FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Sonifikace obrazu
Student:	Radka Kolembusová
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Počítačová grafika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout způsob sonifikace dat z obrazu a generování zvuku a hudby.

1) Proveďte rešerši:
   a) metodiky sonifikace,
   b) obdobných projektů.
2) Analyzujte současná softwarová řešení pro sonifikaci obrazu.
3) Navrhněte způsob, jakým vhodně namapovat modalitu zvuku na výstup obrazové analýzy.
4) Navrhněte a implementujte prototyp realizující toto řešení.
5) Otestujte na vhodných datech a debatujte výsledky.

Elektronicky schválil/a Ing. Radek Richtr, Ph.D. dne 9. února 2022 v Praze.

Bakalářská práce

SONIFIKACE OBRAZU

Radka Kolembusová

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Radek Richtr, Ph.D.
12. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěla bych poděkovat svému vedoucímu Ing. Radku Richtrovi, Ph.D., za pomoc při psaní práce, dále rodině za trpělivost a podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2022 .....

Abstrakt

Tématem bakalářské práce je návrh způsobu sonifikace dat z obrazu a generování zvuku a hudby na základě vlastností jeho barev a detekce objektů v obrazu pomocí umělé inteligence. Práci zahajuje rešerše obsahující metodiku sonifikace, přibližující odborné termíny akustiky, hudební teorie a oblasti barevných prostorů. Dále jsou zde popsány a zhodnoceny projekty podobného zaměření. Probíhá hledání a vyhodnocení mapování modalit mezi vizuální a zvukovou doménou. Analýza technologických možností se soustředí zejména na uživatelskou přívětivost a přenositelnost. Po grafickém a softwarovém návrhu je implementován prototyp aplikace a prováděno uživatelské testování.

Klíčová slova sonifikace obrazu, generování hudby, detekce objektů, AI, tvorba hudební notace, podobnost barev, CIE DELTA E, rozdíl jasu, průměrná saturace, struktura písně, harmonizace melodie, funkční harmonie, přenesení umění

Abstract

The topic of the bachelor thesis is the design of image data sonification and the generation of music and sound with use of color similarity and artificial intelligence object detection methods. The work begins with a research including the methodology of sonification, approaching the technical terms acoustics, music theory and color spaces. Furthermore, projects with similar subject of investigation are described and evaluated. The thesis continues with searching and evaluating modalities mapping methods between the visual and the audio domain. The analysis of technological possibilities focuses mainly on user-friendliness and portability. After the graphical and software design, a prototype of application is implemented and user testing is performed.

Keywords image sonification, music generation, object detection, AI, musical notes creation, color similarity, CIE DELTA E, brightness difference, average saturation, song structure, melody harmonization, functional harmony, art transformation

Seznam zkratek

GUI	Graphical User Interface
BPM	Beats per minute
MIDI	Musical Instrument Digital Interface
SMF	Standard MIDI File
MP3	MPEG-1 Audio Layer III nebo MPEG-2 Audio Layer III
WAV	Waveform audio file format
DXF	Drawing Exchange Format
SVG	Scalable Vector Graphics
SSML	Speech Synthesis Markup Language
PDF	Portable Document Format
JPG	JPEG File Interchange Format
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
MIT	Licence MIT (Massachusettský technologický institut)
AI	Artificial Intelligence

Kapitola 1
Úvod

Hudba je součástí života téměř každého z nás. Najdeme ji kdekoliv a kdykoliv. Ať už máme zapnuté rádio nebo jen nevědomky naslouchámе rytmicky bubnujícím kapkám deště na sklo okna. Hudba je přítomná v každém okamžiku naší existence, stejně jako tlukot srdce. Obdobnou zkušenost máme s obrazy, dalším výsledkem našich smyslů. Jsou součástí našich představ a myšlenek. Svou barevností, strukturou a obsahem nám otevírají dveře ke vzpomínkám, které jsme zažili. Zrak nám dává vizuální pohled na svět a zdání o jeho uspořádání. Zvuk tento dojem zesiluje přidáním další dimenze.

Pro prehistorické civilizace byly tyto dvě složky reality nerozlučitelné. Přírodní zvuky i vizuální vjemy však byly v průběhu evoluce napodobovány lidmi a jak čas postupoval, postupně také ztrácely svůj původní význam. Tak se stalo, že obraz byl oddělen od zvuku a zvuk od obrazu. Jejich vztah však nebyl odstraněn úplně. Naopak. Díky nejrůznějším rituálům primitivních kmenů vznikaly mezi těmito oblastmi vazby nové, do té doby nevídané. Písně a obrazy se slévaly v představivosti lidí a pojily se s příběhem, další nedílnou složkou pozemské kultury.

Dnešní pohled na obě tato odvětví je oproti starším generacím doplněn o spoustu nových poznatků. Pomocí moderních technologií lidé pochopili jejich další rozměry. Příkladem může být Fourierova transformace, díky které dokážeme převést zvuk i obraz do frekvenčního pásma a rozložit je na jednotlivé bázové funkce. Úpravou parametrů těchto funkcí a následné inverzní transformace dosahujeme změny původního signálu. Máme tedy v rukou mocný nástroj.

I když jsme však rozložili obraz i zvuk na jednotlivé kousky stavebnice a kostky se zdají být sobě podobné, dokážeme je opět spojit dohromady tak, aby jejich vztah dával smysl? Co vlastně zvuk a obraz, potažmo hudbu a obraz, spojuje? Jsou to vzpomínky? Je to lidská představivost? V naší historii se tyto vjemy objevovaly bok po bohu zejména v umění, které vyprávělo příběh.

Otázkou však zůstává – Jak nalézt tento příběh? Jakou písní bude hrát plátno malíře, když zde autor již není, aby nám prozradil okolnosti událostí, které svým uměním ztvárnil? To je problematika, kterou se zabývá tato práce. Snaží se nalézt ten bájný most mezi hudbou a obrazem. Používá přitom moderní technologie a nástroje, jako je detekce sémantiky scény pomocí umělé inteligence a digitalizace obrazových i hudebních děl. Sleduje význam barvy, která má v počítačovém světe mnoho podob.

Klade si za cíl přispět porozumění mezi světem barev kolem nás a soustavu nejrůznějších zvuků, kterým říkáme hudba. Využít ho mohou například lidé, kteří hledají smysl života v umění nebo ti, kteří doposud obrazy našeho světa neměli možnost spatřit. Snaží se přinést kulturní obohacení obrazů, co již léta leží ve vitrínách, aniž by dostaly šanci v plném rozsahu vyprávět svůj příběh.

Kapitola 2
Členění práce

Práce začíná literární rešerší, ve které je popsána metodika sonifikace a příbuzné projekty. Následuje analýza těchto projektů a vyhodnocení jejich výhod a nevýhod. Dále probíhá zkoumání a porovnávání mapování modalit pro účely sonifikace obrazu. Vhodné technologické řešení hledá nadcházející kapitola, po které následuje grafický a softwarový návrh. Další kapitola se zabývá implementací prototypu a poslední část práce tvoří testování.

2.1 Cíle

Kapitola *Přiblížení odborných pojmů* má za cíl metodicky objasnit čtenáři odborné termíny používané v práci. To zahrnuje popis definic pojmů, jejich krátké vysvětlení a uvedení do širších souvislostí.

Cílem kapitoly *Související projekty* je dohledat a objektivně popsat již existující projekty, které se problémem zabývají nebo jsou jinak přínosné možnému řešení problému bakalářské práce. Neočekává se kompletní výčet existujících projektů. Důraz bude kladen na projekty, které autorka dále využije v práci.

Část *Analýza vlastností projektů* si klade za cíl analyzovat a subjektivně zhodnotit projekty zkoumané v předchozí kapitole. Hlavní náplní je srovnání projektů mezi sebou a provedení analýzy výhod a nevýhod. Na základě dat tohoto srovnání jsou vyvozeny závěry pro další postup v návrhu prototypu.

V oddílu *Mapování modalit* je cílem nalézt a porovnat možná řešení mapování modalit z oblasti vizuální do oblasti zvukové s přihlédnutím k předchozím projektům. U každého mapování jsou posouzeny jeho plusy a mínusy. Na závěr je vybráno mapování, které bude dále použito v bakalářské práci.

Kapitola *Návrh technologických možností* má za cíl zhodnotit a porovnat možné technologie, které budou následně využity pro implementaci řešení. Toto hodnocení zahrnuje analýzu rozhraní aplikace, určení programovacího jazyka a návrh podoby vstupu a výstupu. Na závěr následuje volba podporovaných operačních systémů.

Část *Návrh uživatelského rozhraní* rozvíjí cíl připravit podobu a formu uživatelské rozhraní na komunikaci s uživatelem. Je odkázáno na předchozí verzi projektu, zmíněn prvnotní návrh a následně rozšíření o další prvky. Na závěr je ukázána finální podoba rozhraní.

V kapitole *Softwarový návrh* je cílem navrhnout softwarové části pro budoucí implementaci řešení problému. Návrh zahrnuje analýzu funkčních a nefunkčních požadavků, model případů užití

a podobu softwarové architektury.

Cílem části *Implementace* je popsání hlavních částí programu. Dále je zde rozebráno propojení grafického uživatelského rozhraní a výpočetní části aplikace. Předposlední sekci tvoří oddíl zabývající se vysvětlením zastavení výpočtu aplikace, po němž následuje uvedení hlavních použitých knihoven.

Kapitola *Testování* má za cíl provést uživatelské testování aplikace. To zahrnuje připravení person, připravení testovacích dat, scénářů pro testování a následné moderování testů. V této části jsou výsledky testování také vyhodnoceny.

Poslední obsahovou částí práce je kapitola *Budoucí rozšíření*, s cílem nastínit, jak by mohl být prototyp aplikace dále vyvíjen a rozšiřován. Jsou zde zmíněny některé nápady, které se již do rozsahu práce z prostorových nebo časových důvodů nevešly.

Kapitola *Závěr* si klade za cíl vyhodnotit celý průběh práce, poukázat na funkcionality aplikace a posoudit splnění požadavků. Jsou v ní shrnuty jednotlivé části práce.

V přílohách se dále nachází *Ukázky výsledků práce* a *Návod k instalaci a spuštění aplikace* pod systémy Windows 10 a Manjaro Linux. Dále instrukce ke spuštení aplikace ze zdrojového kódu. Je tu schován i popis adresářové struktury přiloženého media.

Část I

Rešerše

Kapitola 3

Přiblížení odborných pojmů

V práci se vyskytuje několik termínů, které by čtenáři, neznalého této problematiky, mohly působit potíže. Z toho důvodu se začíná vysvětlením těchto pojmů ve formě stručného slovníku.

3.1 Akustika a hudební teorie

> *„Akustika je věda o zvuku, jeho vzniku, šíření a vnímání.“*
>
> — L. Zenkl, ABC hudební nauky [1]

Pro lepší orientaci v této bakalářské práci je nejdříve třeba čtenáře seznámit s několika základními pojmy akustiky a hudební nauky. Jednotlivá klíčová slova budou představena od obecnějších k těm více do hloubky rozvedeným. Hudební teorie je velmi rozsáhlý obor, pro který je typická velká romanitost v pravidlech a interpretacích jednotlivých pojmů, jako to zpravidla u umění bývá. Proto se pro účely této práce uplatňuje značná míra zjednodušení a některé nepodstatné detaily jsou v textu vynechány. Při vysvětlování pojmů autorka vychází z knih *ABC hudební nauky* od L. Zenkla [1], *Učebnice harmonie* od J. Kofroně [2] a z diplomové práce *Harmonizace lidových písní u lidových souborů* od Š. Kytnerové [3].

Zvuk vzniká chvěním hmoty, která rozechvívá okolní vzduch. Ten následně vibruje ve sluchovém ústrojí člověka. Tyto vibrace pak jedinec vnímá jako zvuk. Zvuky se dělí na tóny a hluky.

(Hudební) tón je zvuk vzniklý pravidelným chvěním hmoty s určitou výškou, kterou lze většinou určit velmi přesně. Dle nástroje, na který je hrán, získává určitou barvu. Na základě toho, jak dlouho tón zní, určujeme jeho délku. Hudební tón může být slyšet silně, ale také slabě, to je dáno jeho hlasitostí. Hudba využívá tóny jako svůj základní kámen. Příkladem může být stisk klávesy na klavíru nebo brnknutí o strunu kytary.

Hluk je zvuk, který vzniká nepravidelným chvěním hmoty, a jeho výšku většinou nelze určit, nebo ji lze určit pouze přibližně (vysoko, ve střední poloze, hluboko). Podobně jako u tónu, lze u hluku rozpoznat barvu, délku a hlasitost. Díky vlastnostem nepravidelného chvění se však může barva

i hlasitost v průběhu znění zvuku měnit. V hudbě se hluky využívají například v podobě perkusí (zvuk malého bubnu, činelu).

Kmitání hmoty (zdroje) lze názornit periodickou křivkou. Rychlost kmitání vnímáme jako výšku tónu (čím pomalejší kmit, tím hlubší tón) a odchylky od středu osy jako jeho hlasitost (čím dál, tím silnější). Jednoduché křivky (sinusoidy) dávají za vznik nejjednodušších tónům – sinusovým tónům. Ty se běžně v přírodě skoro nevyskytují a můžeme je vyrobit například pomocí oscilátorů. Tóny, které se vyskytují všude kolem nás – tóny složené, jsou tvořeny z několika sinusových tónů různých výšek (nazýváme je tóny částkové). Pokud tyto tóny uspořádáme podle výšky, vytvoří tzv. `harmonickou řadu`. Zvuky s nepravidelnou výškou se nazývají neharmonické. Ucho člověka pak vnímá tyto složené tóny jako tón jediný a jeho výšku usuzuje dle prvního částkového tónu v harmonické řadě (nejspodnějšího sinusového tónu).

Harmonická řada je uspořádána dle výšek tónů dle určitých pravidel – poměru jednotlivých frekvencích vůči sobě, anebo chcete-li velikosti intervalů. Nejprve je oktáva, pak kvinta, kvarta, velká tercie, malá tercie, a tak dále. Částkové tóny v ní mohou znít různě silně, některé mohou i chybět. Když se na řadu podíváme z hlediska frekvencí kmitočtů, zjistíme, že částkové tóny jsou celonásobkem částkového tónu prvního (základního). V některých oblastech techniky se těmto frekvencím říká *vyšší harmonické*. Každý vyšší tón oktávy má kmitočet dvakrát větší, než spodní tón oktávy.

Spektrum tónu zahrnuje všechny harmonické i neharmonické částkové tóny. Jestliže ve spektru převládají neharmonické částkové tóny, jde o hluk.

Barva zvuku je závislá na poměru, výšce a síle částkových tónů v tónovém spektru. Díky tomu tak může mít stejný tón hraný na kytaře barvu jinou než na pianu.

Půltón je nejmenší výšková vzdálenost mezi dvěma tóny, používaná v evropské hudbě. Tento pojem může značit i jednotlivé tóny chromatické řady.

Celý tón je výšková vzdálenost dvou půltónů.

Základní tónová řada obsahuje sedm základních tónů: `C`, `D`, `E`, `F`, `G`, `A`, `H`. V této řadě se dává důraz pouze na výšku tónu, nikoliv na jeho další vlastnosti. Obsahuje dva půltóny (`E`–`F`, `H`–`C`) a pět celých tónů (`C`–`D`, `D`–`E`, `F`–`G`, `G`–`A`, `A`–`H`)

Odvozený tón je tón, který byl od základního tónu vytvořen zvýšením nebo snížením výšky o půltón. Při každém zvýšení se k základnímu tónu přidá křížek (`C` → `C#`) a při snížení malé písmeno b (`C` → `C`).

Enharmonická záměna je různý notový zápis dvou stejných tónů. Příkladem je `C#` a jeho ekvivalent `D` v temperovaném ladění nebo různé zápisy tónů `B` (také ozn. jako `A#`) a `H` (také ozn. jako `B`) napříč světovými kulturami.

Tónová soustava je přehledné uspořádání všech tónů (bez tónů odvozených), používaných v hudbě, pouze dle jejich výšek (barva, délka ani hlasitost se neberou v úvahu). Základem této soustavy je základní tónová řada, která se devětkrát za sebou opakuje v různých výškových polohách. Tóny se označují svými jmény ze základní tónové řady, spolu s číslovkou, která určuje, ve kterém jsou opakování. Kdybychom měli vypsat tón C ve všech jeho opakováních, bylo by to následovně: C_2, C_1, C, c, c^1, c^2, c^3, c^4, c^5.

Oktáva je pojem s mnoha významy. Kromě rozsahu tónů mezi tónem základní tónové řady v tónové soustavě k jeho nejbližšímu opakovanému jmenovci vyšší výšky, tak bývá označován i interval o výškové vzdálenosti 12 půltónů. Příkladem může být vzdálenost/rozsah mezi tónem c^3 a tónem c^4.

