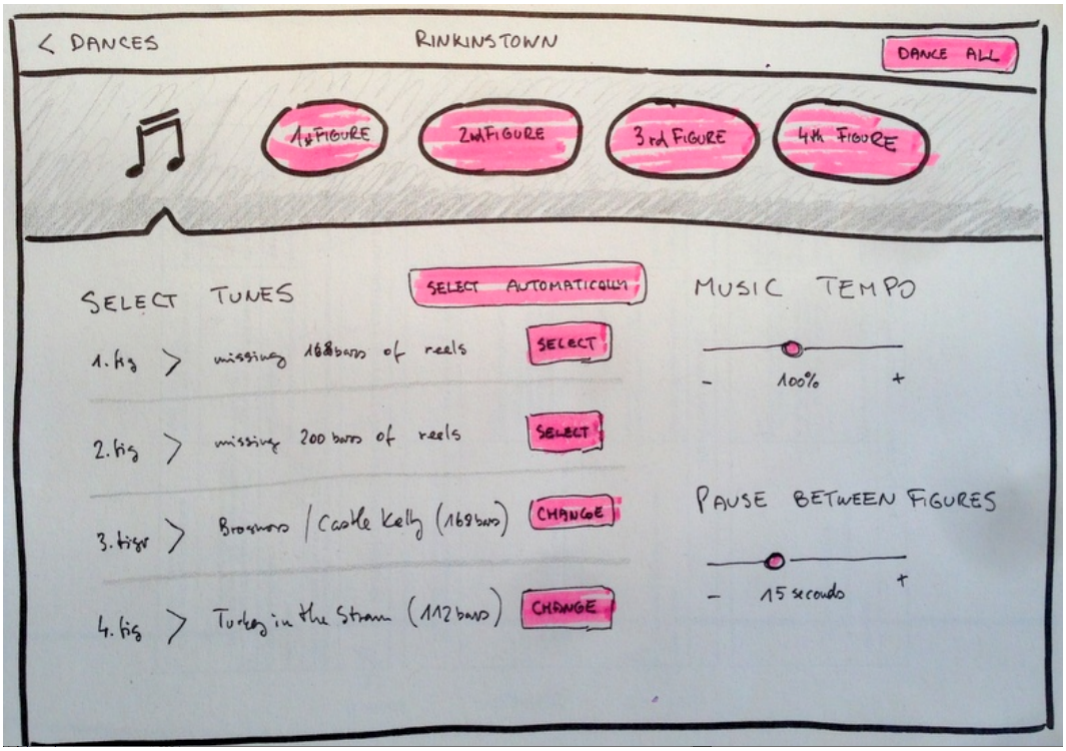


Obrázek 4.1: Prototyp obrazovky pro výběr tanců

4.1.4.2 Detail tance a figury

Popis Obrazovka umožňuje uživateli vybrat hudbu ke každé figuře. Rovněž zvolit její tempo a pauzy mezi jednotlivými figurami. Tlačítko nahoře spustí přehání celého tance od začátku až do konce napříč všemi figurami. Kliknutím na kulatý název figury se zobrazí popis kroků figury.

Zpětná vazba testování Testerům bylo dosti nepříjemné, že hned vybírají hudbu k figurám, přitom ve všech případech je nejprve zajímalo jaké taneční kroky jsou v provní figuře, proto bude obrazovku s výběrem hudby nějakým způsobem zredukovat a integrovat ve zmenšené podobě do detailů jednotlivých figur. Vybrat hudbu je nutné nejpozději u figury kterou chci tančit. Dále tlačítko "dance all"většině testerů nevyhovoval, spíše by čekali, že si zatančí jednu figuru a pak si vyberou ručně další. Jeden tester poznamenal, že by bylo dobré aby aplikace nedovolila spustit figuru k tanci pokud není vybrána hudba - v cílové aplikaci tedy toto naimplementujeme formou Alert okna před spuštením figury k tanci. Za zbytečné považovali zejména učitelé nástroje pro zpomalení hudby s argumentem, že na během volání pomalejší hudba přidanou hodnotu nemá - vhodnější je pomalá hudba bez volání a pro tento účel již existují samostatné aplikace.



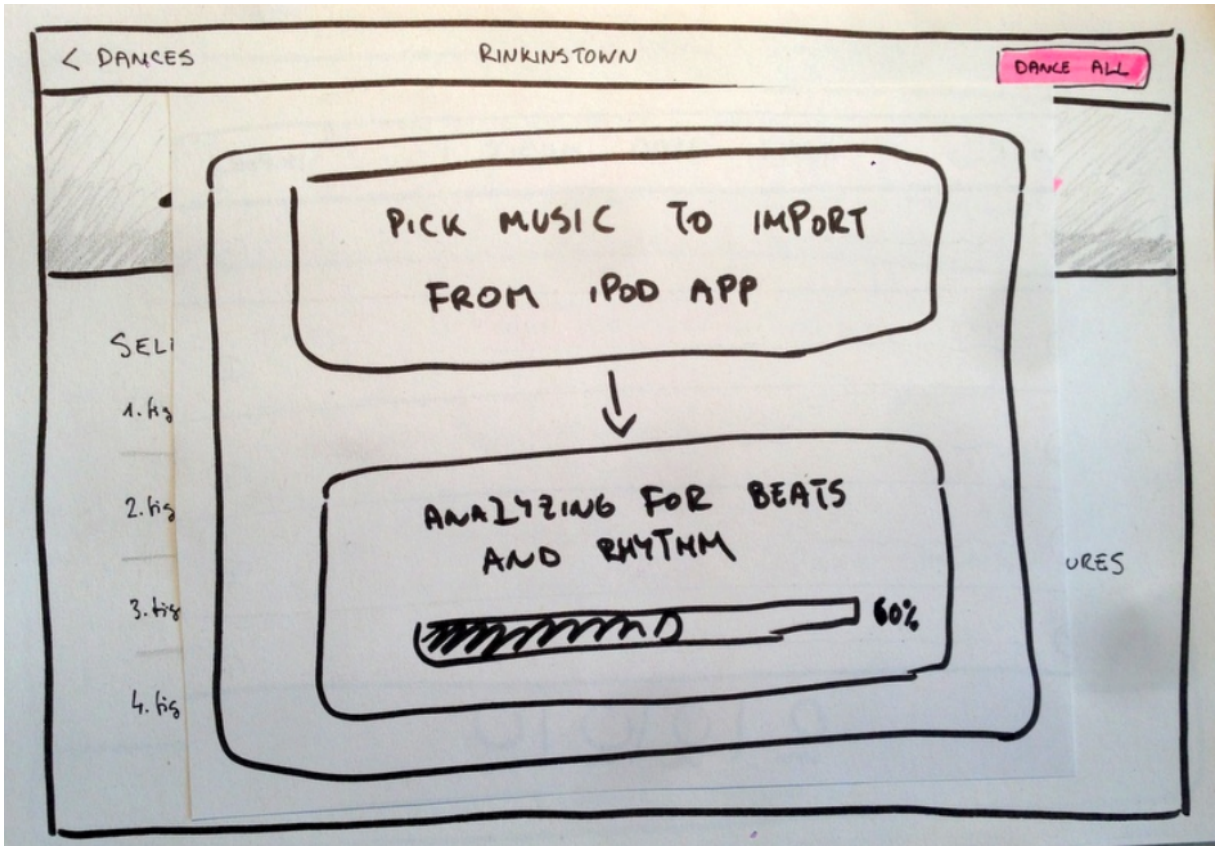
Obrázek 4.2: Prototyp obrazovky pro zobrazení detailu tance a figury

4.1.4.3 Výběr hudby k figuře

Popis Zobrazuje uživateli seznam dostupných skladeb, ve kterých již byly rozpoznány doby a rytmus. Skladby je možné filtrovat podle rytmu a u skladeb s dostatečným počtem taktů je zobrazeno tlačítko vybrat. V horní části menu tlačítko plus zobrazí iPod hudební knihovnu a dovolí vybrat uživateli skladby pro import do aplikace. Po importu skladeb se automaticky zobrazí obrazovka se stavem analyzovaných skladeb. Tlačítko přehrát přehraje úryvek skladby.

Zpětná vazba testování Tlačítko pro přidání bylo naprosto jasné všem, filtrování dle rytmu také. Uživatelé žádali o vylepšení aby byla aplikace schopná importovat najednou více než jen jednu skladbu - tento dobrý a užitečný nápad jsem následně zapracoval. Uživatelům by bylo příjemné kdyby se při výběru rytmu automaticky spustilo přehrávání úryvku skladby. Tlačítko pro přehrání samotné nikdo z testerů nevyužil, proto jej v příští verzi vypusím.

.



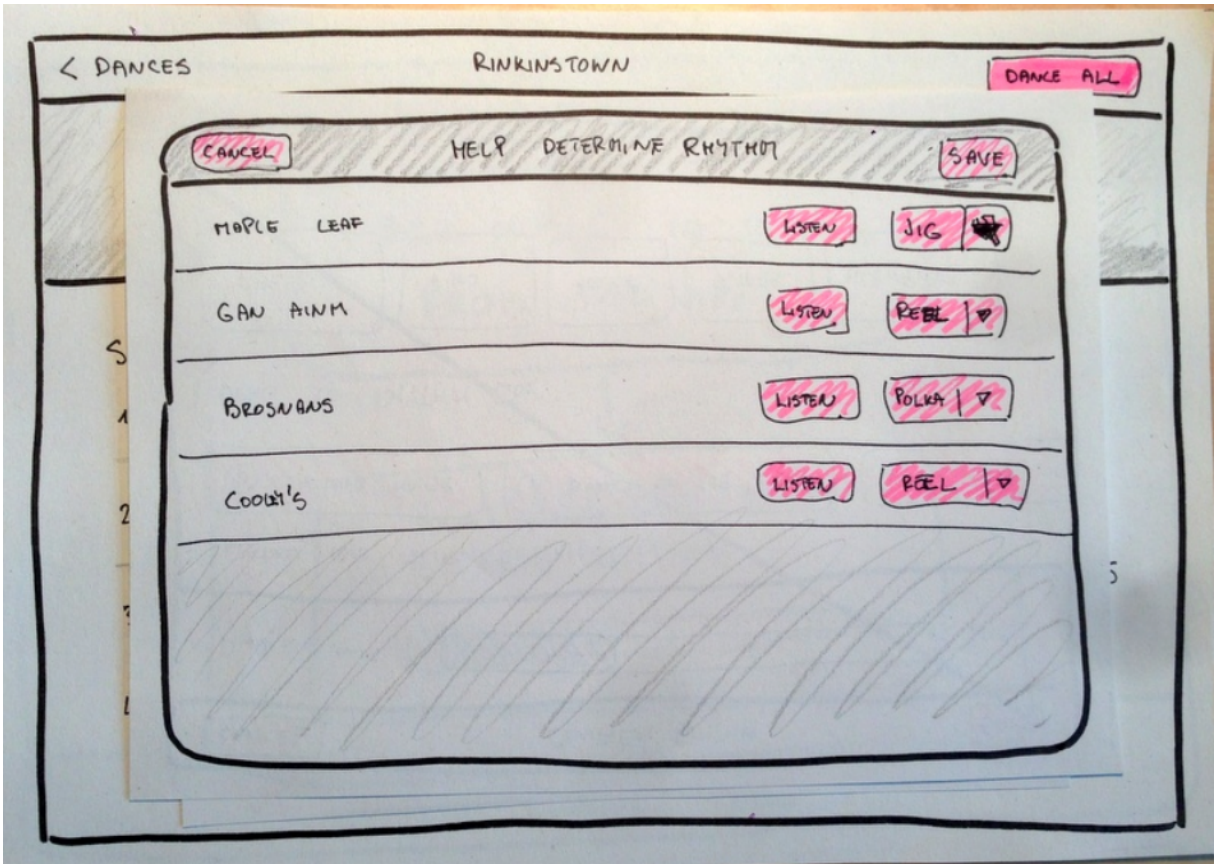
Obrázek 4.3: Prototyp obrazovky pro výběr hudby k figuře

4.1.4.4 Stav právě analyzovaných skladeb

Popis Seznam znázorňuje stav a průběh analýzy jendotlivých importovaných skladeb

Zpětná vazba testování Rozložení prvků na obrazovce bylo uživatelům jasné a přehledné. Nedocházelo ke zmateným situacím. Ovšem zaznamenal jsem během testování, jeden vzor v chování uživatelů, který se opakoval velmi často. Totiž při dodatečném upřesnění výběru rytmu uživatelé téměř pokaždé nejprve klikli na tlačítko "Listen"pro přehrání skladby, aby zjistili příslušný rytmus a následně zvolili rytmus ve vedlejším roletkovém ovládacím prvku. Poznamenal jsem si do poznámek k vylepšení, že by bylo vhodné zrušit tlačítko listen a automaticky rovnou spustit přehrávání hudby v momentě, kdy uživatel klikne na výběr rytmu v roletce. Dalším návrhem bylo zobrazení informace o průběhu analýzy jednotlivých staveb, v prototypu byly totiž zobrazeny pouze názvy skladeb. Myslím, že bude příhodné zobrazit například procentem vyjádřený průběh detekce hudebních dob.