(Absolutní) výška tónu je měřena v počtu kmitů tělesa za vteřinu (kmitočtem) v jednotkách Hertz (Hz). Příklad výšky hlubokého tónu je 24 Hz a vysokého tónu 4000 Hz.

Relativní výška tónu nebere ohled na přesný kmitočet tělesa, které tón produkuje, ale je dána výškovým vztahem k tónu jinému. Hudba je na těchto relativních vztazích založena. Využívá se jich například při situaci, kdy zpěvák není schopen vyzpívat vysoké tóny v melodii písně. Protože si však pamatuje relativní vzdálenosti mezi jednotlivými tóny, zazpívá ji o několik poloh níže, ve výškách posunutých o konstantní počet půltónů. Pro člověka bez absolutního sluchu (schopnost rozeznávat tóny dle jejich absolutních výšek) znějí tyto své melodie stejně.

Chromatická řada je tvořena všemi 12 tónovými výškami, které v oktávě rozlišujeme: C, C#, D, D#, E, F, F#, G, G#, A, B, H. Její tóny jsou od sebe výškově vzdáleny vždy stejně – o půltón (narozdíl od základní tónové řady).

Stupnice je klesající/stoupající řada hudebních tónů seřazených dle jejich výšek v rozmezí jedné oktávy s určitými pravidly, určujícími vzdálenosti mezi jednotlivými stupni stupnice a počet tónů v ní obsažených. Každá stupnice má základní tón, kterým začíná a končí. Nejznámnějšími druhy stupnic jsou stupnice durová a stupnice mollová.

Základní tón stupnice je počátečním a koncovým tónem stupnice, která se podle něj též jmenuje. Příkladem může být stupnice G dur, která začíná tónem G. Ve vztahu k tónině, se tomuto tónu říká tónika.

Tónina je volné pořadí tónů stupnice v hudbě. Některé tóny ze stupnice jde vynechat, ale je důležité, zachovat uspořádání tónů tak, aby tónika vynikala jako tón hlavní. Stupně tóniny se označují pořadovými čísly dle svého umístění ve stupnici.

Tónika tvoří první stupeň a také základní tón v tónině. Například tónina G dur má jako tóniku tón G. Tónikou velmi často melodie začíná nebo se k ní vrací. Melodie, která je ukončena tónikou, působí na posluchače dokončeným dojmem (již nepociťuje nutnost dalšího pokračování).

Diatonická stupnice je stupnice, ve které se vyskytují různé tónové vzdálenosti mezi jejími stupni.

Stupnice durová je diatonická stupnice, která má mezi jednotlivými tóny vzdálenosti (v půltónech): 2, 2, 1, 2, 2, 2, 1. Má osm stupňů. Protože se však tón na VIII. stupni opakuje (oktáva), řadíme ji mezi sedmistupňové stupnice. V angličtině se nazývá *major*.

Stupnice mollová je aiolským módem stupnice durové (existuje k ní paralelní durová stupnice se stejnými tóny, která se jmenuje dle jiného tónu).[1] Příkladem může být stupnice `C dur` paralelní ke stupnici `a moll`. Vzdálenosti mezi jednotlivými tóny v půltónech jsou: 2, 1, 2, 2, 1, 2, 2. Taktéž je sedmistupňová diatonická stupnice. V angličtině se nazývá *minor*.

Interval je výšková vzdálenost dvou tónů. V hudbě se většinou neuvádí přesná výšková vzdálenost v poměru frekvencí, ale pouze přibližná - například v půltónech. Samotný poměr frekvencí tónů je dán použitou soustavou ladění hudebního nástroje. Pokud znějí dva tóny současně, mluvíme o **harmonickém intervalu**, pokud následují za sebou, nazýváme tento **interval melodický**.

Délka tónu je označení trvání noty v čase (délka znění), které je vztaženo k předznamenání příslušného taktu. V hudbě rozlišujeme několik hlavních délek tónů: *nota celá, nota půlová, nota čtvrťová, nota osminová, nota šestnáctinová, nota dvaatřicetinová, …* Každá délka tónu může být prodloužena o polovinu své délky. To je značeno připsáním tečky k notě.

Pořadí tónů melodických intervalů je dalším dělení intervalů. Pokud melodie vzniklá v pořadí vyšší–nižší tón, interval je stoupající (svrchní). Je-li to naopak, nazývá se tento interval klesající (spodní).

Části intervalu jsou dvě. Tónu, od kterého interval tvoříme se říká `základní` a konečnému tónu `intervalový`.

Jména intervalů v klasické hudební teorii bývají často zavádějící. Jméno intervalu má zpravidla dvě části, první **jméno základní** určuje jeho vzdálenost od základního tónu v jeho durové stupnici. **Jakost intervalu** upřesňuje vzdálenost v půltónech (protože durové stupnici není všech 12 tónů). Navíc díky enharmonickým záměnám[2] se dvěma stejným intervalům může říkat zcela jinak. To se zdá být pro následné programování aplikace velmi nepraktické. Tomuto problému lze předejít převedením klasických jmen intervalů na jejich vzdálenost v půltónech v rámci chromatické řady.[3] V tabulce 3.1 je toto mapování vyjádřeno. V textu jsou používány dle potřeby oba způsoby názvosloví.

[1] Pokud začneme durovou stupnici od VI. stupně, dostaneme nějakou stupnici mollovou.

[2] Předpokládáme temperované ladění.

[3] Tento čin je obhájen samotným vypočítáváním intervalů z durové stupnice a jejich postupným odvozováním. Pokud vytvoření hledaného intervalu (například malé tercie od tónu `C`), je třeba: 1. Určit základní tón (tady tón `C`)) 2. Utvořit durovou stupnici se základním tónem 3. Najít stupeň intervalu shodný se stupněm stupnice (čistá tercie, to je tón `E`) 4. Snížit/zvýšit tento tón o určitý počet půltónů na základě jakosti intervalu (zde je jakost malá, snížíme tedy čistou tercii o půltón a dostaneme tón `E`) 5. To je pak intervalový tón intervalu. Tímto způsobem lze získat každý stoupající interval v evropské hudbě. Pro klesající interval je princip obdobný (jen se ptáme, v jaké stupnici s hledaným základním tónem se nachází intervalový tón na daném stupni intervalu).

Pokud však na základě tohoto návodu vždy vzniká durová stupnice, která má pevně dané vzdálenosti mezi jednotlivými stupni v půltónech, závisející na pozici základního tónu v chromatické řadě, vyplývá z toho, že jakýkoliv interval z klasické hudby má pevně danou svou velikost (počet půltónů), o který se musí základní tón posunout v (cyklické) chromatické řadě, aby se dostal k tónu intervalovému. Proto není nutno ho pokaždé znovu a znovu přepočítávat.

Přenesený interval je druh interval, který svým rozpětím přesahuje oktávu. Vznikne tak, že základní tón přeneseme o jednu nebo více oktáv níže, anebo tón intervalový o jednu nebo více oktáv výše. Příkladem je interval přenesená sekunda (nóna).

Tabulka 3.1 Vzdálenosti intervalů v půltónech

Stupeň	Počet půltónů	Jméno v klasické hudební teorii	Některé alternativní názvy	Příklad
I.	0	prima	(čistá) prima	`C–C`
	1	malá sekunda	zvětšená prima	`C–C#`
II.	2	sekunda	(velká) sekunda, zmenšená tercie	`C–D`
	3	malá tercie	zvětšená sekunda	`C–D#`
III.	4	tercie	(velká) tercie, zmenšená kvarta	`C–E`
IV.	5	kvarta	(čistá) kvarta, zvětšená tercie	`C–F`
	6	zmenšená kvinta	zvětšená kvarta	`C–F#`
V.	7	kvinta	(čistá) kvinta, zmenšená sexta	`C–G`
	8	malá sexta	zvětšená kvinta	`C–G#`
VI.	9	sexta	(velká) sexta, zmenšená septima	`C–A`
	10	malá septima	zvětšená sexta	`C–B`
VII.	11	septima	(velká) septima, zmenšená oktáva	`C–H`
VIII.	12	oktáva	(čistá) oktáva, zvětšená septima	`C–c`
	13	zmenšená nóna	zmenšená přenesená sekunda	`C–c#`
	14	nóna	(čistá) nóna, přenesená sekunda	`C–d`

Konsonance (souladnost) intervalu se určuje u intervalu harmonického. Za konzonantní jsou považovány všechny intervaly čisté (1, 4, 5, 8), velká tercie, malá tercie, velká sexta, malá sexta. Opakem konsonance je **disonance** (nesouladnost).

Akord jsou současně znějící nejméně tři tóny různých výšek. Dle počtu zároveň znějících tónů akordy třídíme na trojzvuky, čtyřzvuky, pětizvuky a vícezvuky. Podle stavby intervalů v akordu na akordy složené z tercií nebo kvart. Názvy druhů akordů obvykle vycházejí z názvu intervalu nejnižšího a nejvyššího tónu.

Kvintakord je trojzvuk složený ze dvou tercií. Jméno získal na základě intervalu mezi nejnižším a nejvyšším tónem (kvinta). Dle jakosti tercií rozlišujeme čtyři druhy kvintakordů: durový (velká a malá), mollový (malá a velká), zvětšený (velká a velká), zmenšený (malá a malá). Jednotlivé tóny jsou pojmenovány dle stupňů v diatonické stupnici (od nejvyššího po nejnižší): prima, tercie, kvinta. Kvintakordy jsou nazývány dle své primy. Příkladem může být `kvintakord C dur` s tóny: `C` (čistá prima), `E` (velká tercie), `G` (čistá kvinta).

Septakord je čtyřzvuk složený z kvintakordu a jedné další tercie. Jeho krajní tóny tvoří septimu. Dle kombinace malých a velkých tercií můžeme vytvořit celkem sedm septakordů. Septakord má pak dva názvy – jeden udává typ kvintakordu (dur → tvrdě, moll → měkce, zvětšený → zvětšeně, zmenšený → zmenšeně) a druhý jakost vzniklé septimy (malá septima, velká septima, zmenšená). Pro příklad lze uvést `tvrdě malý septakord` $\texttt{C}^7$ s tóny: `C` (čistá prima), `E` (velká tercie), `G` (čistá kvinta), `B` (malá septima), kterému se přezdívá také `dominantní`.

Zn.	Stupeň	Jméno v klasické hudební teorii	Příklad akordu v tónině `C dur`	Tóny akordu
T	I.	Tónika	`C dur`	`C`, `E`, `G`
	ii.	Střídavá dominanta	`Dmi`^{7}	`D`, `F`, `A`, `C`
	iii.	Vrchní medianta	`Emi`^{7}	`E`, `G`, `H`, `D`
S	IV.	Subdominanta	`F dur`	`F`, `A`, `C`
D	V.	Dominanta	`G dur`	`G`, `H`, `D`
	vi.	Spodní medianta	`Ami`^{7}	`A`, `C`, `E`, `G`
	vii.	Neúplná dominanta	`Hmi`$^{7/5-}$	`H`, `D`, `F`, `A`

Tabulka 3.2 Harmonické funkce akordů pro tóninu C dur

Nónový akord vznikne, pokud k septakordu přidáme další tercii. Dostaneme pětizvuk, jehož krajní tóny tvoří interval nóny. Dle jakosti nóny pak vzniká buď velký nónový nebo malý nónový. Pro příklad `malého nónového akordu` `C`^{9} s tóny: `C` (čistá prima), `E` (velká tercie), `G` (čistá kvinta), `B` (malá septima), `C#` (malá nóna).

Harmonická funkce akordu v tónině je založena na snaze akordu po pohybu. Je dána tím, že ne všechny akordy v tónině mají stejnou důležitost. Dle toho je dělíme na akordy hlavní a vedlejší. Nejvíce důležitým je tzv. `tónika`, akord na I. stupni tóniny. Dalším hlavními akordy jsou `dominanta` (V.) a `subdominanta` (IV.), které mají k spodním nebo horním tónům tóniky kvintový poměr. Zatímco `tónika` symbolizuje pocit klidu, `dominanta` má tendenci stoupat vzhůru. Po jejím přehrání cítí posluchač nutnost nějakého pokračování, ideálně k tónice. Subdominanta také tíhne k tónice, ale na opačnou stranu. Její tóny budí dojem, že chtějí klesat dolů. Pocit klidu nastane, pokud zahrajeme po dominantě tóniku a taktéž po subdominantě (tam lze k tónice přejít přes dominantu). Tyto tři hlavní akordy nám stačí na vyjádření tóniny (uvedení všech jejich tónů). Akordy tvořeny na ostatních stupních tóniny se nazývají vedlejší, které se používají pro obvzláštnění harmonie. S akordy hlavními tvoří sekundový nebo kvintový poměr a jejich funkce jsou obdobné akordům hlavním. Zatímco hlavní akordy se tvoří pomocí tónů v příslušných dur stupnicích a jejich stupně se značí velkými římskými písmeny, vedlejší se zpravidla tvoří ve stupnicích mollových a značí se písmeny malými.[4] V tabulce 3.2 je vidět přehled harmonických funkcí, spolu s jejich značením.

Kadence je posloupnost akordů, ve které se vystřídají všechny základní harmonické funkce. Příklad: `T` → `S` → `T` → `D` → `T`.

Tónové rozložení je možné si představit jako cyklickou mřížku, která dává do harmonické souvislosti jednotlivé půltóny v ní uvedené v závislosti na jejich okolí.

Nota symbolizuje vlastnosti hudebního tónu (zejména jeho výšku, délku a hlasitost) v nezvukové podobě (většinou v notovém zápise či v lidské řeči). Slouží jako návod pro hudebníky, kteří po jejím přečtení vyprodukují na svůj hudební nástroj daný hudební tón, čímž dostane svou barvu.

Pomlka je druh noty vyjadřující ticho. Při pomlce hudební nástroj nehraje žádné tóny.

Rytmus je střídání tónů různých délek.

[4] Platí pro durovou tóninu. Například u mollové je to naopak.

Tempo udává rychlost, ve které se skladba hraje. Obvykle se měří v `BPM` (Beats per Minute), což značí počet úderů za minutu. V skladbě je následně potřeba uvést, jaký typ noty časový úsek mezi dvěma údery metronomu představuje. Obvykle se používá nota čtvrťová.

Takt je základní členění notového zápisu hudební skladby. Každý takt má předem určeno, kolik dob se do něj vleze (čitatel zlomku). Dále pak hodnota jednotlivých dob v notách (jmenovatel zlomku).

Doba je stejně dlouhý časový úsek v taktu.

Dynamika je název pro sílu přednesu (hlasitost) hudby. Pomocí dynamických znamének v notách hudebník ví, jestli hrát silně nebo slabě, s přízvukem nebo bez. Skladatel může vyžadovat i postupné zesilování/zeslabování tónu v určitých pasážích skladby.

Přízvuk (akcent) zdůrazňuje označený tón/akord oproti ostatním výrazným zesílením.

Repetice je opakování části skladby. V notovém zápisu bývají takty repetice označeny uvnitř dvou dvojteček s taktovou čárou.

3.2 Barevné modely a prostory

V následující kapitole budou stručně přiblíženy barevné modely a prostory používané v bakalářské práci. Bude také vysvětlen rozdíl mezi barevným modelem a barevným prostorem. V textu bude vycházeno především ze zdrojů: *Barevné prostory a správa barev* [7], *Grokking the Gimp* [8] a *Velká kniha barev* [9].

Barva je vlastnost pozorovaného objektu, která se projeví, když na něj dopadne světlo. Objekt část světla pohltí, část sebou nechá projít a zbytek světla odrazí. Tento paprsek následně letí až do oka pozorovatele. Dle toho, které vlnové délky světla objekt odráží, ale také které vlnové délky světlo obsahovalo, se v lidském oku aktivují tři druhy tzv. *čípků*. Každý z nich je citlivý na

Obrázek 3.1 Originální obrázek (muž s ohněm), zobrazený v barevném prostoru `sRGB`). Zdroj: *Wikipedia* [4].

Obrázek 3.2 Světlost obrázku (složka lightness) zachycená v barevném prostoru `L*a*b*`. Zdroj: *Wikipedia* [5].

Obrázek 3.3 Světlost obrázku (složka value) zachycená v barevném prostoru `HSV`. Zdroj: *Wikipedia* [6].

jinou vlnovou délku: *červenou, zelenou* a *modrou*. Dle jejich zareagování na výskyt různých vlnových délek[5], pak mozek dostane informaci o barvě [9].

Spektrální barva je monochromatická barva. To znamená, že ji jde vyjádřit bez skládání více světelných vlnových délek.

Barevný model je teoretický způsob popisu barvy. Barevné modely obvykle dělíme na *aditivní* (založeny na skládání světelných zdrojů) a *substraktivní* (založeny na odrazu světla od povrchů).

Barevný prostor je konkrétní množina barev. Může být nezávislý na zobrazovacím zařízení nebo závislý. Obvykle bývá založen na *barevném modelu*.

RGB je aditivní barevný model, který vychází z toho, jak člověk vnímá barvu (pomocí tří čípků citlivých na *červené, zelené* a *modré* světlo). Ve světě počítačů bývá nejběžnějším způsobem, jak zachytit informaci o barvě v obrázku. Lze si jej představit jako jednotkovou krychli v jejíchž vrcholech jsou černá, bílá, červená, zelená, modrá, azurová, purpurová a žlutá barva [7].

CIE RGB je barevný prostor, závislý na zobrazovacím zařízení, který je tvořen na základě `RGB` barevného modelu [7].

HSV je barevný prostor, založený na barevném modelu `HSV`, který vychází z `RGB`. Skládá se ze tří složek: *Hue (barevný tón)*, *Saturation (sytost)* a *Value (jas)*. Někdy bývá označován jako `HSB`. Převládající spektrální barva je určena pomocí barevného tónu. Sytost určuje příměsi jiných barev a jas množství bílého světla [7, 8].

L*a*b* (CIELAB) je barevný prostor, nezávislý na zobrazovacím zařízení. Skládá se ze tří souřadnic: `L`, `a`, `b`. Umožňuje vypočítání objektivních odchylek (rozdíl barev) mezi jednotlivými barvami z odchylek jasu (složka `L`) a odchylek chromatických souřadnic (složek `a` a `b`) [7]. Rozdíl barev je vypočítáván ze vzorce *CIE DELTA E* [10] a představuje důležitou, obecně uznávanou, metodu hodnocení rozdílu barev, podle které jde porovnávat např. kvalitu monitorů, shodu mezi tisky apod. [7]. Jak už bylo zmíněno, `L*a*b*` velmi dobře reflektuje, jak se liší jednotlivé barvy z hlediska světlosti. Na obrácích 3.1 (original), 3.2 (`L*a*b*`) a 3.3 (`HSV`) je vidět srovnání barevného prostoru `L*a*b*` a `HSV` z hlediska světlosti obrázku. Obrázky se tolik liší zejména proto, že se prostor `HSV` dívá na všechny barvy z hlediska světlosti stejně. Oproti tomu barevný prostor `L*a*b` zachycuje vnímanou intenzitu jasu v závislosti na odstínu (žlutou vnímá člověk je světlejší než modrou).

[5] Z toho vychází později zmíněný model `RGB`.

Kapitola 4

Související projekty

Mapování modalit z oblasti vizuální do zvukové není v historii lidstva žádnou novinkou. Existuje řada projektů, která se touto problematikou zabývala a dospěla k rozmanitým výsledkům. Následuje stručný popis několika prací, které autorka vyhodnotila pro svůj postup jako nejvíce inspirativní.

4.1 Cross-Domain Analogy: From Image to Music

V roce 2017 byl v rámci mezinárodního workshopu *Musical Metacreation* [11] představen software, který si kladl za cíl inspirovat se kreativitou z obrazu a použít ji pro skládání rockové hudby. Autorky analyzovaly vlastnosti vizuální a zvukové domény a navrhly několik vhodných relací mezi nimi. Tyto vztahy byly následně použity při implementaci. Výsledný program generoval `MIDI` soubor vytvořeného hudebního artefaktu, skládající se z hlavní melodie, akordového doprovodu a bicích. Generované hudební skladby byly dány k posouzení několika dobrovolníkům.

			Main Melody		Scales	Rhythm		Percussion	Harmony
			Notes	Total Duration	Major vs Minor	BPM	Time Signature		Chords
Image	Image Dimensions			X			4 / 4 (Rock)		
	Color	Individual Pixels	X						
		Variations				X		X	
		Histogram						X	X
		Warm and Cold Colors			X			X	X
		Same Color Regions	X						

Obrázek 4.1 Přehled použitých relací obrázku a hudby v práci [11]

Tvorba relací

Ve vizuální doméně došly autorky k závěru, že současné technologie nedostačují tomu, aby odhalily přesnou sémantiku obrázku. Rozhodly se tedy pojmout obraz více abstraktně, a to na základě barev, kterými je reprezentován a které také tvoří základní kámen relací. Podobně zabarvené pixely se shlukují dle vzorce o barevném rozdílu [10] do tzv. *regionů*, které jsou chápány jako samostatná jednotka. Převažující barva v dané části obrázku je určena pomocí histogramů.

Používají se tři přístupy k mapování barvy na zvuk. První z nich je založen na popisu barvy regionu a aplikuje se při tvorbě hlavní melodie. Jednotlivé barvy jsou rozloženy do tří složek: *hue* – odstín, *value* – světlost, *intensity* – bledost. Následně je využito toho, že počet základních barevných odstínů je stejný jako počet hudebních tónů v chromatické řadě (12). Každý odstín tedy získá určitý tón dle obrázku 4.2. Míra světlosti odpovídá zařazení tónu do oktáv (čím vyšší světlost, tím vyšší oktáva). Bledost oproti tomu působí na dynamiku noty (více bledá barva znamená nižší hlasitost). Délka tónu se řídí velikostí (počtem pixelů) barevného regionu.

U tvorby akordového doprovodu autorky vycházejí z předpokladu, že teplé barvy vyjadřují převážně šťastné emoce a chladné barvy emoce spíše negativní a smutné. To reflektují v zvukové doméně kvalitou akordu – pro teplé barvy je použit durový akord (v hudbě znějící jako veselý), zatímco pro barvy chladné akord mollový (v hudbě používán pro smutné pasáže). Obrázek 4.2 zobrazuje rozdělení barev na teplé a chladné.

Pro kompozici bicích se vycházelo ze 42 rytmů o jednom taktu, používaných v populárních rockových písničkách. Pro každý z nich byla předem vypočtena tzv. *emoční hodnota*, která je při generování písně porovnávána s emoční hodnotou dané části obrázku. Při tomto porovnávání vycházejí autorky z možnosti porovnání jednotlivých emočních vlastností hudby v závislosti na *tónině*, *hlasitosti*, *tempu*, *hustotě not* a jejich *délek* a emočních vlastností obrázku v závislosti na *teplotě barvy*, *průměrné velikosti barevného regionu*, *bledosti barvy* a jejím *odstínu*.

Na základě analýzy oblasti zvukové vznikly dva přístupy ke generování hudebních artefaktů: `přímý`[1] a `harmonizovaný`[2]. První z nich pohlíží na jednotlivé sektory obrázku odděleně. Ten druhý omezuje akordový doprovod na čtyři různé akordy, bicí na dva různé rytmy pro celou píseň a klade důraz na pohled na obrázek jako celek.[3]

Vstupní parametry

Protože se nenaskytla možnost program otestovat, budou zde brány v úvahu vstupní parametry zmíněné v odborném článku. Zásadním vstupem pro běh programu je zdrojový obrázek. Oproti hudbě však není u obrázku mnohdy jednoznačné, kde začíná a kde končí. Tato věc je však pro sestavení hudebního díla velmi důležitá. Proto se autorky rozhodly dát uživateli na výběr celkem z osmi možností, zahrnující roh obrázku, kde se má začínat, a různé směry čtení (schéma 4.3).

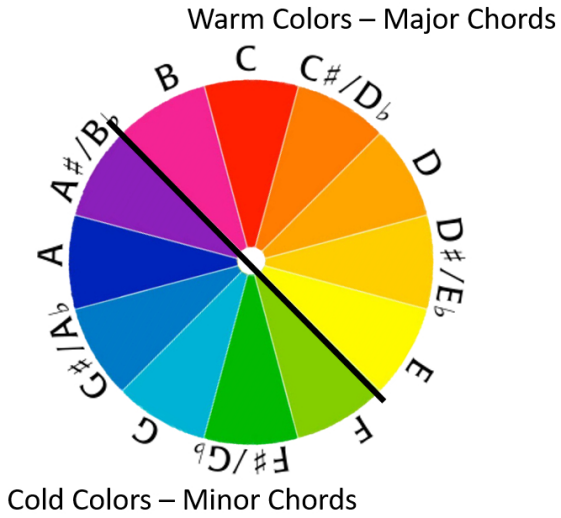
Algoritmus

Nejprve je na základě míry obměny barev v obrázku určena *rychlost* generované hudby v BPM. Tato barevná variace je počítána s pomocí průměrné velikosti barevného regionu poměrově na řádcích i sloupcích. Následně je tato procentuální hodnota poměrově namapována na rozsah 50–150 BPM.

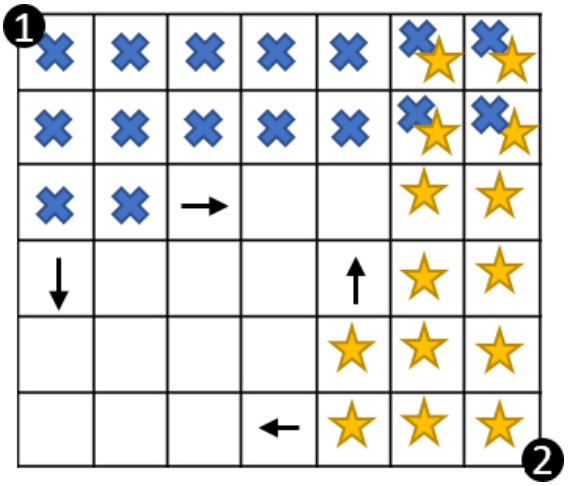
[1] raw

[2] harmonized

[3] Autorka *Joana Borges Teixeira* následně tyto dvě verze rozšířila o genetický algoritmus v rámci své diplomové práce [12]. Jeho počáteční populace z předešlých verzí vychází.



Obrázek 4.2 Mapování barev na tóny



Obrázek 4.3 Možnosti směru čtení obrázku

Tento rozsah byl zvolen dle předpokladu, že rockové písničky se obvykle pohybují v tempu 90–110 BPM a rozšířen za účelem zvýšení rozlišení barevné variace obrázků.

Vzhledem k tomu, že jde o kompozici rockové hudby, ve které se většinou skládá v taktové předznamenáním $\frac{4}{4}$, rozhodly se autorky pro zachování fixní hodnoty. Poté se obrázek rozdělí na *sektory*, které by měly být v ideálním případě rozměru blížící se ke čtverci. V úvahu je bráno, aby výsledné hudební dílo nebylo ani moc dlouhé ani moc krátké: 1 min $\leq$ délka hudebního artefaktu $\leq$ 3 min.

Pro každý sektor je dle principů popsaných výše vygenerována část melodie, jeden akord pro rytmický doprovod a vybrán vhodný rytmus bicích. V rámci tvorby melodie pro každý barevný region v sektoru vznikne tón. Dle toho, zda jde o přímou nebo harmonizovanou verzi algoritmu, se generuje buď akord dle dominantní barvy sektoru, anebo náhodný akord dle pravidel pravděpodobnostně svázaných se čtyřmi hlavními dominantními barvami v obrázku. V druhém případě se po celou písničku mohou hrát maximálně čtyři různé akordy. Výběr rytmu bicích probíhá v přímé verzi na základě *emoční hodnoty sektoru*, zatímco v harmonizované verzi jsou vybrány dva nejvíce vyhovující rytmy na základě *emoční hodnoty celého obrázku*. První z nich hraje po většinu písně a druhý do toho pravidelně vstupuje. Dle autorek to má za následek zvýšení variability písně.

Nástrojové obsazení jednotlivých hudebních linek je následující: bicí linka – bicí, hlavní melodie – piano, akordový doprovod – piano. Vygenerované hudební stopy se následně převedou do `MIDI` formátu a ten se přenechá uživateli jako výstup.

Výsledky

Vzniklé hudební artefakty byly dány dobrovolníkům k posouzení. Autorky zjišťovaly reakce na dvě základní otázky: *Zní tento zvukový záznam jako hudba? Přijde vám jako rocková hudba?*

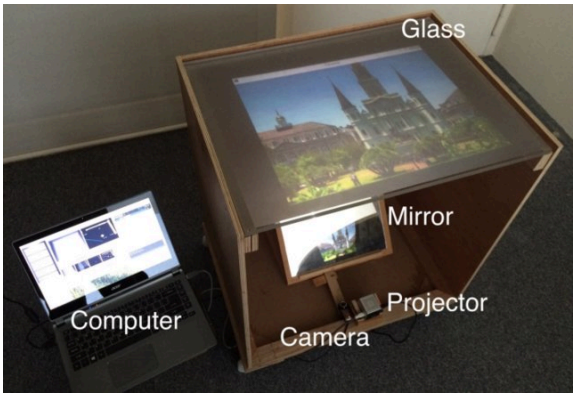
Výsledky[4] byly dle autorek překvapivě dobré. Dle statistiky publikovaných v odborné práci byly v průměru všechny artefakty uznané za hudbu. Klasifikace audio stopy jako rockové muziky oproti tomu dopadla docela špatně. Ani ne polovina lidí přiřadila artefakty k rockovému žánru. Autorky

[4]Webová stránka, kde měly být publikovány výsledky, je nedostupná a na záznamu z konference [13] není zvuk dostatečně slyšet. Při posuzování kvality hudebních děl se tedy vychází ze stanovisek autorek textu.

to vysvětlují použitím zvuku piana namísto zvuku kytary, která je pro rock tolik typická.

4.2 Generate expressive music from picture with a hand-made multi-touch music table

Jedná se o provizorně sestavené zařízení, které bylo představeno na mezinárodní konferenci *New Interfaces for Musical Expression* [14] v roce 2015. Projekt byl navržen pro expresní vyjadřování hudby z obrazu a určen zejména pro začínající hudebníky. Zařízení se skládá ze skleněné tabule, zrcadla, projektoru, webové kamery a počítače, jak je vidět na obrázku 4.4. Dle autora má jít spíše o novodobý hudební instrument a možnost inspirace, než o generátor hudby.



Obrázek 4.4 Multi-touch music table

Tvorba relací

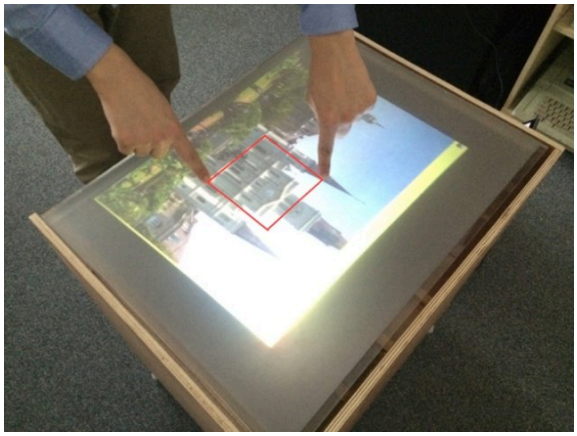
Uživatel je v projektu vnímán jako osoba tvořící pomyslný most mezi časovou osou generované hudby a bezčasovostí obrázku. To on určuje, která oblast obrázku se přehraje jako první. Svými vstupy (tvar gesta, jeho velikost a kategorie, tlak doteku na desce, pozice doteku vzhledem k promítanému obrázku) určuje jakým způsobem budou dále interpretované parametry obrázku (svítivost a barva).

Práce se inspirovala v jevu označovaném jako *synesthesia*. Lidé ovlivnění tímto fenoménem [15] mívají přenášené vjemy z jednoho smyslu do toho druhého. Někteří z nich při poslechu hudebního tónu pozorují určitou barvu [16]. Autor projektu vychází při tvorbě relací z prací J. L. Caivano [17] a S. E. Palmera [18], kteří vztahy barvy a výšky tónu zkoumali. Tím se utvořili relace: barva části obrázku určuje výšku tónu a zářivost přibližuje oktávu.

Vstupní parametry

Celý program je řízen pomocí dotyků na skleněnou tabuli, kam zespodu promítá projektor přes zrcadlo zdrojový obrázek. Uživatelova interakce s touto tabulí je zachycena webkamerou a předána počítači ke zpracování. Využit je nástroj oboru počítačového vidění – software pro vizuální sledování *Community Core Vision* a protokol *TUIO*[5].

[5] Tangible User Input/Output protocol [19]



Obrázek 4.5 Ohraničení oblasti obrázku prsty

Pomocí gest, kde může uživatel zapojit až 10 prstů naráz, je utvářena hudba. Gesta jsou rozdělena do tří kategorií. Prvním gesto vzniká ohraničením 2D oblasti (viz obrázek 4.5) a v hudební rovině reprezentuje akord. Pokud uživatel pouze přejede po obrázku prstem, definuje 1D křivku, na jejíž bázi systém zkonstruuje a přehraje melodii. Třetí způsob generování hudby se skládá ze sekvence několika krátkých dotyků. Gesto je ukončeno přiložením dlaně na obrázek. Jednotlivé doteky prstů formují rytmický vzor, který se přehrává dokola do doby, než uživatel opětovným dotykem dlaně tuto smyčku přeruší.