.



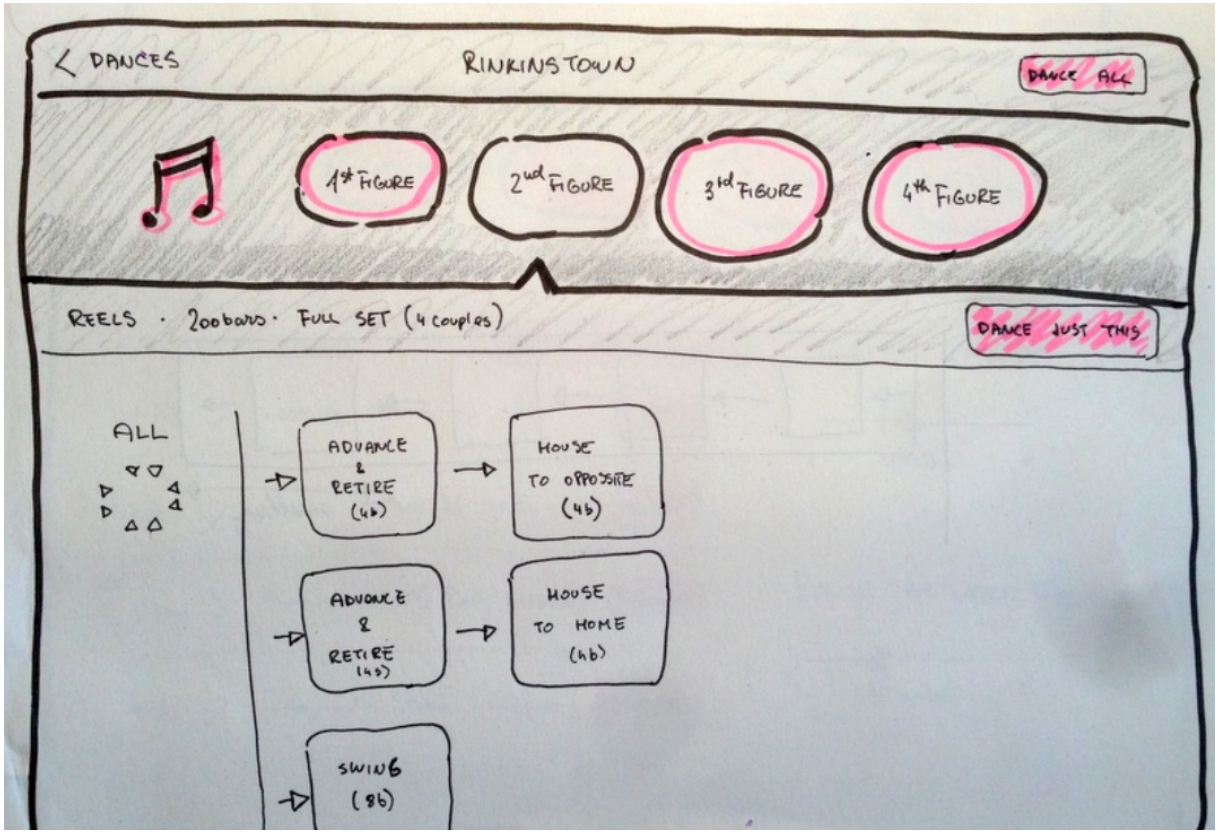
Obrázek 4.4: Prototyp obrazovky pro zobrazení stavu právě analyzovaných skladeb

4.1.4.5 Popis kroků figury

Popis Zobrazuje rytmus figury a počet taktů, také tlačítko pro spuštění tančení pouze této figury. Většinu zbývající obrazovky pak zabírá popis tanečních kroků v aktuální figuře.

Zpětná vazba testování Testeři jak učitelé tak začátečníci, přiliš nerozuměli zobrazení tanečních kroků a často by mnohem raději uvítali běžné zobrazení kroků jako je zavedené například na DanceMinder nebo uváděné v knihách o setových tancích. Většinu testerů mátly šipky mezi jednotlivými tanečními kroky. Nebyli si jistí zda se májí kroky tančit zároveň souběžně, čí zda opravdu postupně za sebou tak jako čteme psaný text. Často jsem zaznamenal také požadavek na možnost otevření podrobnějšího vysvětlení kokrétního tanečního kroku.

.

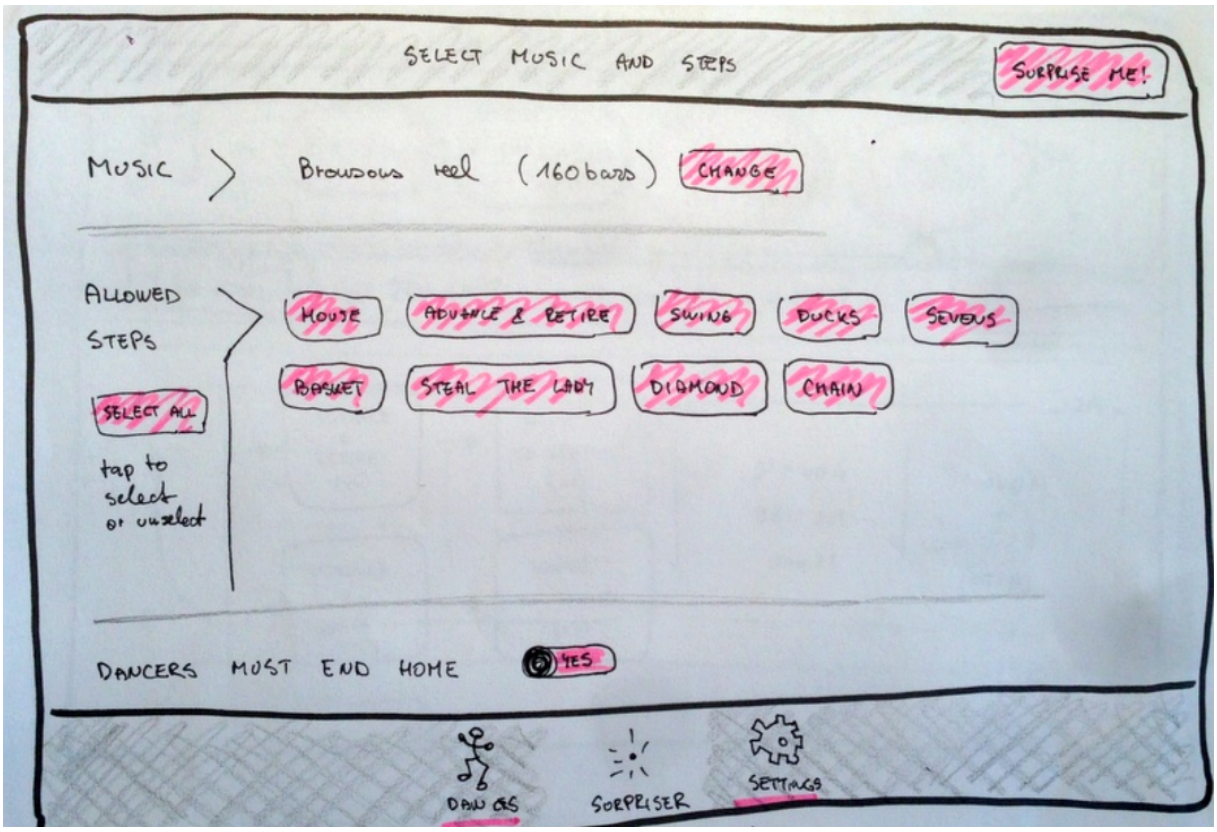


4.1.4.6 Generátor tanců

Popis Umožní uživateli vybrat hudbu a množinu tanečních kroků ze které se má tanec generovat.

Zpětná vazba testování Obrazovku pochopila správně přibližně pouze polovina testerů. Obrazovce rozuměli téměř výhradně testeři, kteří jsou často v roli učitele a potýkají se s problémy, které má tato obrazovka ambice řešit. Potenciál pro učitele byl poutavý a lákavý k vyzkoušení. Shodovali se však, že tato část pro není prioritní. Možná by bylo vhodnější tuto funkcionalitu oddělit do samostatné aplikace. Tři testeři by uvítali možnost vytištění vygenerovaného tance - tento požadavek nepovažuji za prioritní protože u zbývajících uživatelů nikdy nevyvstal.

.



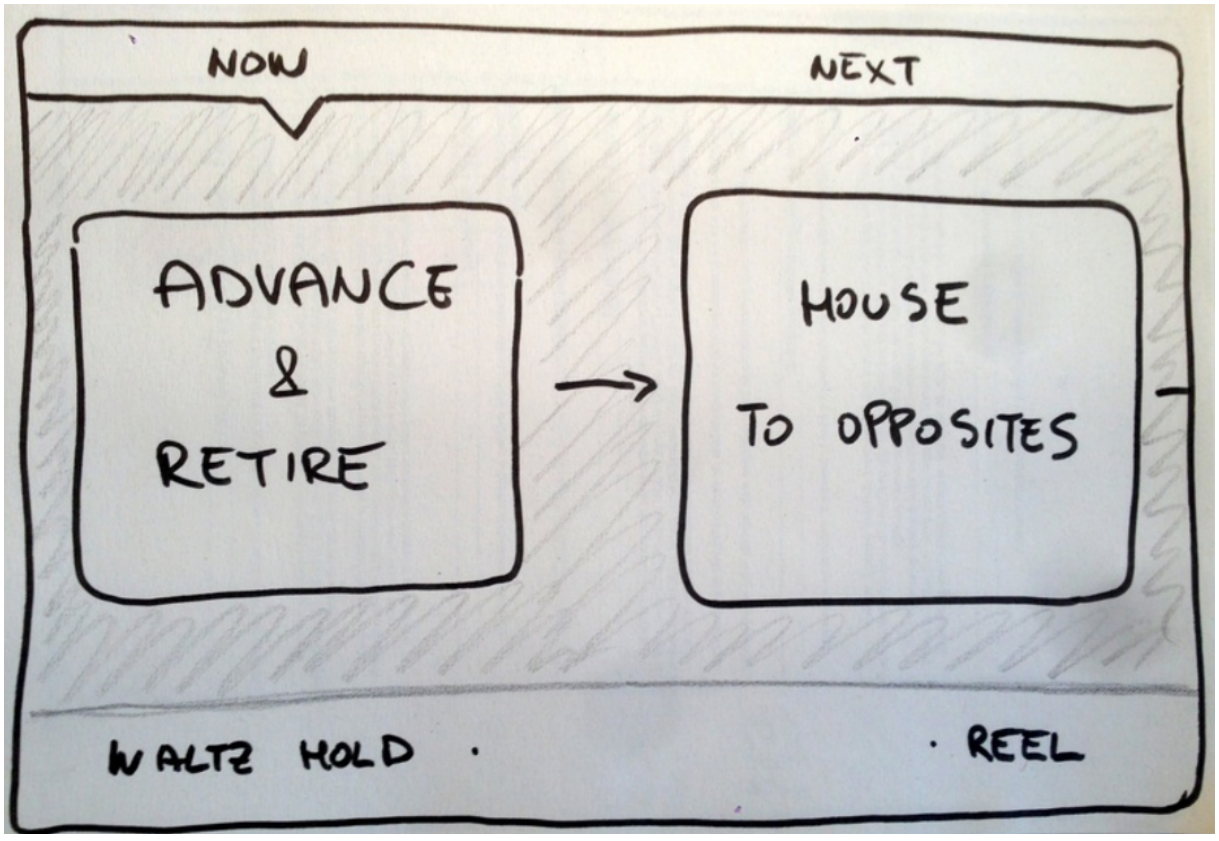
Obrázek 4.5: Prototyp obrazovky pro generátor tanců

4.1.4.7 Vizuální nápověda při volání

Popis Znázorňuje aktuální a blížící se krok včetně rytmu a držení rukou.

Zpětná vazba testování Obrazovka byla testerům jasná. Žádali však o možnost pozastavit a opětovně spustit hudbu i napovídání - toto by bylo určitě rozumné přidat. Dále z uživatelského testování vyplynulo, že by se uživatelům líbilo pokud by jim stroj před hlášením prvního kroku tance ještě navíc odpočítal počáteční hudební doby, na což jsou zvyklí od svého učitele. Jde prakticky o jednoduché počítání od jedničky do osmičky, tanečníkům to však dovolí se připravit na start tance. Během testování v zahraničí se staršími uživateli jsem zjistil, že kontrast barev a tučnost písma je nedostačující, proto jsem se v úpravách uživatelského zaměřil také na výběr jasnějších a kontrastnějších barev. Tento text by měl být maximálně čitelný pro co nejširší spektrum tanečníků.

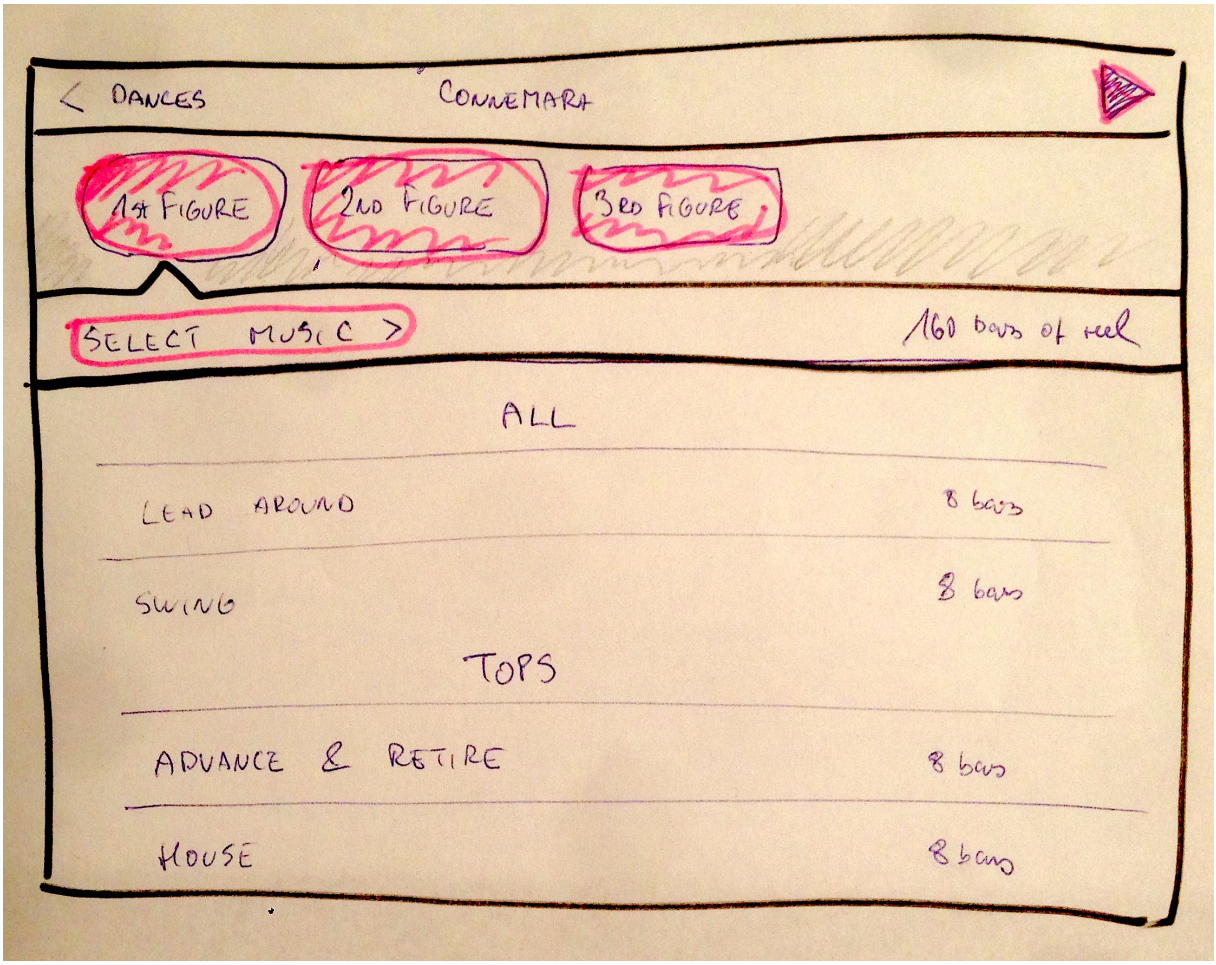
.



Obrázek 4.6: Prototyp obrazovky pro zobrazení vizuální nápovědy při volání

4.1.5 Vylepšený high fidelity prototyp

Díky testování prvního prototypu jsem získal zpětnou vazbu od testerů a byl nucen vymyslet zejména obrazovku s detailem figury jinak, lépe, tak aby byla pro uživatele srozumitelnější. Nahradil jsem diagramový popis tance zápisem tance zavedený v knížkách o setových tancích. Rovněž jsem rovnou vypustil problematickou obrazovku s výběrem hudby pro všechny figury najednou a upravil tlačítko pro spuštění napovídání kroků tak aby se spustilo pouze pro vybranou figury (automaticky nebude nadále pokračovat další figurou).



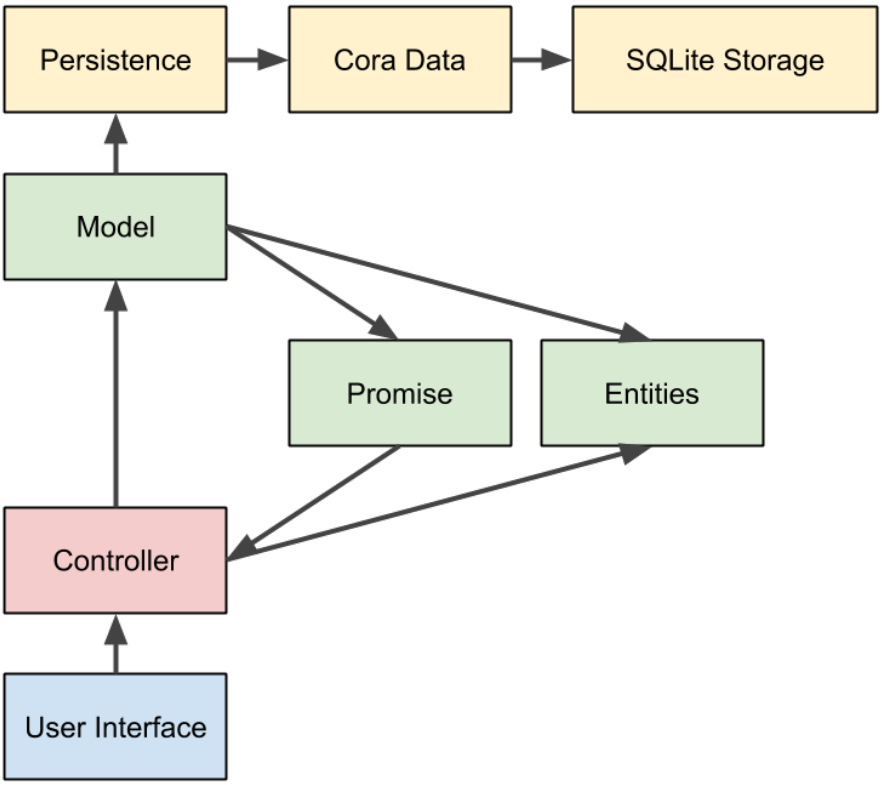
Obrázek 4.7: Vylepšený high fidelity prototyp

4.1.6 Shrnutí

Testování vyzdvihlo jako nejpodstatnější část obrazovku s se seznamem kroků pro aktuální figuru, dále pak obrazovku se stavem analýzy importovaných skladeb a obrazovku s právě volaným krokem.

Pro finální implementaci vynechám sekci pro generování tance, protože není pro cíl aplikace klíčová a algoritmus jsem ověřil v prototypu napsaném v Javě (viz příloha složka java-generovani-tance). Dále vynechám funkci pro zpomalení hudby během hlášení kroků, protože ji uživatelé považují za zbytečnou - přesto by nebylo problém zpomalení hudby doimplementovat pomocí knihovny SoundTouch (viz sekce Analýza).

4.2 Architektura



Obrázek 4.8: Diagram důležitých základních komponent aplikace

Architektura aplikace je rozdělena podle principu Model-View-Controller do několika částí, kdy každá část zodpovídá za určitou povinnost a snaží se minimálně záviset na ostatních vrstvách.

Uživatelské rozhraní je vyvářeno prostřednictvím Controlleru (řídícího kódu), který rozhoduje o tom co se do rozhraní naplní, co se kdy zobrazí a jak se bude reagovat na interakci uživatele. Controller je poslední výstupní bod aplikace proto na této vrstvě probíhá vytváření instancí závislostí pro objekty zejména modelu.

Model je navržen s ohledem na Dependency injection, tedy že všechny objekty na kterých modelové třídy závisejí jsou deklarovány v konstruktoru modelové třídy a není možné vytvořit modelovou třídu aniž by ji nebyly předány závislosti. Díky ctění dependency injection

principu jsou tyto třídy mnohem lépe testovatelné a kód je přehlednější protože vnějšímu kódu jasně říká co vyžaduje ke svému fungování.

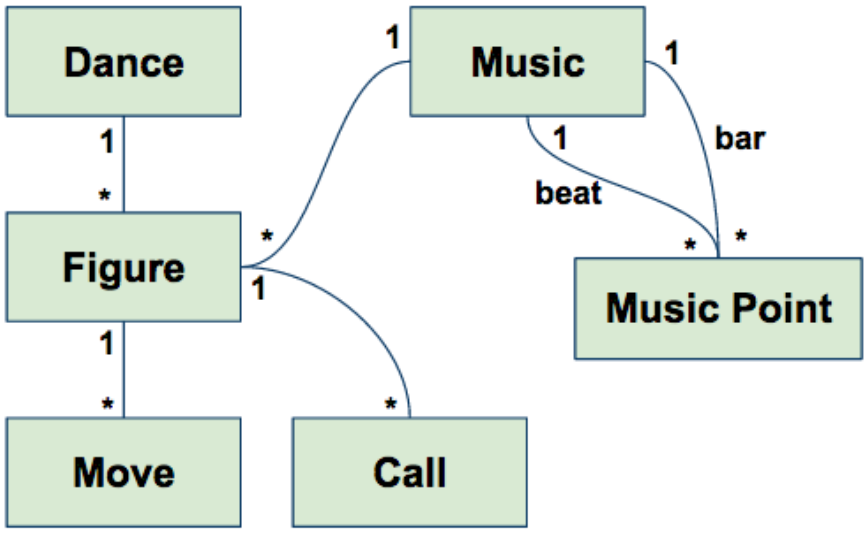
Pro podporu asynchronního chování aplikace jsem zavedl do architektury také pojem Promise, jejíž objekty hrají důležitou roli při komunikaci mezi modelem a controllerem. Vždy když controller žádá model o nějakou službu, je controlleru vrácen okamžitě objekt Promise, díky kterému si může controller definovat kód pro případ úspěchu a selhání modelové metody. Průběh modelové metody zabalené v Promise je spuštěn na jiném vlákně proto jeji řešení neblokuje hlavní vlákno starající se o vykreslovaní uživatelského rozhraní.

Entities označují obyčejné čisté objekty, které slouží pouze k transportu dat mezi modelovými třídami a controllerem, případně mezi jednotlivými modelovými třídami. Nemají žádné závislosti, můžou být předávány bezpečně mezi vlákny.