Algoritmus

Obrázek je nejprve rozdělen na segmenty o stejné[6] velikosti. Pro každý segment je následně vypočtena výška tónu a jeho oktáva dle výše uvedených vztahů. Tyto hodnoty se předpočítají pro daný obrázek jednou a během produkce hudby už se nemění. Program vyhodnocuje uživatelovu interakci se zařízením v reálném čase a na základě jednotlivých druhů gest hraje příslušné hudební prvky.

Tvorba akordu vychází z vlastností segmentů ležících v dílčí části obrázku. Z histogramu této oblasti je zjištěno, které výšky tónů a jakých oktáv se vyskytují v segmentech nejčastěji. Následně se vybere 4–8 nejčastějších, ze kterých se dle určitých pravidel vytvoří akord. Do tohoto výpočtu je vnesena jistá náhodnost randomizovaným počtem tónů, které mohou být v akordu. Délka akordu je vypočtena nelineárním postupem vztahem k velikosti ohraničené části obrázku.

Melodie je určena 1D křivkou. Segmenty, které křivka protne určují svou svítivostí oktávu a nejčastější barvou hudební tón. Tlak prstu uživatele určuje hlasitost daného tónu. Postup přiřazení délky jednotlivých tónů nebyl v článku zmíněn. Ve verzi prezentované autorem je možno hrát až dvě melodie současně.

Rytmický doprovod vzniká na základě sekvence alespoň 4 po sobě jdoucích klepnutí na skleněnou tabuli. Vzor je potvrzen a jeho přehrávání ukončeno kontaktem celé dlaně s tabulí. Přesné

[6] Algoritmus využívá neměnného rozlišení promítaného obrázku 640 x 480 px.

odvození těchto not není bohužel nikde zmíněno. Autor pouze zmiňuje, že uživatel vyťukává toto gesto na desku ve svém vlastním rytmu.

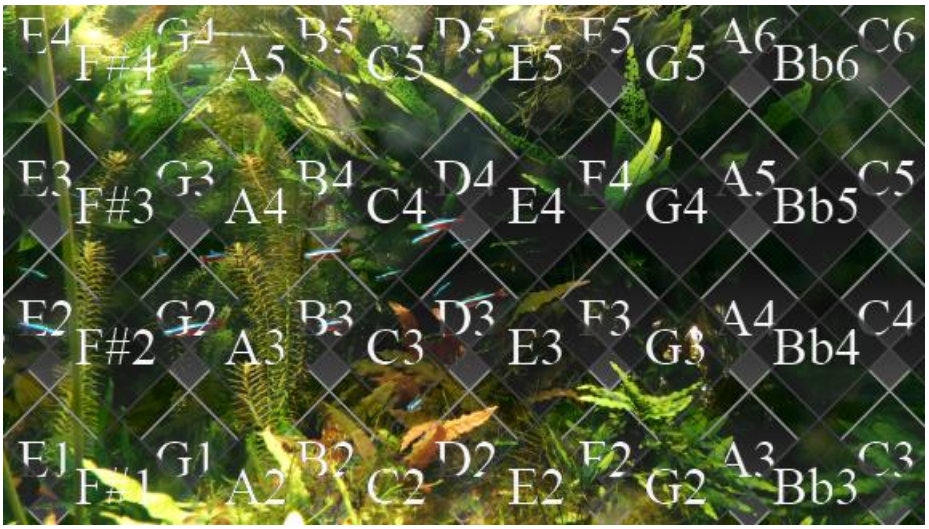
Výchozí nastavení projektu používá chromatickou stupnici se všemi 12 tóny. Dle autora je to však možné upravit. To stejné platí pro hraní více než dvou melodií zaráz. Vzhledem k tomu, že celé zařízení je vyrobeno z rozkladatelných částí, je možné jednotlivé komponenty měnit a seřizovat dle potřeby.

Výsledky

Projekt bohužel neposkytl žádné výsledky, které by byly v době psaní bakalářské práce dohledatelné.

4.3 Sonification of Fish Movement Using Pitch Mesh Pairs

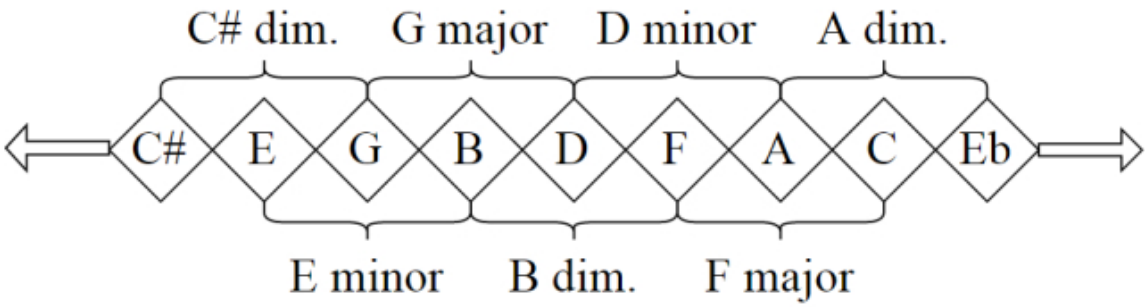
Dalším příspěvkem *New Interfaces for Musical Expression 2015* bylo nové schéma pro tónové rozložení pojmenované `Pitch Mesh Pairs` [20]. Projekt se inspiroval L. Eulerem a jeho tónovou mřížkou [21]. V jedné z ukázek práce je toto rozložení použito na vybrané hodnoty sonifikace pohybu ryb v akváriu.



Obrázek 4.6 Ukázka aplikace nového schématu výšky tónů na pohybu ryb

Tvorba relací

Princip nového rozložení spočívá ve vzájemném překrytí dvou mřížek zaplněných půltóny. Mřížky jsou překryté takovým způsobem, aby bylo snadné přejít z jedné mřížky do druhé. Obrázek 4.6 ukazuje výřez mřížek.



Obrázek 4.7 Výřez tónového řetězu s ukázkou některých kvintakordů

Struktura mřížek je dána tzv. `tónovými řetězy`[7] (na obrázku 4.7), které se nacházejí v jejich řádcích. Tónové složení těchto řetězů je od sebe odlišné pouze oktávou tónu. Vertikálním směrem vzhůru oktáva roste. Lze tedy říci, že mřížku vždy tvoří základní tónový řetěz s rekurentní funkcí změny oktávy. Tento základní tónový řetěz je u obou mřížek rozdílný.

Tónový řetěz označuje cyklickou posloupnost tónů, které mezi sebou mají určitý harmonický vztah. Dva tóny po sobě tvoří interval malé, velké nebo zmenšené tercie. Tři po sobě jdoucí tóny vytvářejí kvintakord (kvintakord durový, mollový nebo zmenšený), čtyři septakord a pět nonový akord.

Pro generování kvintakordů (durových, mollových nebo zmenšených) jsou potřeba dva tónové řetězy. V prvním z nich, začínající posloupností `zmenšeného kvintakordu C#mi`$^{5-}$ (`C#, E, G`), jich lze nalézt 18 z celkových 36. Tento řetěz bude mít v sobě kvintakordy s dominantní a subdominantní harmonickou funkcí. Zbývající jsou vygenerovány z posloupnosti začínající kvintakordem `C dur`: `C, E, G`. Kvintakordy v tomto druhém řetězu mají pro změnu tónickou harmonickou funkci. Každý z řetězů zde uvedených tvoří základní řetěz jedné z mřížek.

Obrázek 4.8 Ukázka spojení dvou tónových řetězů

Hudební vlastnosti rozložení vznikají při spojení mřížek na vhodném místě, respektujícím protínající se tóniny jednotlivých tónových řetězů (obrázek 4.8). Pro ukázku s rybami jsou důležité zejména tyto z nich: Čím menší počet sousedních tónů v jednom řetězu naráz zní, tím více je zvuk souladný a naopak. Je to dáno tím, že konsonance se od terciových intervalů, přes kvintakordy

[7] pitch chains

k septakordům a nónakordům postupně zmenšuje. Dále to, že první mřížka obsahuje akordy s dominantní a subdominantní harmonickou funkcí, zatímco druhá akordy s funkcí tónickou. Přepnutí mezi mřížkami tak způsobí změnu funkční harmonie.

Sonifikace pohybu

Aplikace `Pitch Mesh Pairs` na sonifikaci pohybu ryb spočívá v překrytí této dvojice mřížek přes video zachycující běžný život v rybím akvariu. Kdykoliv nějaká z ryb náhle zrychlí svůj pohyb, vygeneruje se zvuk dle její pozice v akváriu. Horizontální směr určuje zvolenou mřížku. Pokud se ryby pohybují do vzájemně opačných stran, tóny zní pospolu nesouladně. Oproti tomu, když se ryby pohybují stejným směrem a nacházející se blíž u sebe (jako hejno), produkuje systém harmonii souladnou. Pokud se hejno naráz rozhodne pohnout v opačném směru, díky proložení dvou mřížek s navzájem opačnými tonickými funkcemi akordů, to způsobí harmonickou změnu v melodii. Čím výše se ryba v mřížce nachází, tím vyšší má přehrávaný tón oktávu. Pokud ryba plave spíše dolů, oktáva se snižuje, naopak když plave spíše nahoru, zvyšuje se.

Mřížky lze přes video libovolně natahovat a tím upravovat výsledky sonifikace. Pokud jsou mřížky roztaženy horizontálně do stran, a tím pádem dojde k omezení rozsahu tónů, je větší šance, že ryby plující ve stejném směru, i když docela daleko od sebe, budou produkovat souladné tóny. Pokud budou mřížky roztaženy vertikálně, dojde k omezení rozsahu počtu oktávy. Ryby plující nahoru tak například budou zvyšovat výšku tónu pomaleji, než kdyby mřížky roztaženy vertikálně nebyly.

Vstupní parametry

Vzhledem k tomu, že projekt byl koncipovaný jako ukázka aplikace nového rozložení, nemá uživatel možnost interakce s aplikací přes uživatelské rozhraní. Může však upravit zdrojový kód [22] ve skriptovacím jazyku *MATLAB*, a tak si program přizpůsobit dle svého gusta. Měnit se dá například zdrojové video, jak často se má samplovat a po jak velkou část videa se má sonifikace provádět. Dále je možné upravit výstupní `MIDI` zařízení, rychlost ryb, které spouštějí produkci tónů a další parametry ovlivňující výslednou sonifikaci.

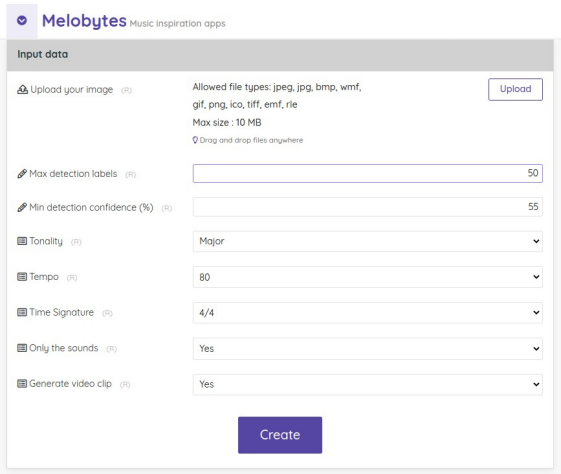
Výsledky

Výsledky sonifikace ryb a obrazu jsou k dispozici na serveru YouTube [23], a to v podobě videí. Na Githubu autora se nachází zdrojový kód programu, se kterým je možné vygenerovat další ukázky hudebních artefaktů [22].

4.4 Melobytes

Webová stránka `Melobytes` [24] od firmy *Gardos Software LTD* poskytuje sbírku několika online aplikací pro práci s médii. Mnoho z nich slouží pro generování hudby. V následujících odstavcích budou představeny ty aplikace, které tématicky nejvíce souvisí s touto bakalářskou prací. Všechny prezentované aplikace poskytují uživateli službu pomocí GUI. Výsledek je možné si poslechnout buď jako `MP3` nebo v `MIDI` formátu, případně jej sdílet na sociálních sítích nebo stáhnout v offline podobě.

AI Image to sound využívá *umělé inteligence* pro detekci objektů, motivů, scén a textů v obrazu, a na základě těchto podnětů generuje zvuky, které s nimi souvisejí. Některé parametry, za pomoci kterých aplikace zvuky generuje, lze upravit v nastavení (viz obrázek 4.9). Výsledkem jsou zejména zvuky z každodenní reality (lidské hlasy, zvuky letadla, řezání motorové pily, vrtání, cinkání sklenic, ...) spojené dohromady. Zvuk může obsahovat i hudební prvky (hra na piano, zvuky kytary). Je možné nechat si vygenerovat videoklip, který je spojením generovaného zvuku, zdrojového obrázku a závěrečnými titulky.



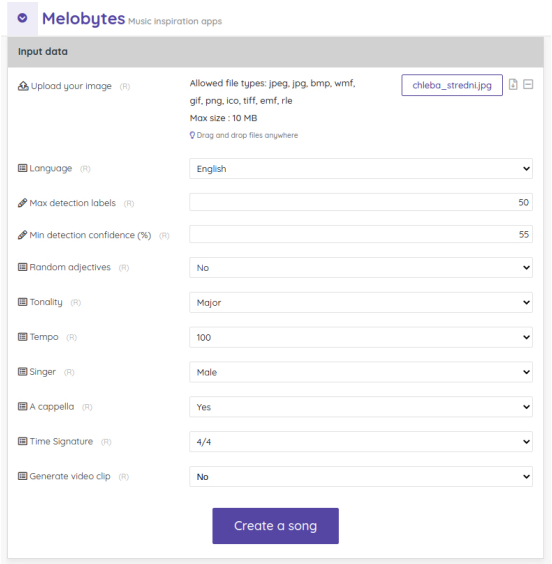
Obrázek 4.9 Ukázka AI Image to sound

Obrázek 4.10 Zdrojový obrázek – chléb ve formě

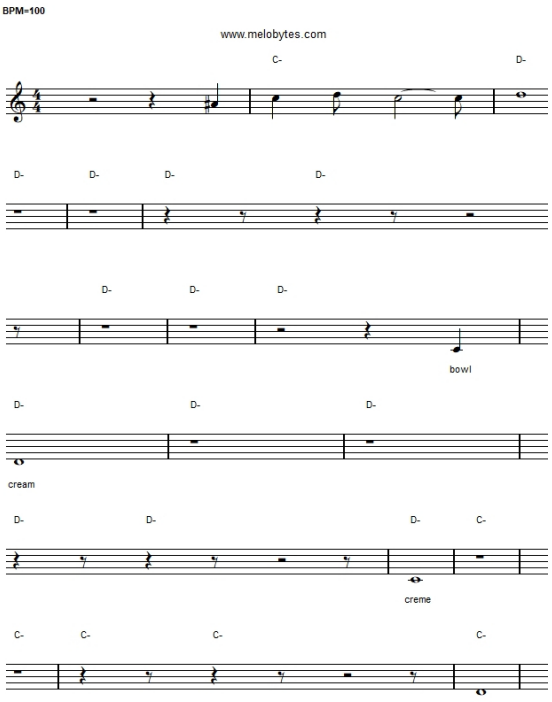
AI Image to song funguje na podobném principu jako předchozí aplikace – také pomocí *umělé inteligence* detekuje podněty v obrazu, a v jejich důsledku generuje hudbu. Výsledkem je audio nahrávka písně a notový zápis s vygenerovaným textem. Nahrávka neobsahuje záznam lidského hlasu, ale pouze jej imituje. Notový zápis je k dispozici v podobě obrázku. Pokud si uživatel přeje k písni vygenerovat i videoklip, aplikace umístí do dolní části videa (pod zdrojový obrázek) text písně a s plynoucím časem některé jeho části obarvuje. Součástí klipu jsou závěrečné titulky. V závislosti na parametrech pro generování videoklipu nemusí být hlavním předmětem pouze zdrojový obrázek, ale například animovaná postava, která pohybuje ústy v rytmu zpívaného textu. Výsledek může vypadat například takto: ze zdrojového obrázku 4.10 byl za použití parametrů uvedených v uživatelském rozhraní 4.11 vygenerován notový zápis, jehož výřez lze nalézt na schématu 4.12.

Image/Video to Music poskytuje možnost převést na hudbu kromě obrázku i videozáznam. Přes rozhraní (ukázka na obrázku 4.13) není možné nastavovat žádné parametry. Výsledkem je hudební videoklip, který obsahuje buď zdrojový obrázek v pozadí nebo variaci na původní videoklip se závěrečnými titulky, dále audio záznam a hudební dílo v `MIDI` formátu. Metoda převodu z vizuální domény do zvukové není na webu zmíněna.