Persistence označuje vrstu aplikace starající se o ukládání dat do databáze prostřednictvím iOS služby Core Data, která efektivně udržuje graf objektů a ukládá jej do SQLite databáze. Graf objektů v Core Data pracuje s objekty nazývanými ManagedClass, které jsou protějšky Entit v modelu, avšak nesmí být předávány mezi jednotlivými vlákny. Ukládání dat probíhá na vlákně v pozadí.

4.3 Model domény

4.3.1 Datový model



Obrázek 4.9: Diagram vyztahů entit

Datový model popisuje základní prvky cílové domény irských setových tanců. Základní entitou je Dance, reprezentující tanec. K tanci lze přiřadit vždy více figur. Ke každé entitě Figure pak lze asociovat uspořádané taneční kroky (entita Move). K figuře lze také přiřadit

uspořádané objekty entity Call, která posuje věty a čas nápovědy k tanci vycházející z podrobnějšího popisu uloženého jako posloupnost kroků u figury. Další velice důležitou entitou je Music, která popisuje uloženou hudební nahrávku a všechny k ní identifikované hudební doby a takty.

4.3.2 Detekce hudebních dob

Objekt pro detekci hudebních dob bude fungovat jednorázově pro jednotlivou skladbu. Pokud bude potřeba analyzovat další skladbu, pak se vytvoří nový objekt. Během detekce je alokováno veliké množství proměnných a je dobré je pravidelně čistit, když už nejsou potřeba.

Konstruktor Metoda bude očekávat cestu k souboru, ve kterém má detekovat hudební doby. Budou k dispozici také konstruktory, které umožní zvolit jinou velikost okna pro zpracování signálu a upravit množinu hraničních frekvencí pro fázi filterbank.

Find Beats Metoda zahájí hledání hudebních dob ve skladbě a dovolí definovat anonymní funkce prostřednictvím, které dostane odpověď se všemi nalezenými dobami a informací o průměrné rychlosti BPM (beats per minute)

Progress Objekt umožní definovat anonymní funkci prostřednictvím, které bude model hlásit jeho aktuální průběh s analýzou. Myšlenka je taková, že průběh probublá až do uživatelského rozhraní, kde se ukáže uživateli jako zbývající množství práce.

4.3.3 Dance book

Objekt bude sloužit pro správu tanců, nastavení figur, výchozích skladeb pro figury a dalších souvisejících věcí. Veřejné metody budou vrace t výsledek prostřednictvím Promises.

Konstruktor Bude očekávat objekt NSManagedObjectContext potřebný pro práci s databází. Na veškeré ukládání do databáze budou metody čekat, to však nevadí protože poběží na vlákně v pozadí a pro controller jsou zabaleny v Promise.

Count all dances Spočítá veškeré existující tance

All dances Vrátí všechny tance, ovšem vrátí jejich datově mělkou variantu, tedy bez figur a tanečních kroků. Pro potřeby výpisu názvu tanců tato data stačí.

Import dances from JSON Zajistí přečtení JSON zápisu tance, vytvoření odpovídajícího grafu tanečních objektů a jeho uložení do databáze.

Detail of Dance Vrátí pro zadaný tanec jeho podrobnější návod, tedy jeho figury včetně tanečních kroků.

Figure save default music Uloží do databáze vybranou hudbu k zadané figuře. Tato hudba bude pak automaticky vybrána při příštím použití figury.

4.3.4 Music Library

Objekt bude sloužit pro veškerou správu hudby. Dovede importovat hudbu z knihovny iPod do aplikace, hledat požadované uložené skladby, také bude fungovat jako prostředník k objektu pro detekci dob. Konstruktor bude opět požadovat instanci kontextu pro persistenci.

List music Vypíše seznam všech importovaných skladeb. Bude schopen výpis filtrovat dále podle rytmu, počtu nalezených taktů případně umožní hledat v názvu.

Compute bars Pro zadanou entitu Music bude metoda umět spočítat počet taktů. Podkladem pro výpočet budou dostupná měření čas hudebních dob a informace o rytmu.

Save music Uloží veškeré změny v zadaném objektu skladby do persitentního úložiště.

Detect beats and bars Metoda tvoří fasádu nad detekcí dob a výpočtem taktů a usnadňuje tak použití detekčního algoritmu.

Import media item from iPod Zajistí uložení audio souboru do adresáře aplikace díky předanému popisu iPod reprezentace skladby

Path for music Vrátí cestu k uloženému audio souboru.

4.3.5 Caller

Objekt bude zajišťovat zejména vygenerování posloupnosti vět k hlášení v určitém čase pro zadanou figuru.

Calls for figure Metoda vyvoří seznam vět vhodných k volání figury. Zajistí aby pokud tančí stejný pár vícekrát za sebou, aby nebylo hlášené jméno páru zbytečně mnohokrát. Stačí jednou.

4.4 Řídící kód

Pro organizování a směrování logiky aplikace je důležitá vrstva řídícího kódu (controllerů). Představuje spojovací článek mezi byznys logikou a grafickým uživatelským rozhraním aplikace. V iOS mobilních aplikacích jsou zastoupeny třídou AppDelegate a třídami odvozenými z UIViewController.

Dependency injection Vrsta controllerů tvoří nejvyšší vrstvu, proto jsou zde tvořeny instance závislostí pro objekty vyžadující dependency injection. Jsou to zejména objekty modelu. Tím, že se instance vytvářejí na nejvyšší vrstvě je snadné pro jiné vývojáře nalézet souvislosti pro třídu a pracovat dále s nimi.

Pro doplňování závislostí jsem využil ještě třídu AppDelegate, která je první instancí spouštěnou hned po startu aplikace. Každá iOS aplikace vždy prochází nejprve touto třídou a pak pokračuje zobrazením hlavní obrazovky. Právě proto, že jde o první instanci, je věkem nejstarší a tedy je ideálním místem, kam umístit kontejner pro vytvářené znovu použitelné instance objektů. Do třídy AppDelegate jsem umístil jako veřejně dostupný například objekt NSManagedObjectContext sloužíci k ovládání databáze na pozadí. Instanci třídy AppDelegate lze snadno získat ve kterémkoli controlleru a lze tedy odsud jednoduše doplňovat sdílené instance objektů.

Dances Controller Controller nejprve zjistí zda jsou v databázi uloženy nějaké tance. Pokud jsou pak je vypíše do seznamu. Pokud nejsou, pak provede import vestavěného JSON souboru se zápisem výchozích tanců. Jakmile je import dokončen zobrazí všechny uložené tance. Tanec je možné rozkliknout do detailu. Při otevírání obrazovky detailu, je této obrazovce předán objekt Dance, aby následující obrazovka věděla k čemu detail zobrazit.

Dance Detail Controller Ihned po obdržení objektu Dance si controller načte detailní strukturu tance včetně figur a tanečních kroků. Jakmile je detail znám, zobrazí první figuru a její kroky včetně nezbytného rytmu a potřebného počtu taktů. Z tohoto controlleru je možné pokračovat na controller pro výběr hudby nebo na controller pro volání tanečních kroků.

Caller Controller Controller po obdržení figury, kterou má hlásit si pro figuru nechá prostřednictvím modelu sestavit posloupnost volaných vět. Dále při vytváření objektu hudby připraví časovač, který v příslušný čas nechá syntetizátor hlasu zavolat větu. Hudbu je možné zastavit a znova spustit.

Music Picker Controller Controller zobrazí dostupné analyzované skladby a po klinutí na některou z nich ji vybere a prostřednictvím modelu uloží do databáze. Následně se vrátí na původní controller. Controller dovoluje také importovat nové sladby z iPod knihovny, po importu je automaticky spuštěn controller pro analýzu skladeb.

Music Analysis Controller Controller po obdržení skladeb k analýza začne okamžitě detekovat prostřednictívm modelu beaty a takty. Uživateli pak dovolí vybrat rytmus, při výběru rytmu přehraje automaticky usek skladby aby uživateli usnadnil poznání rytmu.

4.5 Nasazení

Prvním krokem je založení vývojářského účtu na developer.apple.com. Základní Apple Developer program je plně zdarma, ovšem abychom byli schopni aplikaci publikovat v App Store je nezbytné zakoupit placený program iOS Developer. Pro registrované vývojáře je

k dispozici iOS Developer centrum, které pod sebou združuje veškeré služby a portály potřebné pro správu vyvíjených aplikací, testovacích zařízení a poskytují přístup ke studijním materiálům.

Nejen před samotným publikováním, ale nejlépe už během vymýšlení nápadu aplikace je dobré seznámit se s pravidly a omezeními, která se Apple klade na aplikace publikované v App Store. Apple rozhodně odmítá aplikace, které během testování padají, které používají neveřejná Apple API, přesně replikují funkce existujících nativních aplikací. Aplikace nesmí používat mikrofon nebo fotoaparat bez vědomí uživatele. Je dovoleno používat In-App Purchase pouze pro placení statků uvnitř aplikace.

Pro novou aplikaci je nutné v iOS Provisioning Portal založit nový identifikátor App ID. Identifikátor může být přesný explicitní nebo volnější wildcard. Wildcard identifikátor se používá třeb během vývoje aplikací. Explicitní identifikátor je nutný pokud aplikace využívá iCloud, In App Purschase nebo Push Notification. Dále je potřeba vytvořit podle návodu distribuční certifikát, kterým si vývojář prokáže identitu během odíslání aplace do AppStore. Poslední nutností je provisioning profil, který v sobě nese seznam povolených zařízení k testování, případně obahuje kličový popis aplikace.

Popis, metadata, ikona a screenshoty pro novou aplikaci se vkládájí do portálu iTunes Connect. Nahrávaná grafika má obvykle určité předepsnané minimální rozměry. Je potřeba přibližně pět různých velikostí ikony. Screenshotů aplikace může být maximálně pět, proto je důležité se dobře zamyslet co je podstatné zobrazit. Metada obsahuí verzi a číslo sestavení (build number) aplikace, díky, které pak koncoví uživatelé vědí, zda je k dispozici novější verze. V této části se rovněž nastavuje cenová hranice aplikace a časová strategie vypuštění na veřejnost. Nově lze pomocí podsekce v Testflight rovněž snadněji zahájit interní testování.

Nyní, kdy jsou veškeré administratitivní věci připravené, můžeme začít aplikaci nahrávat. Proces nahrání do App Storu a validaci zajišťuje XCode nebo Application Loaderu. Schválení obvykle trvá kolem jedno až dvou týdnů.

Kapitola 5

Realizace

5.1 Mobilní aplikace

5.1.1 Detekce hudebních dob

Byl jsem velmi překvapen jak obtížné je převést prototypový kód Matlabu do programovacího jazyku nižší úrovně jako je Objective C, které místy navíc využívá starší volání pro jazyk C.

Pro zpracování signálu je na platformě iOS nejlepší volbou framwork Accellerate od Apple, který disponuje širokou škálou funkcí optimalizovaných pro různé datové typy i různé potřeby. Dohromady s knihovnou cBLAS pro obecnější matemické funkce mi velmi usnadnily implementaci comb filter algoritmu.

Před samotným zpracováním hudby je nutné definovat formát dat, který budeme potřebovat. Zde jsem zvolil PCM formát pouze v jednom zvukovém kanálu (mono). Veškerou zpracovávanou hudbu do aplikace vždy nejprve zkopíruji, aby ji bylo možné snadněji zpracovávat. Navíc pokud uživatelé smažou hudbu z databáze, zůstane jim alespoň v naší aplikaci a budou ji moci stále využít.

Čtení proudu PCM vzorků provádím po úsecích daných oknem. Velikost okna je na začátku algoritmu zvolena a zarovnána tak aby měly funkce frameworku při zpracování co nejméně chyb. Po přečtení úseku jsou vzorky odeslány funkci pro analýzu. Zmíním, že čtení po blocí je prováděno tak aby se bloky asi ve 20% od konce překrývaly s následujícím blokem, tím je zajišťeno stálejší předpovídání detekovaných dob. Navíc je používání posouvacího okna pro zpracování takto velikého objemu nekomprimovaných dat také vhodnější z pohledu šetrnosti na využití paměť a procesoru mobilního zařízení.

Během analýzy se vždy nejprve zaměřím na provedení první fáze algoritmu nazvané filterbank, která rozdělí zvukový signál podle frekvenčních pásem do několika samostatných posloupností vzorků. Výsledek fáze filterbank v implementaci aplikace uchovává ve struktuře matice (vlastní definovanný objekt BankMatrix), který obsahuje signál rozdělený podle zadaných frekvenčních pásem. Tato vlastní definice matice disponuje metodami, které usnadňují čištění zabraného prostoru v paměti. Rovněž díky tomu, že jde o objekt, který obsahuje pásma v jednom místě je snadné jej celkově předávat dalším fázím algoritmu.

Další fází je vytvoření obálky pomocí polovičního Hannova okna, čímž se v signálu zvýrazní rychle rostoucí vlny (obvykle představují právě doby). Dále provedu spočítání difference a nakonec jednosměrné usměrnění (rectify) signálu, výsledkem je poztivní signál, kde jsou lépe poznatelné vrcholky dob.

Další fáze provede důležitou detekci tempa pomocí comb filteru. Tuto detekci provádím nad zjednodušenou množinou vzorků pro zrychlení algoritmu. Stejně tak nad stejnou množinou vzorků provedu určení fáze comb filteru. Díky tempu a fázi jsem nyní schopen určit časy jednotlivých dob. Proto je všechny vypíšu do společného pole.

Jakmile jsou nalezeny všechny časy dob příjde na řadu čištění dob. Nejprve odstraním doby které byly nedopatřím vytvořeny příliš těsně vedle sebe a představují tak vlastně jednu jedinou dobu. Dále vyčistím příliš tiché doby na začátku a na konci skladby. K tomu musím znova přečíst vzorky signálu a spočítat sumu amplitudy vzorků mezi jednotlivými dobami. Jakmile máme spočítáno naleznu průměrnou hlasitost prvních několika dob. Pokud některé doby na začátku jsou pod průměrnou hlasitostí, pak je vypustím.

Nyní jsou nalezeny všechny zajímavé doby a jsou vráceny modelem ven do aplikace.

5.1.2 Načítání hudby v iOS

Hudbu z iPod knihovny nelze zpracovávát ve formě PCM okamžitě. Přístup k hudebním souborům iPod knihovny je v iOS dovolen pouze komponentě Core Audio, která nabízí metody pro řízení přehrávání hudby. Pro zpracování signálu je nutné hudbu nejprve zkopírovat z iPod knihovny do adresáře aplikace. Pak ji lze načíst ve formě PCM a prostřednictvím nativní knihovny Accellerate zpracovávat.

5.1.3 Hlášení kroků

Pro hlášení kroků jsem zvolil vestavěný syntetizátor hlasu od Apple. Nástroj disponuje několika jazyky a mnohé další lze stáhnout dodatečně. Při testování syntetizátoru jsem narazil na problém s delší inicializací třídy. Což je nešikovné pokud by se to stale během přehrávání hudby a vzniklo by tak v hlášení nepatrné zpoždění. Nakonec jsem dospěl k menšímu triku kdy, těsně před spuštěním hudby nechám promluvit syntetizátor velmi rychle a ztišeně - to zatím stačilo pro inicializaci a pak prováděl syntetizátori hlášení téměř okamžitě. Možným rizikem je, že pokud bude syntetizátor příliš dlouho potichu, může preventivně dealokovat některé zdroje a při dalším hlášení tak může míž zpoždění. To by ovšem mělo opět byt možné vyřešit tichým hlášením těsně před potřebným hlasitým hlášením.

Alternativou by jinak bylo nahrání jednotlivých kroků a jejich souběžné přehrávání k hudbě. Tento způsob by dovolil pěknější hlas volajícího ovšem dosti by omezoval množinu hlášení, bylo by totiž nutné s každý novým krokem přidat audio nahrávku hlášení.

Před samotným spuštění prvního hlášení nápovědy je nutné a nezbytné provést naplánování času, kdy přesně zahlásit taneční krok. Z uživatelského testování jsem se dozvěděl, že by uživatelé velmi ocenili kdyby jim stroj před hlášením prvního kroku tance ještě navíc odpočítal hudební doby, na což jsou zvyklí od svého učitele. Jde prakticky o jednoduché počítání od jedničky do osmičky, tanečníkům to však dovolí se připravit na start tance. Během přípravy volání aplikace postupně rochází posloupnosti kroků figury a zapisuje si k jednotlivým krokům, kdy je zahlásit. Aplikaci si vlastně rovnou předpřipravuje věty které bude syntetizátor

hovořit. Rozhoduje zde také u každého kroku zda má cenu zahlásit který tanečník či taneční pár má tento krok začít tančit.

5.1.4 Promises

Promises mnoho logiky v controlleru výrazně zpřehlednily. Mé osobní rané zkušeností s programováním v iOS mi v některých chvílích přinesly neblahé zkušenosti s přílišným zanořováním anonymních funkcí (blocks) a vznik problému nazvývaného "callback hell". Callback hell je označení nesprávné praktiky softwarového inženýrství, kdy vývojář u složité logiky využívá anonymních funkcí pro zpracování části logiky, která se má provést například po dokončení jiné aktivity. Problém, který se vyjeví obvykle až s postupem časem, je příliš hluboce zanořená spousta anonymních funkcí, které nelze rozumně udržovat. Nešikovnou věcí je také nízká znovupoužitelnost takto vytvořené logiky, přitom potřeba znovu využít tuto logiku dříve nebo později určitě příjde.

Promises vývojáře motivují přesouvat důležitou opakující se logiku zpět do vrstvy Modelu, kam vlastně stejně patří. V modelu se kód zabalí do běžné funkce, která přijímá v argumentech informace nezbytné pro další vykonání logiky. Až doposud to vypadá jako běžná práce s modelovou třídou. Místo, kde se tato praktika liší je návratová hodnota funkce. Návratovou hodnotou bude právě takzvaná Promise, které lze rovnou upřesnit i charakter obsahu dat, který bude vracet dalšímu kódu. Touto definicí kód vlastně slibuje, že po zavolání se vrátí zpět s určitou informací a informuje uživatele, že se vrátí až kdy bude tuto informaci znát.

U promises lze tedy krásně využít vlastnosti zřetězení volání, kdy můžeme propojit existující modelové funkce za sebou podle konkrétní aktuální potřeby. Jde prakticky o naplánování kroků určitého procesu, který je nutné provést. Kód který toto zřetězení provádí dá signál k zahájení vykonávání řetězu. Řetěz událostí se obvykle vykonává na pozadí ve vedlejším procesu a umožňuje tedy uvolnit ruce například hlavnímu vláknu a předejít tak zasekávání uživatelského rozhraní. Výhodou, kterou nám takové to zřetězení přináší je úžasná vyjadřovací schopnost kódu, který je pak s časem snadněji udržovatelný.

Následující ukázka představuje jak je možné zřetězit několik promises do logiky zajišťující, že pokud nejsou načteny žádné tance, pak se automaticky importují a zobrazí až po dokončení importu. Výhodou je perfektní asynchronnost celé aktivity. Nejprve vidíme, že se vytvoří instance objektu DanceBook, která obstarává logiku týkající se kolem tanců. Vytvoření instance tohoto objektu vyžaduje předání dříve definovaného kontextu pro přístup do databáze (prostřednictvím vrstvy CoreData). Jakmile je instance vytvořena, můžeme využívat naplno její metody. Kód ukázky níže se vyskytuje v řídícím kódu jedné třídy a spouští se těsně před tím než je uživateli zobrazena obrazovka s výpisem tanců. Tento řídící kód tedy ví, že aby dovedl uživateli zobrazit něco rozumného musí si o tyto informace říct. Ovšem zatím, v čase, kdy ještě data nejsou dostupná, pak řídící kód zobrazí prázdnou tabulku. Kód tedy pomocí metody countAllDances() požádá o spočtení dostupných tanců v databázi a hned za toto volání zavěsí kód, který se má provést po úspěšném získání počtu dostupných tanců. Rovnou v tomto místě nadefinuje, že pokud tance jsou k dispozici pak pokračuje dalším voláním v řetězu promises. Pokud tance nejsou žádné, pak se do řetězu vloží ještě jeden uzel který bude muset zajistit import tanců ze zápisu tance do databáze aplikace. Zde je krásně vidět, že kód pro stažení dostupných allDances() tanců je volán pouze jednou. Pokud by se

kód zapisoval přes anonymní funkce a callbacky, musel by být nejméně na třech místech aby pokryl všechny možné situace, kdy některé procesy třeba selžou. Pomocí promises pokud by došlo k chybě přeruší se celý řetěz a vyhodí se vyjímka, kterou lze odchytit. Pro odchycení vyjímky lze použít volání "catch()", které se provede v případě, že se v řetězu vyskytne chyba. V příkladu tedy úspěšný řetěz pokračuje stažením tanců a následně spustí blok kódu, který informuje uživatelské rozhraní aby překreslilo tabulku a zobrazilo seznam tanců.

```
let appDelegate = UIApplication.sharedApplication().delegate as App
self.danceBook = DanceBook(context: appDelegate.backgroundContext!)
self.danceBook.countAllDances().then(body:{ numberOfDances -> Prom
    if (numberOfDances > 0) {
        return nil
    }
    return self.danceBook.importDancesFromJson(NSString(
        contentsOfFile: NSBundle.mainBundle().pathForResource("danc
        encoding: NSUTF8StringEncoding,
        error: nil
    )!)
}).then(body: {Void -> Promise<[Dance]> in
    return self.danceBook.allDances()
}).then(body: {dances -> Void in
    self.dances = dances
    self.dancesTableView.reloadData()
});
```

Obrázek 5.1: Ukázka zpracování logiky pomocí bloků

5.1.5 Model

Chování modelu bylo téměř kompletně popsáno v části s návrhem aplikace, přesto zde ještě vypíchnu zajímavosti týkající se realizace modelové vrstvy. Podařilo se mi zvolit jasné pojmenování většiny metod a je tedy snadné poznat už z názvu co metoda provádí uvnitř. Pro zápis jsem zvolil jazyk Swift a organizoval jej do složky Model v XCode projektu. Modelové třídy používají pro své fungování zejména výlučně třídy ve složkách Persistence, Entities a Music Analysis. Závislosti jednotlivých tříd jsou deklarovány vždy v konstruktoru třídy a je tedy snadné při vytváření nové instance najít a doplnit požadované závislosti nezbytné pro běh třídy. Většina modelových metod je ohleduplná a pokud je pravděpodobné, že zajištění odpovědi potrvá delší čas, je definována metoda vždy s návratovou hodnotou slíbenou pomocí návrhnového vzoru promises.

Instance modelových tříd vytvářím téměř výhradně ve vrstvě řídícího kódu (Controller). Případně některé modelové objekty vznikly složením z jiných modelových tříd. Jediné objekty dostupné napříč aplikací jsou datové entity, které slouží k přehlednému a ucelenému přenosu informací napříč systémem. Nejzajímavějšími jsou třídy DanceBook, MusicLibrary a Caller.

DanceBook je určně jak název napovídá k práci s tanci, zejména k získávání informací o tancích. Třída dovede zjistit zda nějaké tance uchovává, kolik jich uchovává nebo vrátit kompletní podobu. Dovede importovat tance ze speciálního JSON zápisu, kdy zápis transformuje do struktury která je vhodnější pro účel použití dat v aplikaci. JSON taneční notace totiž

například úsporně pracuje s definicemi kroků a dovoluje psát zapisovateli stručně, ale přitom dostatečně popisně. Pro zpracování v aplikaci je například lepší posloupnost kroků pro figuru převést do ploché struktury, kde se sice ztratí informace o dříve přetížených objektech, ale přesto je dostatečné, že krok obsahuje veškere dostupné informace v jednom obektu, třeba, že informace v posloupnosti duplikuje. Nástroje pro hlášení pak mohou s posloupností snadněji pracovat a připravovat ji po napovídání.

MusicLibrary je třída obstarávající veškerou práci s hudbou v aplikaci. Vrací výpis dostupné hudby, vrací kompletní graf záznamu hudby, dovede spočítat takty z dodaných hudebních dob a rytmu, zvládá ukládání grafu zaznamenávající podrobnosti hudby, dovede importovat do aplikace audio soubor z iPod knihovny, zvládne u hudby odvodit její rytmus na základě dostupných meta informací. Pro zbytek aplikace obsahuje také fasádu usnadňující žádost o detekování hudebních dob ve skladbě.