Drawing to music bere na svém vstupu soubory ve formátu `DXF` a `SVG`, které představují zdrojovou kresbu. Ta je následně přeměněna na hudební dílo pomocí nespecifikovaného postupu. Uživatel má možnost nastavit několik parametrů převodu (obrázek 4.14). Výsledkem je oproti `MIDI` a `MP3` nahrávky navíc `SSML` soubor. Aplikace automaticky generuje videoklip se zdrojovou kresbou na pozadí.



Obrázek 4.11 Rozhraní AI Image to song

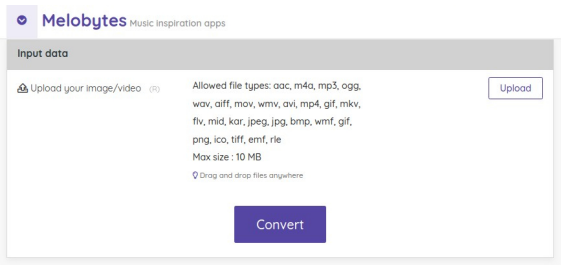


Obrázek 4.12 Výřez výsledného notového zápisu

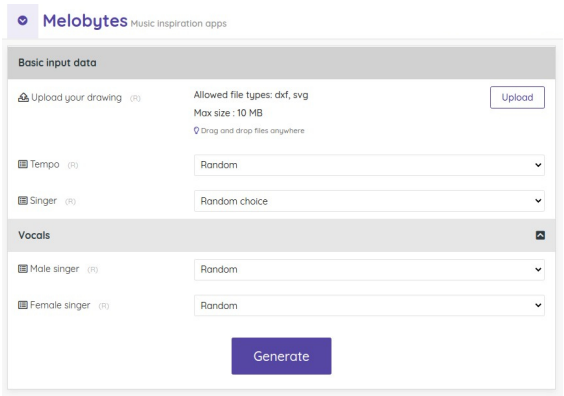
Výsledky

Webová stránka poskytuje v bezplatné verzi omezený počet používání aplikací v závislosti na časovém horizontu (např. počet konverzí za den) a také počet možných stáhnutí výsledků v `MIDI` formátu. Výsledek si je vhodné poslechout jak v `MIDI` formátu, tak i v `MP3` formátu. Podoba se totiž může výrazně lišit v kvalitě i přiřazení hudebních nástrojů k jednotlivým stopám.[8]

[8] Zřejmě vinou nedostupnosti některých zvukových samplů použitých při převodu z `MIDI` do `MP3`.



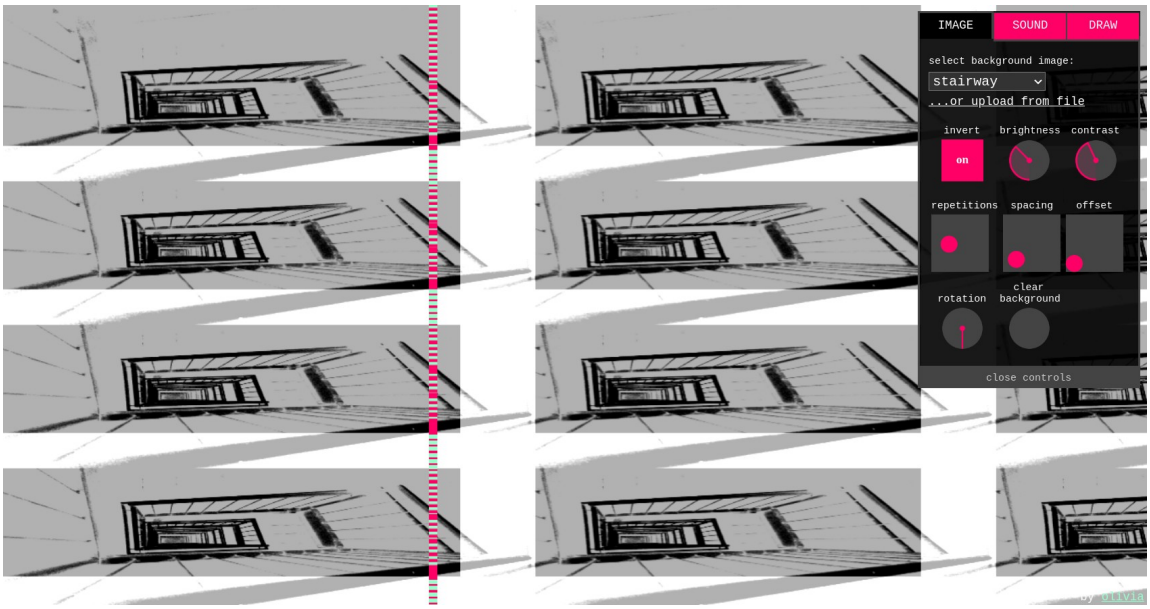
Obrázek 4.13 Rozhraní Image/Video to music



Obrázek 4.14 Rozhraní Drawing to music

4.5 Pixelsynth

`Pixelsynth` je webová aplikace [25], která na základě informace o odstínech šedi ve zdrojovém obrázku generuje sinové vlny a pak je skládá dohromady, čímž vzniká zvuk podobný hudbě. Autorka aplikace, *Olivia Jack*, se dle vyjádření na svém blogu [26] při její tvorbě inspirovala analogovým ANS syntenzátorem, který v roce 1937 navrhl Evgeny Murzin.



Obrázek 4.15 Aplikace Pixelsynth

Vstupní parametry

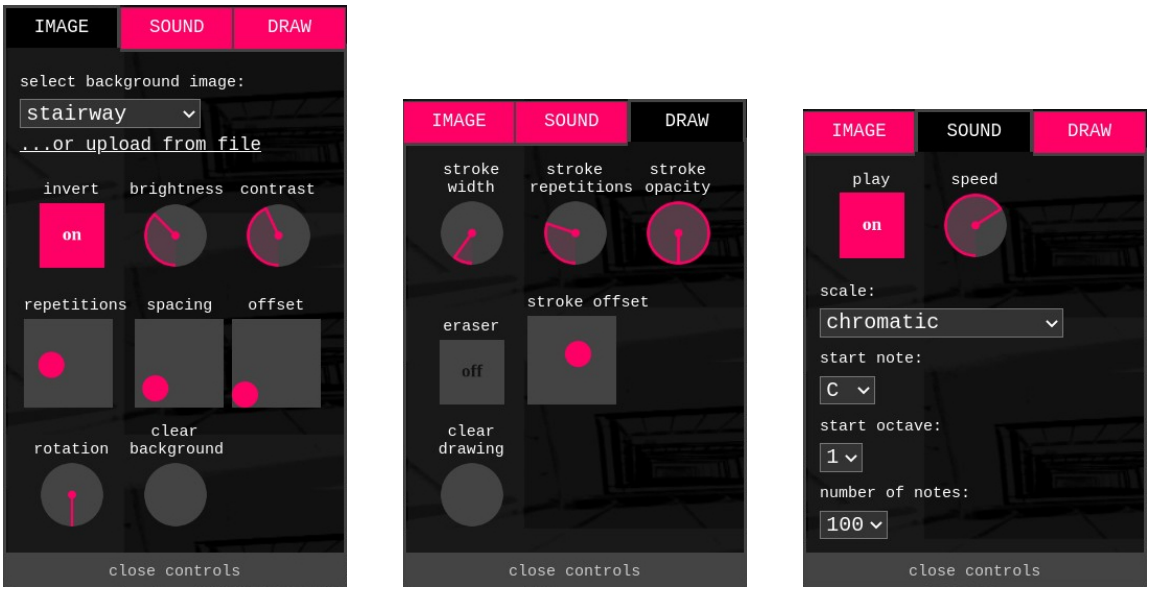
Aplikace disponuje rozmanitou škálu parametrů, které lze použít pro modifikaci generovaných zvuků. GUI v podobě tří panelů (obrázek 4.16) usnadňuje uživateli orientaci v jednotlivých nastaveních. Produkce hudby i aktualizace nastavených parametrů probíhá v reálném čase přímo ve webovém prohlížeči za použití technologií *JavaScriptu* a *HTML*.

Obrázek na pozadí lze v panelu `Image` libovolně měnit. Aplikace nabízí několik ukázek obrázků, se kterými si uživatel může hrát, ale také mu nebrání nahrát obrázek vlastní. Ten bude vzápětí převeden do odstínů šedi[9] a promítnut jako pozadí na plátno webové stránky. Následně ho lze přehrávat zleva doprava či obráceně, v závislosti na nastavení. Měnit může uživatel rychlost přehrávání, kontrast, jas, opakovaní obrazu a mnoho dalších věcí (více ve schématu 4.16).

Možnost kreslení na přichystané pozadí plátna je další z nástrojů (hlavička `Draw`), kterým může uživatel ovlivňovat výsledný zvuk. Stane se tak po překliknutí do nastavení kresby v ovládacím panelu aplikace. Stejně jako u pozadí je k dispozici mnoho nastavení (obrázek 4.16), jako například

[9] Pomocí vážené metody převodu z RGB do odstínů šedi.

šířka štětce, opakování kresleného vzoru a jeho průhlednost. Kresbu lze bez poškození pozadí odstranit nebo mazat po částech.



Obrázek 4.16 Ovládací panel aplikace Pixelsynth

Algoritmus

Zvuk vzniká pomocí syntenzátoru, který zastřešuje několik oscilátorů produkující sinusové vlny. Horizontální pozice v obrázku představuje časovou osu skladby a z vertikální vychází její náplň. V ovládacím panelu pod hlavičkou `Sound` (zobrazeno na obrázku 4.16) je možné přizpůsobit některé parametry generování. Hlavními ovládacími prvky jsou tlačítko *spuštění/zastavení přehrávání* a *tempo* generované hudby (směr čtení obrázku). Na základě analýzy zdrojového kódu publikovaného na GitHubu autorky [27] bude na následujících odstavcích ve stručnosti popsán algoritmus aplikace.

Na bázi zvoleného typu *stupnice*[10] a jejího *základního tónu*[11] dojde k vygenerování řady hudebních tónů, které jsou následně vyjádřeny ve svých frekvencích a předány dále jako materiál pro oscilátory. Parametr *počet not v řadě*[12] může být vyšší, než je počet tónů ve stupnici. Pokud algoritmus dojde na konec stupnice, začne znovu od základního tónu, tentokrát o oktávu zvýšeného. Řada začíná od základního tónu oktávy uvedené v parametru *počáteční oktáva*[13].

Pro každý tón je následně vytvořen jeden oscilátor. Těmto oscilátorům jsou přiřazeny vertikální pozice rovnoměrně rozmístěné po svislé ose obrázku. Čím vyšší je frekvence oscilátoru, tím výše je jeho vertikální pozice. Obrázkem se v průběhu generování hudby pohybuje obarvený sloupec, jež určuje aktuálně generovanou část písně a na kterém lze některé z faktorů pozorovat. Zelené úseky představují pozici oscilátorů, které mají dostatečný *gain* (vstupní zesílení). Čím větší získal oscilátor

[10] scale
[11] starting note
[12] number of notes
[13] start octave

hodnotu gainu, tím větší výšku bude mít zeleně obarvený úsek na sloupci. Gain je vypočítáván na základě jasu obrázku v přiřazené části a závisí na něm, jestli daná frekvence oscilátoru (tedy hudební tón) půjde v generované ukázce slyšet.

Výsledky

Aplikace výsledek generování přehrává přímo ve webovém prohlížeči a neposkytuje možnost uložit ho v offline podobě. Uživateli však nic nebrání nahrát si pomocí externí aplikace zvukový výstup a následně ho uložit na disk.

Část II

Praktická část

Kapitola 5

Analýza vlastností projektů

Na základě zkoumaných projektů zachycených v kapitole *Související projekty* byla provedena analýza jejich vlastností. Vycházelo se zejména z vlastností takových, které byly o daném projektu známy.[1]

5.1 Tvorba relací

Různorodost tvorby relací napříč projekty jen dále poukazuje na to, že vztahů mezi obrázkem a hudbou může být mnoho. V následujících odstavcích budou vyhodnoceny a dány do souvislosti hlavní podněty tvorby relací.

Barva

Projekt *Pixelsynth* se rozhodl informaci o barevnosti zatratit a využít pouze jas obrázku. Také práce *Sonification of Fish Movement* nevyužívá pro generování hudby barvu. Tím se však oba projekty dobrovolně vzdávají klíčového prvku obrázku. Barva je totiž věc, která dle studií [28, 29] může velmi ovlivnit vnímání obrázku divákem. Toho si jsou vědomy autorky projektu *Cross-Domain Analogy*, kde každému z 12 základních odstínů barev přiřazují jeden půltón. Přiřazení tónů k jednotlivým odstínům však není pevně dáno a závisí pouze na volbě autorek. O krok lépe postupoval projekt *Generate expressive music*, získáním dat o vztahu mezi barvou a tónem ze studie o lidí se synthezií a absolutním sluchem.

Dalším aspektem, který se k barevnosti obrázku vztahuje, je teplota barvy. Zatímco *Generate expressive music* tomuto jevu nedává žádný prostor, *Cross-Domain Analogy* přidělují studeným barvám mollové akordy a teplým barvám akordy durové. Tedy kvalita akordu je spojena s teplotou barev. Jedním z dalších prvků, souvisejících s barvou, který projekt *Cross-Domain Analogy* používá, je míra barevné variace. Mapuje ji na tempo písně. Toto mapování se zdá být přirozené, neuškodilo by však, kdyby toto mapování bylo podloženo nějakou studií. Ostatní projekty tento parametr obrázku vůbec neuplatňují.

[1] Například u projektu *Melobytes* není k dispozici ani hrubý popis algoritmu.

Jednotka versus celek

Při pohledu na obrázek se divák většinou nedívá na jednotlivé pixely, ale na jejich vzájemnou souvislost. V hudbě taktéž nestojí jedna nota vedle té druhé nezávisle, ale jde právě o relativní vztahy mezi jednotlivými výškami, ať už o melodické nebo harmonické intervaly. Algoritmus by tedy měl dávat alespoň nějaký důraz na okolí pixelu, který posuzuje. U všech posuzovaných projektů se tento princip uplatnil. U některých v míře větší, u některých v míře menší.

Projekt *Sonification of Fish Movement* se zdá být svým pohledem nejpřirozenější. Ryby v plující v hejnech [30], která v přírodě běžně tvoří za účelem obrany před predátory, zde představují celek, zatímco osamocená ryba jednotku. Aplikace z řady *Melobytes* také většinou nepoužívají pouze jednotlivé pixely, ale pomocí AI detekují objekty (jednotky) a určují přibližnou sémantiku scény (celek). Nejen objekty, ale i barvy mohou představovat vhodné rozdělení obrázku do jednotek. Využívá toho projekt *Cross-Domain Analogy*. Koncept slučování podobně zabarvených pixelů do samostatných jednotek dává dostatečný důraz na nejen vertikální, ale i horizontální okolí pixelu, bohužel však projekt vychází pouze z počítání pixelů stejné barvy, které se nacházejí v segmentu. Pohled na jednotku a celek tento projekt dále rozvíjí ve dvou verzí algoritmu. Zatímco *harmonizovaná* verze dává při tvorbě doprovodu důraz na vlastnosti celého obrázku a jeho nejčastěji zastoupených hodnot, *přímá* verze posuzuje jednotlivé části obrázku nezávisle na sobě.

Předchozí zmíněné projekty detekovaly jednotku dle určitého proměnného vzoru. Práce *Generate expressive music* ukazuje, že to nemusí být ani tak složité. Obraz rovnoměrně rozděluje na sektory a jako jednotku používá souhrné vlastnosti jejich pixelů. Asi nejméně propracovanou souvislost jednotky a celku má projekt *Pixelsynth*, když pro každý světlý pixel z předem definované množiny v obrázku spouští oscilátor na příslušné vertikální pozici. Zde mezi sebou zápasí pixely umístěné v jednom sloupci a jde spíše o souvislost harmonickou než sémantickou.

Objekty

Na konci práce *Generate expressive music* autor konstatuje, že lidský mozek při prvním setkání s obrázkem používá pro vytvoření obrazového vjemu zhruba 10 % jeho informace. Také upozorňuje na fakt, že jsou lidé obecně přitahováni k dominantnímu objektu na obrázku. Z toho usuzuje, že objekt v obrázku je pro člověka důležitý a měl by mít při zpracovávání jistou prioritu. Podobně uvažuje i tvůrce *Sonification of Fish Movement*, kde právě živé objekty (ryby) utváří svým pohybem základní data pro generování hudby.

Melobytes u některých aplikací zmiňují, že umělá inteligence hledá objekty ve scéně. Bohužel ostatní projekty detekci objektů neberou vůbec v úvahu a část informace z obrázku tak zahazují. Na druhou stranu pak obrázek posuzují více abstraktně, a ne jako fotografii s přesným uspořádáním. Stejně jako když lidský mozek sleduje realitu se záměrem ji jistým způsobem požít, vychutnat, a ne určit z ní co nejvíce informace. Toho si jsou vědomy autorky *Cross-Domain Analogy*, které konstatují, že na odhalení přesné sémantiky obrázku člověk ještě nemá vhodné technologie. K tomu se dá přidat i fakt, že v různých lidech může stejný obraz budit různé vzpomínky a tedy i pocity.