Caller je model soustředící se na sestavení ideálního rozvrhu nápovědy pro danou figuru a hudbu. Funguje tak, že postupně prochází jednotlivé takty hudby, a počítá kolik taktů právě uplynulo a vždy když myslí, že je vhodné nápovědu zahlásit vytvoří objekt Call s přesným časem a větou k zahlášení. Posloupnost tanenčích kroků považuje v tuto chvíli za frontu, kterou odbavuje metodou FIFO (první dovnitř, první ven). Odebere tedy první krok posloupnosti a vymyslí vhodnou větu k hlášení. Podle délky věty určí s jak velikým předstihem je potřeba aktuální větu hlásit. Tento předstih odečte od počtu taktů potřebných k zatančení tohoto kroku. Pak pokračuje v počítání taktů hudby dokud nenarazí na takt, kdy je potřeba větu hlásit. V tuto chvíli vytvoří objekt Call s časem a větou k hlášení. Následně odeber z fronty další krok, počká až se dotancuje poslední krok a opět opakje proces s vymyšlením věty a určením času hlášení. S ohledem na zpětnou vazbu během uživatelského testování jsem navíc přidal, aby model na začátku skladby vždy odpočítál osm taktů a potom začal s hlášením tanečních kroků.

5.1.6 Persistence

Pro uchovávání dat v aplikaci využívám vestavěného proprietarního Apple frameworku Core Data, který slouží k ukládání grafu objektů do persistentního úložiště. Nejde v pravém slova smyslu o objektově-relační mapovací framework, přestože je mu velmi podobný a přestože skutečně data nejčastěji ukládá do databázových tabulek. Nemělo by se naň nahlížet jako na databázi, ale spíše jako na graf propojených dat aplikace.

Core Data model je tvořen dvěmi částmi

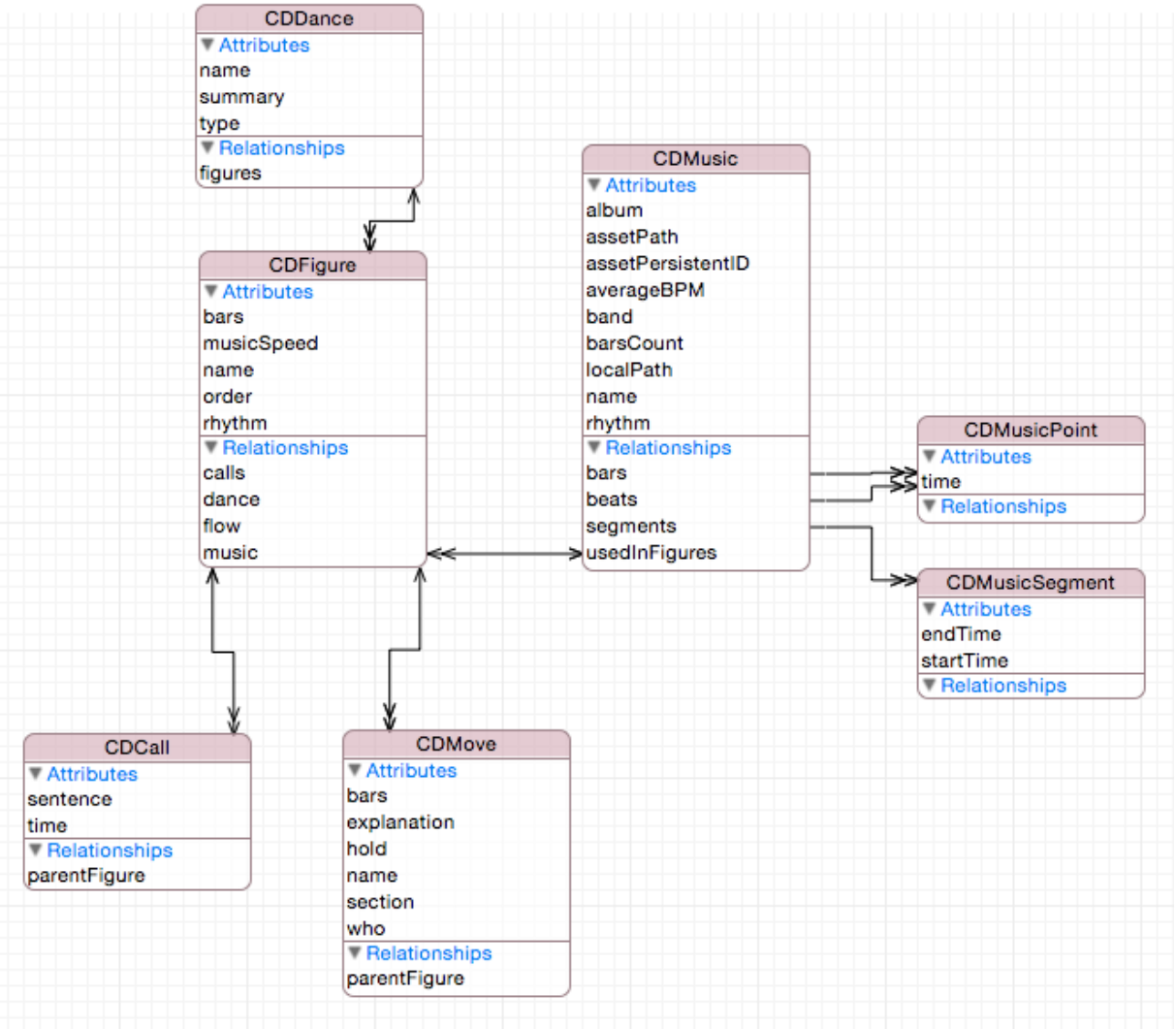
- managed object model ... popisuje schéma vlastností entit a jejich vztahů
- Core Data stack ... uskupení podpůrných objektů (může jich být v aplikaci podle potřeby i větší počet)
  - Persistent Store ... uložiště (může být cache, soubor nebo SQLite databáze)
  - Persistent Store Coordinator ... lze si jej představit jako spojení k databázi, dovoluje k sobě párovat pouze jeden managed ojbect model a jeden persistent store

– Managed Object Context ... lze si jej představit jako pískoviště, kde můžeme vytvářet a průběžně připravovat vztahy a až bude vhodné je uchovat, požádáme kontext o uložení. Splňuje náležitosti běžného transakčního chování, tedy lze uložit všechny změny nebo žádné, a lze všechny předpřipravené úpravy odvolat. Dovoluje k sobě párovat pouze jeden persistent store coordinator.

Prvním krokem, který jsem musel vykonat bylo navrhnutí schématu popisující obsah jednotlivých entit a vztahy, které mezi sebou mají a které by se měly projevit také v aplikaci. K tomuto účelu se v iOS používá vývojářský nástroj XCode umožňující pohodlně vytvořit a editovat soubor *.xcdatamodel (managed object model), kde lze upřesnit vše potřebné k persistenci dat. Definovat entity, vlastnosti entit včetně jejich typu, upřesnit nulovost vybraných položek, dát najevo zda je potřeba indexovat, zda má minimální maximální hodnotu, zda má nějakou výchozí hodnotu, zda k entitě může přistoupit operační systém prostřednictvím vyhledávání. Zejména pak také definovat vztahy mezi entitami, kde lze upřesnit jak jejich směr, vlastníka a kardinalitu vztahu, zda jde o uspořádaný vztah nebo o množinu, volitelnost či pravidlo, které se má aplikovat při smazání (vynulování referencí, kaskádovité smazání). Zapisovat tyto informace lze graficky pomocí jednoduchého editoru. Osobně však preferuji ruční zápis informací pomocí formuláře, kde lze rovnou nastavit všechny potřebné vlastnosti. Z takto definovaného schematu je možné vygenerovat také třídy pro všechny potřebné entity. Tyto entity dědí ze speciální třídy NSManagedObject a usnadňují přístup ke Core Data schematu.

Hlavními datovým objektem je CDDance reprezentující tanec, ke kterému jsou navázány související figury (CDFigure) včetně jejich tanečních kroků (CDMove). K jednotlivým figurám schema dovoluje uložit také například naposledy použitou hudbu (CDMusic). Objekty CDMusic nemají žádné pevné a silně závislé vztahy na objekty z taneční domény a může tedy existovat relativně nezávisle, tento objekt reprezentuje importovanou hudební skladbu a uchovává k ní rovněž všechny rozpoznané hudební doby. Schema je připraveno zaznamenávat také podobné části hudby (pomocí entit CDMusicPoint a CDMusicSegment), tyto entity však v aktuální verzi aplikace zatím nenašly větší využití, počítá se s nimi spíše pro budoucí vývoj. Posledním zajímavou entitou je CDCall, který je připraven v sobě držet určitou sestavenou větu a informaci o čase. CDCall využívá modelová třída pro včasnou přípravu nápověd k tanci.

Důležitou vlastností ryzích Core Data entit je jejich závislost na vlákně, ve kterém byli vytvořeny. Tato skutečnost není na první pohled úplně zřejmá, ale je velmi důležité ji pochopit už na začátku vývoje. Pokud tedy vytvořím entitu na hlavním vlákně, můžu z ní číst všechny její vlastnosti a vztahy, prakticky okamžitě (Core Data na pozadí provede v případě potřeby dotaz do úložiště a počká na odpověď). Ovšem pokud bych tento objekt předal nějakému procesu běžícímu v pozadí na některém vedlejším vlákně pak by tento proces velmi pravďepodobně spadl při pokusu o přistoupení k vlastnosti či vztahu objektu této entity. Zde je problémem situace, kdy je nutné entity pro použití například v Controlleru aplikace vytvářet na hlavním vlákně, čímž můžeme při větším množství entit znatelně zaseknout uživatelské rozhraní. Tomu se snažíme při vývoji mobilních aplikací vyhnout. Jedním možným řešením je používat relativně složitou strukturu několika propojených instancí managed object context, z nichž některé budou vyhrazené pro vedlejší vlákno a jiné pro hlavní vlákno. Toto řešení pak bude mezi vlákny předávat jedinou bezpečnou a rychlou informaci, a tou je objekt NSManagedObjectID, tedy identifikátor entity - pomocí něj si pak, každé



Obrázek 5.2: Diagram entit Core Data schematu

vlákno (ať už je v pozadí nebo v popředí) příslušnou použitelnou verzi entity, kterou může používat bez rizika pádu aplikace. Druhým řešením je vytvořit obyčejné jednoduché čisté Swift nebo Objective C objekty, které budou obsahovat stejné vlastnosti jako entity ve schematu Core Data, protože nebudou dědit z NSManagedObject budou bezpečné pro předávání napříč vlákny. Toto řešení pak bude obvykle na pozadí stahovat z Core Data úložiště bezpečně přistoupitelné entity a jejich obsah bude exportovat do vlastních thread safe entit. Nevýhodou druhé způsobu je nutnost udržovat tuto mezivrstu tříd. Osobně preferuji druhý způsob i za cenu nutnosti udržování mezivrstvy, protože dovolí se vyhnout nepředvídatelným problémům, kdy omylem přistoupíme k objektu vyvořeném v jiném vlákně.

Core data stack obsahuje také podpůrné mechanismy pro ulehčení procesu migrace dat a změn ve struktuře databáze. Každou entitu lze například verzovat. V kódu pak lze například zajistit zpětnou kompatibilitu starších struktur zároveň při použití nového schematu.

5.1.7 Swift

Během realizace jsem se rozhodl využít nového jazyka Swift pro implementaci modelových tříd a controllerů. Zaběhnutý jazyk Objctive-C jsem znal již dříve a chtěl jsem poznat něco nového, právě se rodícího. Syntaxe je poměrně odlišná od Objective-C a vývoj aplikace vyžadoval více času než jsem dříve plánoval. Některé nezvyklé konstrukce jsem často konzultoval s manuálem a příručkami na internetu. Jazyk je stále ještě živý a průběžně se vylepšuje. Občas během vývoje jsem se setkal s nutností přepsat část kódu pouze z důvodu aktualizace vývojového prostředí, které v nové verzi disponovalo novější verzí jazyka Swift, který přestal podporovat některé dříve používané způsoby zápisu. Nešlo o příliš komplikované změny a úpravy, ovšem může to vývojáře dosti překvapit pokud si vyhradí čas pouze na drobné úpravy a ve skutečnosti je nucen opravovat ještě probémy související s aktualizací jazyka.