Směr čtení

Co je to směr čtení a o jaké možnosti generování hudby z obrazu rozšiřuje? Cílem je hudba. Musí tedy padnout otázka z druhé strany: Záleželo by u hudby, kdyby ji někdo pustil v obráceném pořadí nebo přímo neuspořádaně? Ano, záleželo. Hudba by tímto aktem dostala zcela jistě jiný význam.[2] Z toho důvodu je potřeba velmi pečlivě vybírat, kterým směrem bude algoritmus v 2D

[2] Některé hudební skladby byly pro tento účel dokonce designovány: *Johann Sebastian Bach* – Musikalisches Opfer.

ploše postupovat.

Každý z projektů se s tímto problémem vypořádal po svém. Zatímco autorky *Cross-Domain Analogy* nebo *Pixelsynth* daly uživateli na výběr směr, jakým algoritmus bude po scéně postupovat, jiné projekty braly informace o tomto postupu přímo ze scény. Bohužel u projektu *Cross-Domain Analogy* je pořadí barevných regionů v taktu randomizováno, čímž se zahazuje informace o výsledném uspořádání not v sektoru. Umělá inteligence v *Melobytes* se snažila objasnit sémantiku scény, na jejíž závislosti směr čtení zřejmě závisel. Projekt *Sonification of Fish Movement* vybíral místa v obrázku dle vzniku konkrétních jevů (zrychlení pohybu ryb), které byly poslány k dalšímu zpracování. Velmi nápaditě se s interpretací směru čtení obrázku popasoval autor *Generate expressive music*, když uživateli dovolil tento směr určit pohybem jeho dlaní, což posunulo postup *Cross-Domain Analogy* o úroveň výš.

Čas

Interpretace času v obrázku je velmi problematická. Čas je totiž dimenze, kterou obrázek postrádá. Je to kvůli tomu, že vyjadřuje obdobu pohledu zrakem do reality v jednom jediném okamžiku. Lidé to tak však většinou nevnímají. Jsou zvyklí domýšlet si děj či situaci na obrázku díky vlastní představivosti či značkám, kterými umělec tuto představivost ovlivňuje (čáry znamenající pohyb, vysoký kontrast malých ploch vyvolávající vlnění obrazu, atd.).

Problém je tedy určení způsobu, jakým lze tuto představivost nasimulovat v počítači. Může to být změnou rytmu, zvýšením tempa písně nebo nějakým jiným způsobem. Jednotlivé kategorie budou rozebrány v následujících odstavcích.

Tempo

Jediný projekt, který pro výpočet hudebního tempa bere prokazatelně informace ze scény, je *Cross-Domain Analogy*. Jak již bylo zmíněno výše, určuje tempo písničky na základě barevné variace obrázku. Práce *Generate expressive music* pojem tempa vůbec nezavádí – nechává uživatele, aby tempo určoval sám při hře na tento hudební nástroj, což se zdá být vzhledem k účelu projektu vhodné.

Ostatní projekty však tento účel nemají, a přesto tempo do algoritmu vstupuje přes nějakou proměnnou zadanou uživatelem.[3] To je nešťastné zejména proto, že tempo písničky je důležitý parametr, který velmi ovlivňuje pocity posluchače. Měl by tak být řízen především informacemi ze zdrojového obrázku, aby ho využil co nejvíce.

Délka písně

Jednotlivé projekty se různým způsobem staví k maximální a minimální délce písničky. Některé z nich nekladou na délku písně žádné zvláštní požadavky. Příkladem může být práce *Pixelsynth* vytvoří hudební artefakt v délce, jež koresponduje s šířkou obrázku. Nevýhodou tohoto přístupu je, že při malém tempu je hudební dílo příliš dlouhé. Tomu dávají za pravdu autorky *Cross-Domain Analogy*, které jsou přesvědčeny, že příliš dlouhý hudební artefakt by mohl být na škodu, a tak se ho snaží konstruovat tak, aby se vtěstnal mezi 1 až 3 minuty. To také zhruba odpovídá délce písní, které se běžně vyskytují v rockové hudbě, jež si autorky vybraly jako cílový žánr.

Poměrně přímočaré a logické řešení používá *Sonification of Fish Movement*, když omezuje dobu trvání písně v závislosti na délce zdrojového videoklipu, případně uživatele nechává, aby zadal délku

[3] U *Sonification of Fish Movement* je to parametr `beatDuration`, který určuje frekvenci samplování videa, a tedy i výsledné tempo skladby. U ostatních projektů se zadává rychlost písně.

v sekundách. Absence ohraničení délky však může být i předností. Vzhledem k tomu, že výsledek práce *Generate expressive music* má představovat jakýsi novodobý umělecký nástroj, definovat maximální nebo minimální čas generované hudby by nemělo žádný smysl.

Výška tónu

U většiny projektů je výběr výšky tónu[4] značně ovlivněn rozhodnutím autorů projektu. Například projekt *Cross-Domain Analogy* mapuje dvanáct hudebních tónů na dvanáct barevných odstínů bez jakékoliv věcného odůvodnění. Podobně působí i práce *Pixelsynth*, kde je výška tónu pevně svázána s vertikální pozicí pixelu a jeho umístěním v tónině. To ale aspoň dává smysl z pohledu frekvencí jednotlivých tónů, které taktéž stoupají od spodu nahoru.

V projektu *Sonification of Fish Movement* závisí výška tónu na vertikální pozici ryby v akváriu a směru kam plave (mřížka s řetězy), a to relativně vůči nastavitelné mřížce. I přes výhodu generování většího počtu variací je tento postup poněkud nešťastný. A to zejména proto, že za výše uvedeným mapováním nestojí žádná analýza obrazového materiálu, která by u tak důležitého parametru, jako je výška tónu, rozhodně chybět neměla.

Mnohem lepší přístup zvolil autor *Generate expressive music*, když využil studie o vztahu barev a hudebních tónů pozorovaných mimo jiné u lidí s synthesií. Relace tak stojí na poznatcích z reality a je větší šance, že se divákovi bude jevit jako více přirozená.

Kromě zařazení tónu do chromatické řady je potřeba určit také jeho oktávu. Projekt *Pixelsynth* postup určení hudebního tónu a jeho oktávy nijak nerozlišuje, protože díky vertikální pozici aktivního pixelu určuje obě naráz. *Sonification of Fish Movement* stanovuje oktávu také dle vertikální pozice, vertikální pozice ryby. Nevýhody a výhody tohoto řešení již byly popsány výše. Práce *Cross-Domain Analogy* oktávu určuje na základě světlosti barvy. Ač to ve své práci neuvádí, tato relace je založena na vědeckých poznatcích ze studií uváděných S. E. Palmerem [18], které v podobně vytvořené relaci oktávy a zářivosti barvy používá autor *Generate expressive music*.

Délka tónu

Projekt *Cross-Domain Analogy* určuje délku noty celkem přímočaře – z počtu pixelů obsažených v barevném regionu. Díky přímé souvislosti s rozlehlostí barvy, která určuje výšku noty, jde o velmi vhodnou taktiku. Přenáší totiž vztah barvy a její rozlohy do hudební roviny, a to v podobě výšky tónu a jeho délky. Bohužel však v jednom taktu mohou být pouze čtyři tóny a nejsou zde žádné pomlky, což hudební skladbu zbytečně omezuje.

Podobným způsobem si počínala práce práce *Generate expressive music*, když nechala mapovat uživatelem zvolenou velikost plochy (ohraničenou dvěma prsty) na délku akordu. Na základě testování prototypu byl zvolen mezi těmito dvěma modalitami nelineární vztah. Tato cesta má bohužel to omezení, že závisí pouze na vstupu uživatele, což však bylo záměrem autora při tvorbě tohoto hudebního nástroje.

Velká jednotvárnost v délkách tónů bohužel nastává u práce *Sonification of Fish Movement*, který každému tónu přiřazuje konstantní délku. To může na diváka působit nudným dojmem a nereflektuje žádný kus obrázku. Na druhou stranu se tím vytváří umělecký dojem hraní na perkuse.[5] U projektu *Pixelsynth* závisí délka noty na rychlosti, jakou se pohybuje obarvený sloupec. To je dáno vstupem uživatele, zpracovávaným v reálném čase. Podobně jako u předchozí práce je však délka noty pro každý segment obrázku stejná.

[4] Zde je pojmem *výška tónu* myšleno zařazení tónu do chromatické řady.
[5] Zvuk v ukázce připomíná hraní na xylofon.

Dynamika

Velmi dobře má tuto relaci vyřešenou projekt *Sonification of Fish Movement*, když velikost pohybu plující ryby mapuje na rychlost úderu noty v `MIDI`, kterou dále syntenzátory využijí pro vyjádření jeho dynamického průběhu. Dynamika tónu např. u piana je totiž přímo úměrná rychlosti stlačení klávesy [31]. To do jisté míry vyvažuje nudnost při konstantní délce tónu. Práce *Cross-Domain Analogy* přistupuje k problému z pohledu bledosti barev. Autorky na bledost pohlížejí jako na sílu či energii barvy, což koresponduje s hlasitostí příslušné noty. Opět přenášejí relaci vztahu z obrazu do hudby, tentokrát barvy a její bledosti do výšky tónu a jeho hlasitosti.

Projekt *Pixelsynth* určuje hlasitost noty, přesněji gain na vstupu oscilátoru, v závislosti na jasu obrázku. To je podobný princip jako u *Cross-Domain Analogy*. Autor *Generate expressive music* naplno využívá možností tangible rozhraní, když při tvorbě melodie dává do spojitosti tlak na skleněnou desku vyvíjený uživatelem a hlasitost výsledných tónů. Tento přístup je také vcelku intuitivní, díky spojitosti množství vložené síly při hraní na hudební nástroj a výslednou hlasitostí tónu v realitě. Bohužel však využívá příliš mnoho uživatelského vstupu na úkor vlastností obrazu.

Tvorba akordů

Práce *Sonification of Fish Movement* pojem akord používá trochu v jiném smyslu, než je obvyklé. A to, jako výsledek harmonie jednotlivých tónů, které hrají zaráz. Díky uspořádání tónů v mřížce a jejich vzájemné souvislosti se však tento způsob dá vnímat jako rovnocenná sekvence akordů. Kterákoliv z ryb pak může být kusem melodie (střídají se). V projektu *Melobytes* jsou akordy používány v klasickém slova smyslu, jako sekvence souborů tónů doprovázející melodii. Bohužel není znám algoritmus, který by stál za jejich tvorbou.

V projektu *Cross-Domain Analogy* se akordy v přímé verzi tvoří na základě nejvýznamněji zastoupené barvy v sektoru. To z hlediska harmonie melodie a doprovodu jistým způsobem reflektuje kontrast některých barev[6]). Ač se zdá být tato relace logická, ve skladbě mohou často vznikat dizonance. Například pokud je většina sektoru obarvená barvou, která je namapována na tón `E`, ale druhou největší část sektoru tvoří barva, která je namapována na tón `F`, který je s tónem `E` nesouladný. Akord bude postaven na základním tónu `E` a bude znít po celý takt, takže určitě dojde k překrytí nesouladného tónu a akordu.

Kvalita akordů je určena na základě teploty barvy. To se zdá být logické a intuitivní, protože ve studiích bylo prokázáno, že lidé spojují obojí – chladné barvy i mollové akordy – se smutnými pocity a naopak. Prostředníkem mezi teplotou barvy a kvalitou akordu jsou tak emoce posluchače, jak poznamenává *Palmer et. al.* [18]. Jediné co na této relaci může vadit, je to, že barvy tóny namapované k chladným barvám nikdy neutvoří `durový` akord a naopak. Například tak ve skladbě nikdy nezazní `G dur` (světle modrá), ač by to chromatická stupnice umožňovala.

Ve výše uvedené práci se nacházela verze algoritmu, označená jako *harmonizovaná*. Zdá se být překvapením, že v této verzi k žádné reálné harmonizaci hudebních linek nedocházelo. Akord byl vybrán náhodně ze 4 akordů, určených z celkových vlastností obrázku. K jednotlivým tónům melodie měl tedy pouze jistý statistický vztah. Navíc díky náhodné distribuci akordů mezi sektory dochází pokaždé k jinému výstupu programu při stejných datech, což relaci mezi obrázkem a skladbou zbytečně oddaluje.

V projektu *Generate expressive music* tóny akordu taktéž vychází z převládající barvy segmentu. Opět je zde zanesena náhodnost, tedy stisknutí stejného místa na tabulce, stejnou silnou a stejně

[6]Některých, protože tón `E` a `F` jsou v chromatické řadě vedle sebe, a stejně tak i jejich příslušné barvy v chromatickém kruhu. Mezi sousedními barvami velký kontrast nebývá, protože přechody v chromatickém kruhu jsou vcelku plynulé. U nesouladných tónů `C` (červená) a `F#` (zelená) vysoký kontrast s nesouladností odpovídá.

velkou plochou, nevygeneruje pokaždé stejný počet tónů v akordu. Protože projekt má představovat hudební nástroj, lze předpokládat, že tím autor zamýšlel nějakou jeho speciální funkci. Například mohl chtít simulovat nesprávné uchopení strun kytary na pražcích. Když pražce nejsou zmáčknuty správně, tóny strun také nezní. To však ve své práci nezmiňuje.

Práce *Pixelsynth* pojem akordu a skupiny tónů nerozlišuje. Celá skladba je tak tvořena polyfonními hudebními linkami. To je velká škoda, protože mezi nekontrolovanou polyfonnií často vznikají dissonance.

Změny ve funkční harmonii

Práce *Generate expressive music* pro každý sektor tvoří jeden akord a přehrává je v pořadí vybrané uživatelem. Tedy funkční harmonie závisí částečně na uživatelském vstupu. To je podobné, jako bychom uživateli dovolili přehrávat píseň pozpátku či na přeskáčku. Co se však liší u jednotlivých algoritmů, je způsob výběru akordů. Přímá verze určuje akord v závislosti na nejvíce dominantní barvě v sektoru, ale harmonizovaná určí čtyři nejvíce dominantní z celého obrázku, a následně je semi-randomizovaně distribuuje do jednotlivých sektorů. Informace o jakémkoliv směru harmonie je tedy ztracena. Lze proto říci, že přímá verze daleko lépe reflektuje původní harmonickou relaci jednotlivých barev v obrázku, protože pořadí a obsah akordů vychází z jednotlivých oblastí zdrojového obrázku.

U *Sonification of Fish Movement* je změna funkční harmonie řešena ještě lépe, a to střídáním dvou mřížek s navzájem opačnými harmonickými funkcemi akordů dle směru pohybu ryb. Logicky tak reflektuje chování ryb, hlavních objektů ve videu. Otočení ryby nebo hejna do opačného směru způsobí změnu harmonie. Vzhledem k tomu, že funkční harmonie v hudbě reprezentuje jakýsi pohyb[7] a jeho směr, je tato relace téměř dokonalá.

Vzhledem k účelu projektu *Generate expressive music*, se nelze podivovat, že zde žádná souvislost funkční harmonie s obrázkem není. Uživatel volí pořadí akordů v závislosti na svém výběru oblasti malby. U práce *Pixelsynth* jsou jednotlivé akordy definovány všemi naráz znějícími tóny. To, jaké tóny zní, závisí na počtu not zvolených uživatelem a na jasu obrázku napříč aktuálně přehrávaným sloupcem. Pokud je řeč o generované funkční harmonii, v analogii tohoto projektu je myšlen kontrast mezi jednotlivými vertikálními oblastmi obrázku, tedy vertikální kontrast. To je podobné jako u přímé verze algoritmu v projektu *Cross-Domain Analogy*.

Struktura písně

Pouze jeden z projektů řeší strukturu vytvořené písně, a to *Cross-Domain Analogy*. Autorky se zaměřují na rockový žánr, který má dle jejich podání jistou strukturu. To zahrnuje omezení rozsahu tempa, počtu akordů[8], bicích rytmů[8], taktové předznamenání či zařazení konkrétních nástrojových linek. Vzhledem k volbě rockové hudby je to dobrý nápad.

U projektu zaměřeného na vznik hudebního nástroje (*Generate expressive music*) je pochopitelné, že nevyužívá členění generového hudebního artefaktu, protože by to v jeho případě nedávalo velký význam. Je však velká škoda, že ostatní projekty si počínají podobně. Alespoň základní stavební prvky jako *intro*, *verse*, *refrén*, *bridge*, *outro* by mohly být využity.