Poměrně dost času jsem ze začátku strávil pokusy zapsat v jazyce Swift některé příklady používající knihovnu Accelerate (vDSP knihovna pro zpracování signálů). Některé portované funkce jazyka Swift ještě nejsou plně připravené a vyladěné, proto jsem se často setkával s problémy spustit a zapsat vzorce a výpočty potřebné pro detekci hudebních dob. Nakonec jsem se rozhodl vrátit se v této části k původnímu stabilnímu jazyku Objective-C kombinovanému se starším jazykem C, ve kterém bylo možné experimety provádět bez větších obtíží.

Přesto zápis kódu mnohem stručnější a také čitelnější. Zmizely hranaté závorky, které některé příkazy jazyka Objective-C velmi znepřehledňovaly. Přibylo rozlišení typů v položkách kolekcí (podobně jako Generics v Javě). Redukoval se i počet nezbytných souborů pro popis třídy - v jazyce Swift se totiž již nepoužívají hlavičkové soubory, které v Objective C dříve definovaly veřejné rozhraní objektu. Nyní lze třídu definovat pomocí jediného souboru. Z této novinky mám radost, protože bylo občas frustrující nadbytečně popisovat stejnou metodu na dvou místech zároveň, únavné zejména při refaktorování.

V jazyce mě zaujaly nová klíčová slova umožňující snadno a přehledně definovat zda jde o konstantu nebo proměnnou. Klíčové slovo "let"slouží k definici konstanty, a klíčové slovo "var"slouží zase k definici proměnné. Ideální a zajímavé je využití klíčového slova let přímo podmínkách kde můžeme proměnnou definovat jen v případě že je výraz který do ni přiřazujeme nenulový. Tím pádem je pak proměnná přístupná uvnitř bloku if jen když

je tato podmínka splněná. Tento způsob zápisu vede k přehlednějšímu větvení kódu. Jazyk nevyžaduje ve všech případech upřesnění typu proměnné, protože je schopen typ odvodit z kontextu použití kódu, případně během runtime. Nejde však o beztypový jazyk. Typy jsou kontrolovány přísně, ovšem není nutné je psát tam kde je typ jasný z kontextu.

Zajímavou novinkou je také odlišení a explicitní zmínění u každé proměnné, zda může být volitelná, tedy že může obsahovat nulovou hodnotu. Neopak je možné také označit, zda proměnná nesmí být nulová a tedy, že je nutné před jejím použitím zkontrolovat zda obsahuje hodnotu, kterou chceme používat. Tyto skutečnosti se označují pomocí znaku vykřičníku (nenulový) a pomocí znaku otazníku (volitelný) za jménem typu proměnné. Ovšem pro začátečníka v jazyce Swift je místy komplikované odhalit správný způsob jak tyto znaky používat ke svému prospěchu. Naštěstí vývojové prostředí často dovede napovědět, kdy správně použít kterou značku.

Oblíbenou novinkou je také expresivnější přetypovávání proměnných z jejich obecně v kontextu známého typu na typ specifičtější pokud jej potřebujeme použít. Tento princip je perfektní například pro přetypování nově automaticky vytvořené buňky tabulky podle reuse identifikátoru. Ve storyboardu, kde navrhujeme uživatelské rozhraní totiž určujeme často také třídu kterou má buňka použít jako svou základní. Nevýhodou je že starší funkce pro generování buněk tento objekt vrací v obecnější typu a je nutné jej přetypovat na specifičtější, například ten který jsme definovali ve storyboardu.

Spokojený jsem byl také s používání testovacího frameworku, který přehledně informuje o úspěšných a neúspěšných testech. Dokonce nově umožňuje testovat také asynchronně prováděný kód. Toto jsem velmi ocenil právě, protože jsem v celé aplikaci se snažil maximálně používat asynchronní volání abych zrychlil responzivitu uživatelského rozhraní. Lze totiž v testovacím případu definovat identifikátor, který dokud někde v kódu neoznačím jako splněný tak metoda bude čekat na její splnění. Lze rovněž tomuto čekajícímu konstruktu nastavit čas po kterém by měl vypršet a považovat nesplnění za chybu v testu.

5.1.8 XCode

Nástroj XCode pro vývoj iOS aplikací je plně připraven jak na jazyk Swift tak starší Objective C. Většina procesů automatizující generování tříd a souborů podporuje oba jazyky. Oba jazyky lze také kombinovat a využívat zároveň a do určité míry vzájemně do jazyků publikovat rozhraní tříd napsaných v druhém jazyce (pomocí bridging headers). Některé části kódu je nyní pohodlnější a stručnější psát v novém jazyce Swift. Rovnocenně lze vyvíjet zroveň i ve starším jazyce Objective-C. Prostředí XCode nabízí vývojáři nejen editor kódu a správce projektu. Integruje nástroj Interface builder, ve kterém lze pomocí grafického rozhraní kreslit vzhled obrazovek, jejich propojení a tok aplikací. Pro ladění obsahuje nástroj Instruments pomocí, kterého lze sledovat vytížení procesoru, paměti, odhalovat rizika memory leaku, odhalit náročné vykreslovaní grafiky a mnoho dalších. Velmi praktickým je násroj iOS simulátor, díky kterému lze ladit a zkoušet aplikaci bez nutnosti vlastnit skutečné iOS zařízení. Simulátor věrně napodobí většinu běžného chování pravého zařízení včetně simulování GPS pozice uživatele. Omezením simulátoru je například v nemožnosti přístupu k iPod knihovně a nefungující syntetizátor hlasu - proto pro ladění aplikace mé diplomové práce jsem musel často kompilovat aplikaci přímo pro zařízení.

5.1.9 Cocoapods

Během vývoje jsem využíval některé užitečné knihovny třetích stran. Například Alamofire pro webovou komunikaci, PromiseKit pro využití návrhového vzoru promises a Chameleon pro generování ladících barev. Všechny tyto knihovny jsem instaloval pomocí nástroje Cocoapods[31], který slouží k zaznamenávání závislostí projektu na jiné projekty. Umožňuje související projekty stáhnout například ze serveru Github a připravit propojení s hlavním projektem. Jde o správce balíčků závislostí inspirovaný principem fungování podobného nástroje RubyGems+Bundler pro jazyk Ruby. Vyžaduje vytvoření souboru nazývaného Podfile, kam jsou vyčteny všechny potřebné závislosti včetně ideální verze. Pomocí jednoduchého příkazu lze zavislosti stáhnout, nainstalovat a propojit s projektem. Líbí se mi způsob jakým je doplnění závislostí vyřešeno - tedy, že se k mému hlavnímu projektu vytvoří přidružený dceřinný projekt nazvaný Pods, který v sobě automaticky obsahuje nakonfigurované a připravené knihovny, výhodou je že tento projekt se v XCode chytře zkompiluje pouze jednou pro cílovou platformu, a do mého projektu se vkládá už pouze vkládá jako připravený binární soubor a šetří tím čas pro kompilaci mého hlavního projektu.

5.2 Ověření realizace

Hotovou aplikaci jsem několikrát testoval sám osobně doma. Snažil jsem se důkladně kontrolovat zda hlášení odpovídají správným místům v hudbě, jestli se aplikace při používání v některých situacích nezasekává, jestli detekce hudebních dob probíhá správně, ale i únosně rychle. Ošem takovéto testování je velmi specifické a ve skutečnosti vzdálené realitě.

Proto jsem požádal taneční přátelé a učitele, aby mi věnovali svůj čas a umožnili mi vyzkoušet aplikaci ve skutečném provoze. Tedy na lekci irského setového tance. Povedlo se mi uskutečnit 4 návštěvy na lekcích, během kterých jsme zkoušeli tančit tance Connemara, Every Mans Chance a West Kerry. Jsem moc vděčný všem tanečnicím a tanečníkům za jejich trpělivost při počátečních problémech s hlasitostí a vypadkem hlášení u některých skladeb.

Měl jsem štěstí, že tanečníci přítomní na lekcích byli v irském setovém tanci poměrně zkušení a běžně rozumí významu názvu jednotlivých prvků v setovém tanci. Což značně ušetřilo čas pro testování. Učitel nemusel tolik vysvětlovat základní pohyby a kroky. Aplikace má v současné době a verzi pouze ambice napovídat kroky do hudby, nikoli ještě navíc vysvětlovat kroky začátečníkům.

Úplně první testování nebylo příliš povedené, protože jsem měl nejprve hlášení nastaveno tak aby se ozvalo přesně v čase kdy se má krok tančit. Tanec nedopadl příliš dobře protože tanečníci nebyli schopni včas zareagovat na nápovědu aplikace. Po následné diskusi s tanečníky jsme se shodli, že bude vhodnější posunout hlášení o 1 až 2 takty dříve.

Při druhém běhu testování jsme narazili na problém velmi záhy. Již při prvním napovězeném slovu. Hlas byl totiž příliš hluboký a nevýrazný. V tomto případě jsem totiž aplikaci vyjímečně zkusil nainstalovat na čerstvě smazaný a resetovaný iPad, který ve výchozím nastavení nedisponuje tolik kvalitním hlasem syntetizátoru a ani výběr hlasových lokalizací není tolik široký. Figuru tance jsme však dotančili správně, protože ač nápověda nebyla slyšet úplně pronikavě, tak přesto byla správně načasovaná. Pro mě bylo z tohoto testu poučení,

že budu muset kontrolovat zda zařízení má kvalitní hlas syntetizátoru. S ostatními tanečníky jsme se shodli, že nejpěknějším pro ucho a nejpronikavějším je hlas irské ženy. Na druhé místo jsme umístili hlas britského muže, jehož vyslovování znělo nejjasněji.

Třetí testování odhalilo u některých tanců příliš dlouhé věty, které sice většinou začaly včas, ale skončily hrát mnohem později. Délka hlášené věty se navíc pak ještě zvětšila v momentech, kdy je nutné navíc hlásit také pár, který bude krok tančit. S ohledem na tuto zpětnou vazbu jsem pak ladil algoritmus pro plánování nápovědy tak, aby posouval například větu o deseti znacích posunul tři takty dříve než na krok dojde.

Čtvrté testování jsme chtěli vyzkoušet v menším uzavřeném salonku v hospodě. Okolí salonku bylo docela rušné a hluk pronikal také dovnitř. Přesto po připojení iPadu k menšímu reproduktoru jsme slyšeli většinu hlášených nápověd. Zkoušeli jsme vybírat pro každou figuru úplně jinou hudbu. V jednom případě algoritmus rozpoznal některé doby trochu posunuté, proto bylo hlášení ke konci tance lehce zpožděné.

Vybrané tanečníky jsem pak nechal aplikaci ovládat samostatně a sledoval jsem jejich akce. Potěšilo mě, že se vyplatilo vypuštění obrazovky s nastavení skladeb pro všechny figury, protože nyní noví testeři na první pokus věděli, kde mají vybrat hudbu. V jednom případě tanečník přehlédl výběr hudby, avšak alert hláška jej zastavila před spuštěním hlášení kroků (opět díky dřívějšímu testování, kde toto bylo navrženo).