[7] Funkční harmonie vyjadřuje snahu akordu po pohybu, jeho pohybovou energii. Tóny dominantních akordů mají tendenci stoupat vzhůru (dominanta) nebo klesat (subdominanta) k tónice, zatímco pokud po těchto akordech zahrajeme tóniku, dojde k uklidnění [2].

[8] v harmonizované verzi algoritmu

5.1.1 Hudební nástroje

Nebylo by písně a nebylo by hudby, kdyby je nebylo na co hrát. Řeč je o hudebních nástrojích, které svým zvukem dávají notovému zápisu tu správnou barvu. Volba hudebních nástrojů je velmi důležitá pro docílení té správné atmosféry písně.

Kvůli použití nevhodných hudebních nástrojů v projektu *Cross-Domain Analogy* posluchači dost dobře nerozpoznali zamýšlený hudební žánr prezentovaný ve skladbách. Práce *Pixelsynth* se vzhledem k metodě sonifikace žádné klasické hudební nástroje napodobit nesnaží. Zato používá principů aditivní syntézy, za účelem vytvoření hudebních nástrojů úplně nových. Barva hudebních linek tak závisí čistě na obrázku, což je určitě plus z hlediska maximálního využití zdrojů. Bohužel však nepřistupuje k tvorbě spektra tónu, a tedy barvy zvuku, nijak systematicky.

Projekt *Sonification of Fish Movement* generuje `MIDI`, které při přehrávání dovoluje zvolit hudební nástroj dle volby uživatele. To se zdá být výhodou v momentě, kdy uživatel chce změnit hudební nástroj skladby. Tato volba však bohužel nemá žádný vztah k obrázku. V projektu *Melobytes* je použití hudebních nástrojů velmi pestré, což je velké plus, protože tak skladba dostává větší hloubku.

5.1.2 Vstup

Většina zkoumaných projektů používala pro kontakt s uživatelem klávesnici a myš. Jednak pro určení vstupu, jednak pro úpravu parametrů. Výrazná většina lidí v naší době je již na tento standard zvyklá a používání aplikace jim to nijak nebrání. Projekt *Expressive music generation* však přinesl něco, co ostatním projektům chybělo – hmatatelnost, tzv. *tangible interface*. Tím docílil mnohem přirozenější interakce se zařízením, protože používání hmatových gest k utváření reality je pro člověka velice intuitivní. Každá výhoda však bohužel v praxi bývá vykoupena nějakou nevýhodou. U tohoto zařízení je to horší přenositelnost v porovnání s ostatními projekty, které jsou zaměřeny spíše softwarově a ke svému užívání potřebují pouze standardní hardware.

U některých projektů (*Sonification of Fish Movement*, *Cross-Domain Analogy*) nebylo možné parametry měnit pomocí uživatelského rozhraní. To se zdá být pro počítačové laiky velkou zátěží. Stejně tak příliš mnoho nastavení (například v programu Autodesk 3ds Max) může na uživatele bez řádného návodu působit příliš chaoticky a následně ho odradit od jakékoliv interakce s aplikací. Velmi dobře vyřešené uživatelské rozhraní mají aplikace *Melobytes* a *Pixelsynth*. Je přehledné a uživelsky nenáročné.

5.1.3 Výstup

Výsledky projektu *Melobytes* dobře poukázaly na skutečnost, že způsob, jakým generovanou hudbu přehráváme, může zásadně ovlivnit kvalitu výsledného hudebního díla. Ať už je daný artefakt uložen ve zvukové podobě nebo pomocí notového zápisu, je důležité zajistit, aby každý uživatel obdržel výstup ve stejné formě, případně byl o rozdílech mezi jednotlivými formáty poučen.[9]

U mnoha projektů bylo využito univerzálního formátu `MIDI` (*Cross-Domain Analogy*, *Sonification of Fish Movement*, *Melobytes*) což pro je generování hudby postačující. Projekt *Pixelsynth* byl založený na principu aditivní syntézy.

Uživatel by měl dále možnost si vygenerovanou skladbu přehrát a stáhnout opakovaně bez omezujících limitů. Tato možnost nebyla dostupná nejen u projektu *Pixelsynth*, ale i *Generate expressive music*. U projektu *Melobytes* byl počet stažení `MIDI` formátu u neplacené verze značně omezen.

[9] Např. chybějící hudební nástroje při exportu do některých formátů.

Za zmínku stojí i generování hudební notace v jedné z aplikací *Melobytes*. Z výsledku však nebylo jasné, jestli je generována melodie zpěvu nebo jiné polyfonické hudební linky, což mohlo být pro některé uživatele zmatečné.

5.2 Vyhodnocení

V přechozí části byly podrobně srovnány metody souvisejících projektů. Tato kapitola nabízí pohled na věc z hlediska samotných projektů. V podobě bodového shrnutí výhod a nevýhod vykreslí zásadní plusy a mínusy jednotlivých aplikací.

Cross-Domain Analogy

Výhody	Nevýhody
⊕ zachování relací mezi jednotlivými vlastnostmi barvy v generovaných písních	⊖ některé relace jsou utvořeny jen na základě rozhodnutí autorek
⊕ rozlišuje melodii a doprovod	⊖ nevhodná volba nástrojů vzhledem k žánru
⊕ tempo v závislosti na obrázku	⊖ pouze 4 tóny v taktu
⊕ struktura písně	⊖ harmonizace probíhá pouze statisticky
⊕ harmonie odráží kontrast některých barev v sektoru	⊖ mezi melodií a akordem vzniká nesoulad
⊕ kvalita akordu založená na relaci ze studie	⊖ nedeterministický výstup
⊕ využívá informaci o barvě	⊖ žádné pomlky
⊕ obsahuje bicí linku	⊖ některé druhy akordů vůbec nevzniknou
	⊖ ztrácí se informace o pořadí not

Generate expressive music

Výhody	Nevýhody
⊕ využívá informaci o barvě	⊖ špatná přenositelnost
⊕ relace barva → výška tónu podpořena studií	⊖ výsledek nejde uložit
⊕ intuitivní uživatelské ovládání	
⊕ přirozený vztah jednotky a celku	
⊕ délka tónu dle rozlohy	
⊕ intuitivní pojetí dynamiky	

Sonification of Fish Movement

Výhody

- ⊕ mapování dynamiky noty `MIDI` odpovídá velikosti pohybu ryby
- ⊕ relace harmonie ryb → harmonie tónů
- ⊕ přenositelnost
- ⊕ vztah okolí a celku detekuje AI

Nevýhody

- ⊖ nastavitelnost mřížky tvoří umělé hranice mezi obrázkem a hudbou
- ⊖ konstantní délka tónu
- ⊖ tempo v závislosti na vstupu uživatele
- ⊖ chybí struktura písně
- ⊖ chybí GUI
- ⊖ ztracená informace o barevnosti

Melobytes

Výhody

- ⊕ přívětivé GUI
- ⊕ generování hudební notace
- ⊕ přenositelnost
- ⊕ pestrost volby hudebních nástrojů

Nevýhody

- ⊖ v exportu do `MP3` formátu chybí zvuk některých nástrojů
- ⊖ není jisté, k čemu hudební notace patří
- ⊖ tempo v závislosti na vstupu uživatele
- ⊖ počet stáhnutí výsledku je omezen

Pixelsynth

Výhody

- ⊕ přívětivé GUI
- ⊕ možnost zvolení směru
- ⊕ přenositelnost
- ⊕ relace pohyb → funkční harmonie
- ⊕ vstup obrázek → zvukové spektrum tónu
- ⊕ parametrizace hudebních nástrojů

Nevýhody

- ⊖ není možné uložit výsledek
- ⊖ malá souvislost pixelu s okolím
- ⊖ generovaná hudba je příliš dlouhá
- ⊖ konstantní délka tónu
- ⊖ tempo v závislosti na vstupu uživatele
- ⊖ chybí struktura písně
- ⊖ chybí systematičnost hudebních nástrojů
- ⊖ ztracená informace o barevnosti

Kapitola 6

Mapování modalit

Pro generování hudby z obrázku je důležité určit vzdájemné mapování jednotlivých modalit. Cest je mnoho. Protože však není možné vyzkoušet všechny, budou v této kapitole rozebrány výhody a nevýhody některých mapování. Na situaci bude pohlíženo ze zvukové domény, tedy z pohledu výsledku. To znamená, že mapování bude stanoveno dle vlastností hudebního artefaktu. Na závěr bude vyhodnoceno, který postup mapování pro danou vlastnost bude použit.

Tvorba melodie
- # pozice v chromatické řadě
- # oktáva tónu
- # stupnice a tónina
- # délka tónu
- # hlasitost
- # pomlky

Tvorba doprovodu
- # rytmus
- # tvorba akordu
- # změny ve funkční harmonii

Obecné vlastnosti písně
- # pořadí not
- # tempo
- # struktura písně
- # hudební nástroje

6.1 Vedení hudebních linek

Pro mapování modalit mezi obrazovou doménou a zvukovou doménou je nejdříve potřeba určit kolik hudebních linek bude skladba mít. Závisí na tom totiž vertikální harmonie skladby. Také je potřeba určit, jakým způsobem budou hudební linky vedeny. V mapování ostatních vlastností z toho lze následně vycházet.

Jedna melodie / Pouze akordy

Pouhá melodie by mohla znít jako něco nedokončeného, jako objekt na průhledném pozadí. Byla by také ochuzena o harmonii, důležitou emoční složku, působící na posluchače. Podobně je to i se samostatně hrajícími akordy. Akordy mají být k něčemu doprovodné, mohou sice svými harmonickými funkcemi předat jistý příběh a pohybovou energii, ale po čase to může působit nudně.

Výhody	Nevýhody
⊕ Jednoduché na implementaci	⊖ Příliš nudné

Více melodií (a akordový doprovod)

Pokud existuje více nezávislých hlasů, jde o polyfonii, kde harmonie vzniká náhodnými souzvuky. Přesto je potřeba dohlížet na nekřížení hlasů a případný doprovod harmonizovat s oběma hlasy, což může klást nároky nejen na časovou náročnost nebo výpočetní výkon, ale také na podrobnější znalost a praxi v hudební teorii. Na druhou stranu, generovaný artefakt tím docílí větší barevnosti.

Výhody	Nevýhody
⊕ Větší barevnost	⊖ Náročnost na výkon/čas
⊕ Stabilita	⊖ Podrobnější znalost hudební teorie

Hlavní melodie a akordový doprovod

Melodie a akordový doprovod tvoří základ pro vedení čtyřhlasu, jež se v hudbě užívá. Tento způsob tvoří vertikální strukturu písně a přispívá tak k větší stabilitě generovaného artefaktu. Zároveň je třeba dbát na nekřížení hlasů. Není to však tolik náročné jako v polyfonii.

Tón v melodii je doprovázen akordem, přičemž akord má v základu stejnou oktávu jako tón, s duplikovaným základním tónem v podobě basu. Pokud je tón v akordu obsažen v rámci definice obsažen, doprovod ho nehraje.

Výhody	Nevýhody
⊕ Vertikální struktura písně	⊖ Potřeba hlídat křížení hlasů
⊕ Stabilita	

6.2 Tvorba melodie

Melodie představuje v hudebním artefaktu hlavní melodickou linku a postupně provází posluchače celou hudební skladbou. Nedílnou součástí melodie je rytmus.

6.2.1 Zařazení tónu do chromatické řady

Newton, Sir James Jeans a spektrální barvy

Isaac Newton dle *J. L. Caivana* [17] přiřadil sedmi základním spektrálním barvám (červená, oranžová, žlutá, zelená, tyrkysová, indigo, fialová) tóny v diatonické stupnici – tabulka 6.1. Jednalo se o stupnici mollovou přirozenou, se zvýšeným `VI.` stupněm.

Newton tak učinil na základě zjištění, že jednotlivé vlnové délky spektrálních barev mají k vlnové délce červené barvy stejný poměr, jako je převrácený zlomek poměru frekvencí tónů stupnice k tónu základnímu. Pro příklad lze uvést, že tón `C` má k tónu `D` frekvenční poměr $^9/_8$ a poměr vlnových délek `červené` a `oranžové` barvy je $^8/_9$.

Protože se však se stupnicí `moll přirozenou se zvýšeným VI. stupněm` pracuje trochu hůře[1], zdálo se vhodné tento vztah nějakým způsobem transformovat, aby výsledkem byla stupnice `durová`.

[1] tvorba harmonie, navíc stupnice není v dnešní hudbě tolik používána

Tabulka 6.1 Mapování spekt. barev na přiroz. moll. stupnici se zvýšeným `VI.` stupněm (Newton)

Stupeň	Vlnová délka	Barva	Tón v `C moll*`	Poměr frekvencí tónu vůči zákl. tónu
I.	760–675 nm	Červená	`C`	1:1
II.	675–633 nm	Oranžová	`D`	9:8
III.	633–570 nm	Žlutá	`D#`	6:5
IV.	570–507 nm	Zelená	`F`	4:3
V.	507–456 nm	Tyrkysová	`G`	3:2
VI.	456–405 nm	Indigo	`A`	5:3
VII.	428–380 nm	Fialová	`A#`	16:9

Toho se chopil údajný *Sir James Jeans*, které posléze citoval *Maitland Graves* a následně i *J. L. Caivano* ve svém článku. *Sir James Jeans* posunul hranice vlnových délek barev tak, aby odpovídaly převrácenému poměru frekvencí tónů v durové stupnici vůči tónu základnímu – tabulka 6.2

Tabulka 6.2 Mapování spektrálních barev na durovou stupnici *(Sir James Jeans)*

Stupeň	Vlnová délka	Barva	Tón v `C dur`	Poměr frekvencí tónu vůči zákl. tónu
I.	760–675 nm	Červená	`C`	1:1
II.	675–633 nm	Oranžová	`D`	9:8
III.	633-570 nm	Žlutá	`E` *(má být `D#`)*	6:5
IV.	570-507 nm	Zelená	`F`	4:3
V.	507-456 nm	Tyrkysová	`G`	3:2
VI.	456–405 nm	Indigo	`A`	5:3
VII.	405–380 nm	Fialová	`H`	15:8

U jedné barvy/tónu však poměr dle tabulky neseděl. Jednalo se o tón `E`, poměr uvedený v tabulce říkal, že tón `E` je k tónu `C` ve stupnici `C dur` v poměru $^6/_5$. To však není pravda, tón `E` je vůči tónu `C` v poměru $^5/_4$. V poměru $^6/_5$ je tón `D#`. Z tohoto důvodu byla tabulka upravena do podoby zobrazené na schématu 6.3.

Tabulka 6.3 Mapování spektrálních barev na durovou stupnici *(konečná verze)*

Stupeň	Vlnová délka	Barva	Tón v `C dur`	Poměr frekvencí tónu vůči zákl. tónu
I.	760–675 nm	Červená	`C`	1:1
II.	675–608 nm	Oranžová	`D`	9:8
III.	608-570 nm	Žlutá	`E`	5:4
IV.	570-507 nm	Zelená	`F`	4:3
V.	507-456 nm	Tyrkysová	`G`	3:2
VI.	456–405 nm	Indigo	`A`	5:3
VII.	405–380 nm	Fialová	`H`	15:8

Lze si všimnout, že se tabulky 6.1 a 6.3 se liší právě o dva poměry (tóny). Posunout tak stačí dvě hranice barev, a to *žluté* (tón `D#` → `E`) a `H` (tón `A#` → `H`). Když bude zachován stejný princip, tak to znamená, že u *žluté* se poměr vlnových délek barev se změnil z `5:6` na `4:5` a u *fialové* z `9:16` na `8:15`. Protože nový z poměrů je vždy menší než předchozí, bude se počáteční hranice barvy posunovat doleva. Její konečná hranice však zůstane stejná. To znamená, že dojde k mírnému zvětšení rozsahu odstínů, které lidé označují jako *žlutá* a *fialová*, a k mírnému zmenšení rozsahu odstínů, které jsou označovány jako *oranžová* a *indigo*.

Dle poměru zmenšení poměrů vlnových délek bude rozsah oranžové barvy zmenšen na 96 % a barvy tmavě modré (indigo) na 95 %. Například v barevném modelu `HSV` lze vidět, že je tento rozdíl prakticky zanedbatelný. Oranžová má rozsah `15`–`45` a tmavě modrá `195`–`255`. Škála odstínů oranžové se tak posune na `15`–`44` a tmavě modré na `195`–`252`

Při použití výše zmíněné relace se tak převádí množina spektrálních barev, uspořádaná dle svých vlnových délek, na uspořádanou množinu tónů dle výšky. Je známo, že soubor tónů ve stupnici, uspořádaný dle určitých pravidel, tvoří tóninu. Tato tónina barev má tak hudebně popsatelné vlastnosti, které jsou určeny vztahy nejen mezi jednotlivými tóny (barvami), ale i mezi akordy postavenými na jednotlivých stupních (např. harmonické funkce) [1].