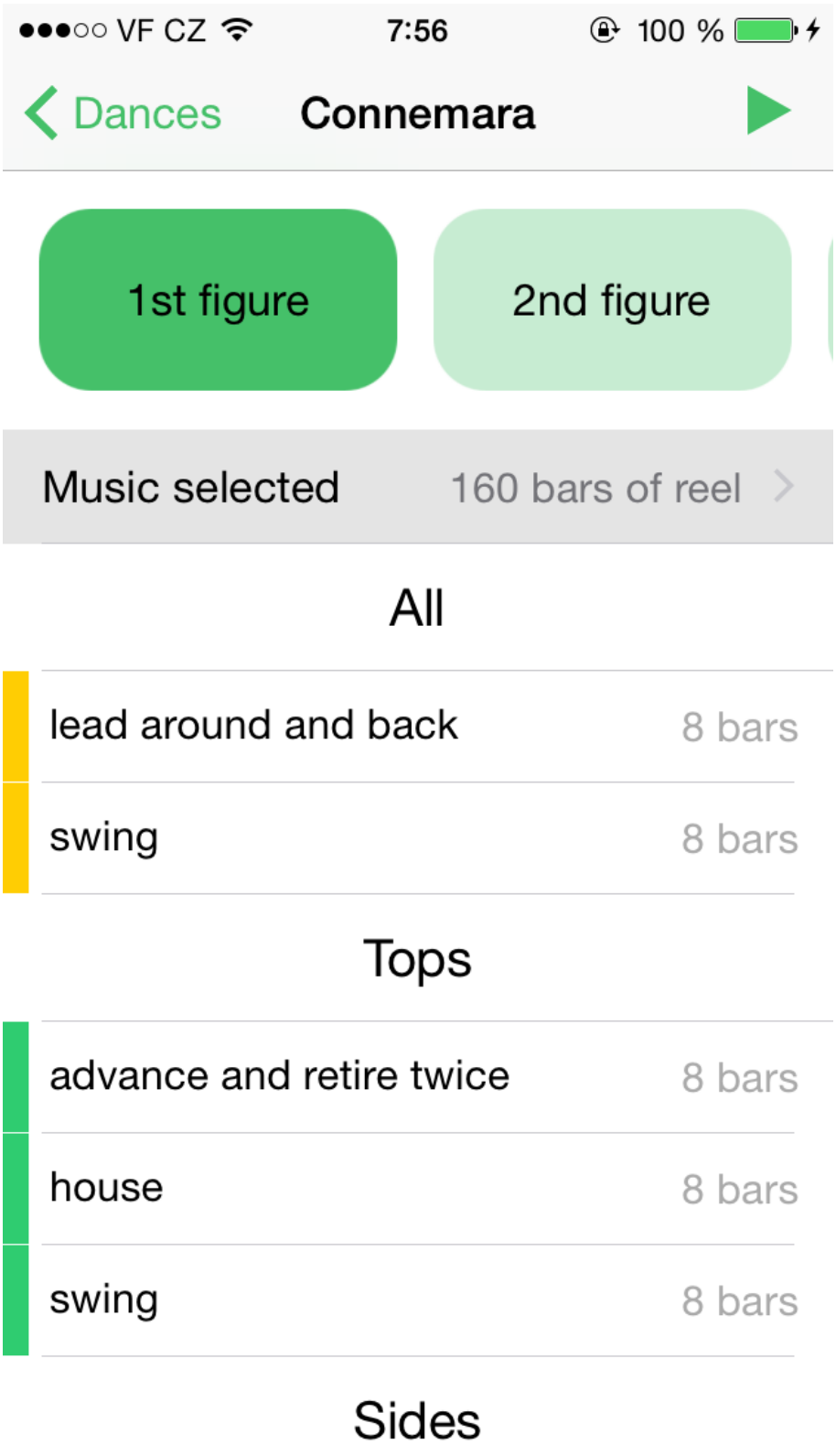
Nadšeni byli tanečníci z vyznačení podobných tanečních kroků v detailu figury. Původně jsem tuto věc zamýšlel jako doplněk a ozdobu, ale byl jsem mile překvapen, že má barevné rozlišení podobných kroků také účitečný smysl a usnadňuje orientaci ve krocích figury.

Během domácího testování jsem dále narazil na problém při počítání taktů u některých skladeb v jigového rytmu, kde by bylo nutné upravit poměr přeskočených dob při výpočtu taktů v modelu aplikace.

Během některých lekcí se mi podařilo natočit průběh testování hlášení kroků. Myslím, že videa jsou velmi pěkná a určitě stojí za shlédnutí. Celkově jsem měl z veškerého testování velikou radost protože mi pomohlo objevit situace, které bych doma od stolu jen tak jednoduše nezopakoval. Motivovalo mě to upravovat algoritmus přimo na lekcích. Rovněž vzešlo z testování spousta dalších nápadů a myšlenek na rozšíření. Tyto nové požadavky bude nutné projít a zrevidovat jakou by mohly mít prioritu pro většinu cílové skupiny..

- pouze tanec ... http://youtu.be/1SY2y0x-Ml0
- pouze použití aplikace ... http://youtu.be/6wBlbLSRIPc

.



Obrázek 5.3: Obrazovka realné aplikace zobrazující první figuru

Obrázek 5.4: Obrazovka realné aplikace zobrazující nápovědu během hlášení blížících se kroků

Kapitola 6

Závěr

6.1 Původní stav a jeho omezení

Při tančení irských setových tanců hraje podstatnou roli hlášení blížících se tanečních kroků včas do hudby. Pokud z nějakého důvodu hlášení není dostatečné dochází v tanci k omylům a účastníci si tanec tolik neužijí.

Příčinou nedostatečného hlášení může být například nezkušenost hlasatele, který neví přesně kdy taneční kroky hlásit nebo hlásí kroky příliš potichu. Problémem může být nedostatečná soustředěnost hlasatele - například pokud se hlasatel-učitel snaží zároveň opravovat chybující tanečníky a zároveň udržet pozornost k hudbě.

Druhým trochu vzdáleněji příbuzným problémem je pak také vytváření choreografií na míru hudbě. Pokud by bylo možné automaticky vytvořit taneční kroky tak aby zohledňovaly vývoj melodie hudby, mnoho by to pomohlo choreografům jako základ při vymýšlení choreografií.

6.2 Výuka tance pomocí aplikace

Pro automatickou detekci dob v hudbě se mi podařilo nastudovat, implementovat a pro potřeby irské hudby vylepšit algoritmus založený na analýze signálu hřebenovým filtrem. Implementace měla svá úskalí hlavně kvůli složitějšímu převodu kódu z Matlabu do nízkoúrovňového jazyka C a poznávání nového jazyka Swift pro programování iOS aplikací. Přesto se mi povedlo vytvořit aplikaci, která je výborně architektonicky navržená, důsledně odděluje logiku od prezentace a dbá na dobrou odezvu uživatelského rozhraní. Díky průběžnému testování při návrhu uživatelského rozhraní se mi povedlo vytvořit aplikaci, která vyhovuje potřebám cílového uživatele.

6.3 Generování tance pomocí aplikace

Aplikace pro generování kroků dovede zpracovat segmentovanou hudbu a vybrat pro ni vhodné taneční kroky s ohledem omezení jako obtížnost kroků nebo podobnost kroků při změně hudby. Výstupem je tanec, který lze použít jako hotovou choreografii pro začínající

tanečníky. Pro pokročilé tančníky lze choreografii použít jako základ pro přidávání ornamentů.

6.4 Budoucí rozvoj

Aktuální aplikace dovede dobře rozpoznat doby v reelovém rytmu. Rád bych v budoucnu vylepšil rozpoznávání také dalších rytmů (zejména jig). Užitečné by také mohlo být propojení aplikace s generátorem tanečních kroků, kdy by aplikace například pro účely tréninku dovedla generovat kroky na míru vybrané hudbě. Pro účely generátoru tanečních kroků by bylo vhodné implementovat také rozpoznávání podobných částí hudby.

Příloha A

Seznam literatury

Literatura

[1] Dr. John Cullinane, 1990, Further Aspect of the History of Irish Dancing

[2] Pat Murphy, 1995, Toss the Feathers: Irish Set Dancing, Mercier Press: Dublin.

[3] Pat Murphy, 2000, The Flowing Tide: More Irish Set Dancing, Mercier Press: Dublin.

[4] Pat Murphy, 2009, Apples in Winter: Irish Set and Social Dancing, self published, Ireland.

[5] Pat Murphy, 2014, Tabhaí dom do Lámh: Irish Set and Social Dancing, self published, Ireland.

[6] Dance Minder, https://www.danceminder.com/ (verze dostupná dne 5. května 2015)

[7] Scheirer E. D., Tempo and Beat Analysis of Acoustic Music Signals, J. Acoust. Soc. Am., vol. 104, pp. 588–601, January 1998.

[8] Davies M. E. P., Plumbley M. D., Context-dependent beat tracking of musical audio. IEEE Transactions on Audio, Speech and Language Processing, 15(3), 1009-1020, 2007.

[9] Klapuri A., Eronen A., Astola J., Analysis of the meter of acoustic musical signals, IEEE Trans. Audio, Speech, and Language Processing, 14(1), 2006.

[10] Driedger J., Time-scale modication algorithms for music audio signals. Master's thesis, Saarland University, 2011

[11] Apple Human Guidelines https://developer.apple.com/library/ios/documentation/userexperience/ (verze dostupná dne 5. května 2015)

[12] Pitch control http://www.dspdimension.com/admin/time-pitch-overview/ (verze dostupná dne 5. května 2015)

[13] Davide Rocchesso - Introduction to Sound Processing, 2003

[14] Rhythm classification https://www.irishtune.info/rhythm/ (verze dostupná dne 5. května 2015)

[15] Dance Minder, Connemara Reel description https://danceminder.com/dance/show/connre (verze dostupná dne 5. května 2015)

[16] Hutchinson Guest, A. (1989) Choreographics: a comparison of dance notation systems from the fifteenth century to the present. Routledge ISBN 90-5700-003-2

[17] Shorthand Notation for Israeli dances http://www.israelidances.com/StepsLegend.htm (verze dostupná dne 5. května 2015)

[18] Beat Detection Algorithms article by Frédéric Patin http://www.gamedev.net/page/resources/_/technical/math-and-physics/beat-detection-algorithms-r1952 (verze dostupná dne 5. května 2015)

[19] Creating Music by Listening by Tristan Jehan http://web.media.mit.edu/ tristan/phd/dissertation/chapter3.l (verze dostupná dne 5. května 2015)

[20] Phase Vocoder by Flanagan, Golden, and Portnoff http://www.ee.columbia.edu/ dpwe/e6820/papers/FlanG66.pdf (verze dostupná dne 5. května 2015)

[21] Time and Pitch Scaling by Surina http://www.surina.net/article/time-and-pitch-scaling.html (verze dostupná dne 5. května 2015)

[22] David Malah (April 1979). "Time-domain algorithms for harmonic bandwidth reduction and time scaling of speech signals". IEEE Transactions on Acoustics, Speech, and Signal Processing. ASSP-27 (2): 121–133.

[23] Záznam solového tance Blackbird https://www.youtube.com/watch?v=4ZnNNmhoOt4 (verze dostupná dne 5. května 2015)

[24] Nahrávka skladby Silver spear včetně not http://www.bbc.co.uk/radio2/r2music/folk/sessions/swf/01.html (verze dostupná dne 5. května 2015)

[25] A Genetic Algorithm for Resource-Constrained Scheduling http://lancet.mit.edu/mwall/phd/thesis/thesis.pdf (verze dostupná dne 5. května 2015)

[26] Large Roulete Wheel in Genetic algorithms http://jgap.sourceforge.net/doc/gaintro.html (verze dostupná dne 5. května 2015)

[27] Illustration for genetic cross over http://en.wikipedia.org/wiki/Crossover (verze dostupná dne 5. května 2015)

[28] Callback hell vysvětlení http://callbackhell.com/ (verze dostupná dne 5. května 2015)

[29] Promise Kit http://promisekit.org/ (verze dostupná dne 5. května 2015)

[30] User interface first, Jeff Atwood, 37signals, http://blog.codinghorror.com/ui-first-software-development/ https://gettingreal.37signals.com/ (verze dostupná dne 5. května 2015)

[31] Cocoapods, https://cocoapods.org/ (verze dostupná dne 5. května 2015)

Příloha B

Seznam použitých zkratek

JSON JavaScript Object Notation

DSP digital signal processing

SOLA sychronous overlap and add

KNN k-nearest neighbors

⋮

Příloha C

Instalační a uživatelská příručka

C.1 Spuštění iOS aplikace

- Předpokladem je práce v operačním systému Mac OS X
- Pokud budete chtít aplikaci vyzkoušet na iOS zařízení je nutné toto zařízení vlastnit, mít jej zaregistrované na developer.apple.com (nutné mít placený vývojářský účet)
- Nejprve nainstalujte aplikaci XCode, například prostřednictvím App Store
- Otevřete soubor ve složce ios-aplikace/dance-caller.xcodeproj
- Vyberte zařízení nebo simulátor a aplikaci spusťte

C.2 Spuštění genetického algoritmu

- Předpokladem je mít naistalované NetBeans IDE
- Otevřete soubor v NetBeans IDE složku java-generovani-tance
- Po spuštění aplikace provede testy a vypíše je do konzole

Příloha D

Obsah přiloženého CD

- readme.txt - popis adresářové struktury a obsahu adresářů
- text - text práce
- matlab - prototypy a testy v matlabu
- ios-aplikace - implementace mobilní aplikace
- java-generovani-tance - implementace aplikace pro generování tance