Tato vlastnost zároveň implikuje omezení skladby na jednu tóninu, což se vzhledem k rozsahu práce zdá být postačující. Z hudební teorie víme, že pokud je použito temperovaného ladění, můžeme libovolně jednotlivé tóniny mezi sebou převádět. Tónina tak může být dalším proměnnou ve vztahu k vstupním datům.

Velku nevýhodou je to, že informace o velikosti jednotlivých příspěvků spektrálních barev do zaznamenáného pixelu je v digitální podobě již ztracena. Je tedy potřeba se řídit převažujícím barevným odstínem, který je pomocí složky *hue* vyjádřen například v modelech `HSV` či `HSL`, a následně dle vlnové délky spočítat ke které spektrální barvě odstín patří, což může vést k jistým nepřesnostem.

Ve vnímání barev se člověk od člověka liší[2], a i díky subjektivní interpretaci obrázku býváme kolikrát napáleni svým vlastním mozkem[3]. Vzhledem k tomu, že vnímání hudby jako umění je také velmi subjektivní, nemusí výše uvedené nepřesnosti pro aplikaci představovat až tak velký problém.

Co však problémem zůstává, je *purpurová barva*. Ve světelném spektru se nenachází, protože spojuje dvě spektrální barvy, které jsou na jeho opačném konci. Řešením je rozdělit škálu odstínů purpurové barvy a přiřadit ji ke zmíněným barvám.

Výhody	Nevýhody
⊕ jednoduchá transpozice do jiné tóniny	⊖ nepřesnost při hledání spektrální barvy
⊕ hudebně využitelné vlastnosti tóniny	⊖ omezení skladby na jednu tóninu
⊕ poskytuje tóninu jako proměnnou	⊖ problém purpurové barvy

Spojení přes chromatický kruh

V předchozích projektech (*Cross-Domain Analogy*) bylo použitu vztahu tónů v chromatické řadě a barev v chromatickém kruhu. Tedy, využití všech známých půltónů v jedné skladbě.

Protože chromatická stupnice není diatonická, mají její tóny mezi sebou stejný interval. Analogicky je uspořádán chromatický kruh. Přednost této možnosti se skrývá zejména v jemnějším rozlišení barevných odstínů. Taktéž jako předchozí možnost, poskytuje při temperovaném ladění možnost transpozice dle libovolného základního tónu.

Využití celé chromatické řady může na první pohled působit jako pozitivum, protože program dostane k ruce „pestřejší paletu“, ale z hudebního hlediska je při používání chromatické stupnice potřeba sledovat mnoho věcí související s harmonií [34] (příčnost, paralelní kvinty a oktávy, nezpěvné kroky), u kterých je potřeba nastudovat podrobněji hudební teorii.

[2] Podivuhodným příkladem může být fotografie šatů z roku 2015 [32], kdy se společnost rozdělila na základě otázky, jakou barvu šaty mají.

[3] Za zmínku stojí obrázek šachovnice Edwarda H. Adelsona, kdy se dvě stejně obarvená políčka jeví jako barevné protiklady [33].

Výhody	Nevýhody
⊕ Využití všech tónů ve stupnici	⊖ Nutnost znát podrobněji hudební teorii
⊕ Jemnější rozlišení barev	
⊕ Poskytuje tóninu jako proměnnou	

Výsledné mapování

Díky zajímavému spojení poměru frekvencí tónů ve stupnici a vlnových délek barev a jednoduše použitelným hudebním vlastnostem durové tóniny, bude v této práci využito prvního způsobu mapování, a to mapování *spektrálních barev na durovou stupnici*.

6.2.2 Oktáva tónu

Dle světlosti – absolutně

Tento postup byl využit již v projektu *Generate expressive music*. Nebylo zde však zmíněno, proč. Ve studiích pozorované subjekty často spojovaly zvyšující se jas barvy s hlasitostí. Studie z roku 1974 od L. E. Markse [35] však ukazuje, že tento vztah není tak jednoznačný, jak by se mohlo zdát. Zvýšení jasu totiž bývá spojováno nejen s hlasitostí, ale také se zvyšující se výškou tónu.

Pokud se na tóny díváme jako na jednotlivé barvy, jak bylo uvedeno výše, můžeme po přiřazení tónu do chromatické řady, na základě spektrální barvy, zařadit tón do oktávy dle jeho jasu.

Nejlépe se pro tento moment hodí barevné prostory, které oddělují složku jasu od barvy a zároveň berou v úvahu různé vnímání jasu u jednotlivých barev, jako to dělá lidské oko. Použitelným je například barevný prostor `L*a*b`.

Výhody	Nevýhody
⊕ Jednoduché na výpočet	⊖ Příliš velké skoky
⊕ Odráží výsledky studie	

Dle světlosti – relativně

Pouhé mapování světlosti na oktávu by však vyvolávalo v případě velmi kontrastních barev vedle sebe velké výškové skoky, což není v hudbě obvyklé. A obvyklé to není ani při vnímání obrazu. Mozek posusuzuje jednotlivé barvy vedle sebe ne absolutně, ale relativně, navíc v sémantickém vztahu k okolí. Optické iluze jako *Checker Shadow Illusion* [33] jsou toho důkazem. Dále je oko zvyklé přizpůsobovat se světelným podmínkám (adaptace oka) [9].

Výchozí bod, o který se bude počáteční tón opírat, pak může být určen vzhledem k celému obrázku, například průměrným jasem. To vyřeší situace, kdy by jinak světlý obrázek měl v počátečním regionu příliš světlou barvu a celkový dojem by tak byl úplně jiný. Navíc tento způsob využívá další faktor obrázku a tvoří tak propojení jednotky (jednotlivá oktáva, jeden barevný region) a celku (průměrná svítivost celého obrázku).

Zařazení do oktávy tak nebude probíhat absolutně, nýbrž relativně. To znamená, že namísto přímého mapování jasu na určitou oktávu se nejdříve algoritmus podívá do blízkého okolí a porovná přírůstek jasu a dle toho určí oktávu.

Výhody	Nevýhody
⊕ Jednoduché na výpočet	
⊕ Odráží výsledky studie	
⊕ Přechod mezi tóny je plynulejší	
⊕ Reflektuje adaptaci lidského oka na světlo	
⊕ Vztah jednotky k celku	

Výsledné mapování

Jak je vidět z výhod a nevýhod, *mapování dle relativního jasu* má větší vztah k přirozenosti vnímání lidského oka. Navíc uplatňuje další faktor obrázku, a to jeho průměrnou světlost (v podání `L*a*b*` průměrnou svítivost). Z tohoto důvodu bude v aplikaci použito.

6.2.3 Stupnice a tónina

V hudební kultuře existuje množství stupnic a jejich variací. Kvůli rozsahu této práce není možno zacházet do přílišných detailů, proto bude pro zjednodušení použito durových stupnic, které jsou diatonické a používány jsou především v evropské hudbě. Každá durová stupnice má svůj základní tón, od kterého je tvořena. Když využijeme celou chromatickou řadu, získáme 12 durových stupnic. Pokud je použito temperovaného ladění, jde každá píseň zahrát v jakékoliv tónině s využitím transpozice.

Dle průměrné světlosti obrázku

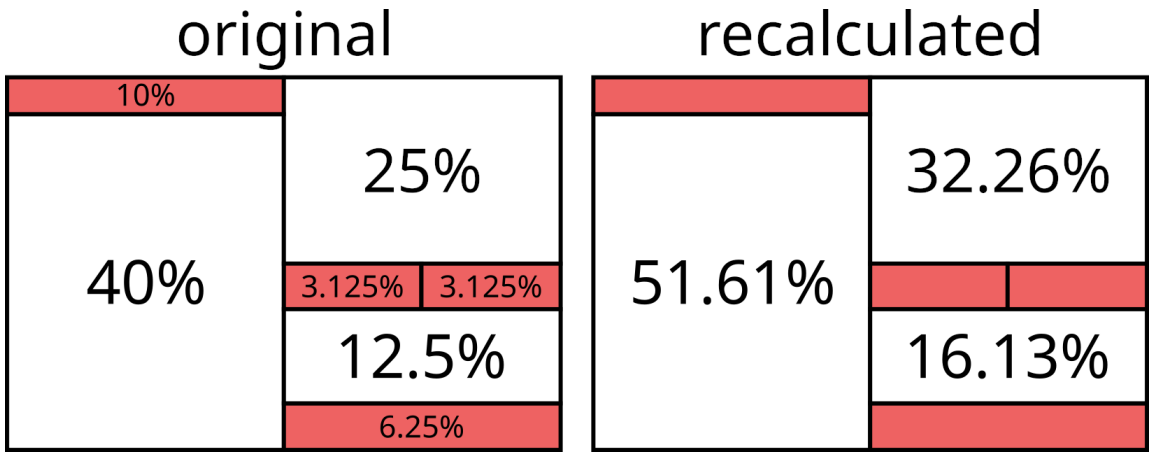
Jak již bylo poznamenáno u kapitoly `Oktáva tónu`, dle Markse [35] lidé spojují vyšší jas obrázku s vyššími výškami tónů. Základní tón tóniny, společně s oktávou, udává reálnou výšku hrajících tónů. Je tedy na místě vztáhnout základní tón tóniny k průměrné světlosti obrázku, a to dle umístění základního tónu v chromatické stupnici. Na druhou stranu by mohlo dojít k situaci, kdy všechny písně s vyšším základním tónem, budou hrány pouze ve vyšších oktávách (počáteční oktáva je vztažena k průměrné světlosti obrázku) a obráceně. Jak již bylo zmíněno, každá píseň se za podmínky temperovaného ladění dá převést do libovolné tóniny, a protože jsou v práci uvažovány pouze tóniny *durové*, nebude mít změna tóniny takový efekt jako měla např. v projektu *Cross-Domain Analogy*.

Výhody	Nevýhody
⊕ podloženo studií	⊖ vztah oktávy a tóniny
⊕ reflektuje vlastnosti obrázku	⊖ na atmosféru nemá velký efekt

Dle volby uživatele

Uživatel bude mít v této možnosti výběr základního tónu pevně v rukou, podobně jako u projektu *Pixelsynth*. Na druhou stranu nebude volba tóniny příliš reflektovat vlastnosti obrázku.

Výhody	Nevýhody
⊕ uživatel si může vybírat	⊖ nereflektuje vlastnosti obrázku
⊕ změna tóniny není tak důležitá	⊖



Obrázek 6.1 Ukázka přepočítání procent dle rozlohy segmentu. Nejmenší délka noty je v tomto příkladu šestnáctinová, což odpovídá 12.5 % rozlohy segmentu.

Výsledné mapování

Vzhledem k tomu, že změna základního tónu tóniny neovlivní atmosféru výsledné skladby tolik, jako by ji měnila, kdyby se měnila z durové na mollovou, bude tato volba ponechána na uživateli. Prostor pro tvorbu a implementaci bude dán zajímavějším relacím.

6.2.4 Délka tónu

Dle rozlohy barevného regionu

Obrázek je rozdělen obdelníkovou mřížkou na díly o stejné velikosti, segmenty. Každý z nich pak představuje jeden takt. V segmentu existují barevné regiony, které reprezentují hudební tóny. Jejich délka odpovídá poměrné rozloze regionu v segmentu. Podobný postup byl využit v práci *Cross-Domain Analogy*.

Vztah velikosti barevného regionu a délky tónu byl také podpořen prací L. J. Caivana [17], který poznamenává, že stejně jako by vjem barvy neexistoval bez rozlohy barevné plochy v prostoru, nemůže tón znít bez své délky. Bez těchto atributů by nebylo možné rozeznat další vlastnosti uvedených elementů (např. frekvenci tónu a světlost barvy).

Ve zmíněném projektu *Cross-Domain Analogy* byly vybrány pouze čtyři nejvýznamněji zastoupené barvy v segmentu, což neumožňuje mít více jak čtyři tóny v taktu. Navíc se ztrácí informace o jejich vzájemné poloze, tedy o případném uspořádání not v taktu.

Pro výpočet délky tónů by se tak mělo pracovat s většinou dostupných barevných regionů, které segment obsahuje. Dále oddělené regiony stejné barvy by neměly splývat v jeden, protože se tak vyloučí tóny stejné výšky v taktu.

V hudbě rozlišujeme několik hlavních[4] délek tónů: *nota celá, nota půlová, nota čtvrťová, nota osminová, nota šestnáctinová, nota dvaatřicetinová, …* Každá z nich svým jménem označuje, jakou část $\frac{4}{4}$ taktu zabírá. Dané části lze převést na procenta a z nich vycházet při přepočítávání poměrné části rozlohy segmentu na délku tónu.

[4]Pokud nejsou uvažovány prodloužené délky.

Tímto vzniká problém příliš malých délek. Nota dvaatřicetinová zabírá 1/32 délky taktu, tedy 3.125 % rozlohy segmentu. Co si však počnou regiony menší? Nota čtyřiašedesátinová se v hudbě příliš nepoužívá, navíc tímto způsobem by šlo noty dělit téměř do nekonečna. Proto je nutno konstantou vymezit minimální délku noty. Tímto se také vyřeší problém prodlužování not. Rozloha regionu, která by procentuálně nezapadla ani do jedné kolonky, bude dělena nejmenší rozlohou v regionu, čímž se získá délka společně s prodloužením.

Regiony, které budou příliš malé, budou ze sektoru vyřazeny. To odpovídá analogii, že příliš malým věcem oko nevěnuje pozornost. Pouze regiony vyřadit, by při určitých okolnostech[5] zanechalo příliš prázdného místa v taktu. Noty tedy budou generovány z již protříděných regionů, ve vzdájemném poměru. Pokud přeci jen dojde k nevyužití místa regionu[6] a délka taktu v notách nebude dávat 100 %, takt bude doplněn pomlkou. To zaručí, se že takty nebudou vzájemně mísit, a nebude tak narušena vazba mezi segmentem a taktem. Popsanou situaci přibližuje obrázek 6.1. Výsledná hudební notace je zachycena na obrázku 6.2.

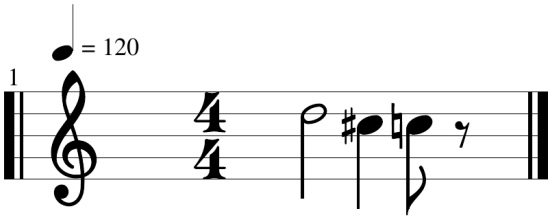
Nevýhodou této metody je to, že mřížka rozdělující segmenty je pevná, a tedy pokud dojde k nerovnoměrnému rozdělení obrázku, poslední segmenty budou kratší než ostatní. To není reflektováno ve délce taktu. Dále, šířka a výška segmentu nevychází z ničeho konkrétního a bude považována za proměnnou, což je umělým prvkem ve vztahu obrázku a generovaného hudebního artefaktu.

Výhody	Nevýhody
⊕ logický vztah mezi modalitami	⊖ poslední segmenty kratší než ostatní
⊕ vycházeno z vědecké publikace	⊖ velikost segmentů nezávisí na obrázku
⊕ tón může být v taktu obsažen víckrát	
⊕ více než čtyři tóny v taktu	

6.2.5 Hlasitost a pomlky

Jak se píše na webu *All About Jazz*, *„Ticho je barvou, pokud je využito“*. Upozorňuje, že světoví umělci využívají pauzy v hudbě stejně bravorně jako své techniky. Stejně jako umělec kreslí plátno barvou, i ticho se dá použít jako odstín scény, ale jen za předpokladu, že je s ním nakládáno efektivně. Hovoří se o tom, jak pauzy v lince jednoho hudebního nástroje nechávají lépe vyznít ostatní nástroje. *Janice Mason Steeves* na svém blogu[36] zmiňuje, že oči hledící na obraz by měly najít místo k odpočinku. Takzvané *velké tiché tvary*. Pak lépe vyniknou důležité objekty v obraze. To je podobná paralela k výše zmíněným poznámkám.

[5] Např. hodně malých regionů.
[6] Např. dělení délkou nejmenší noty nevyjde přesně.



Obrázek 6.2 Hudební notace (přesněji délky not) vzniklá z ukázkového segmentu.

Zajímavé je, že člověk nikdy nepocítí absolutní ticho[7], nemá tedy smysl za ticho považovat pouze nulové hodnoty jiné modality, ale jistý okruh nízkých hodnot, který můžeme následně převést na tiché. V analyzovaných projektech tomuto důležitému prvku přesto nikdo nevěnuje dostatečnou pozornost.

Nedostatečná saturace a světlost

V již zmíněné studii [35] od *L. E. Markse* spojovali lidé hlasitost hudebních tónů s jasem. Většinově byla nižší hlasitost zvuku přiřazována nižšímu jasu a naopak. Z toho lze vyvodit následující: Podobně jako nízká hlasitost zvuku znamená ticho, zanedbatelná intenzita světla představuje tmu. Žádný světelný zdroj, tedy klid. Tma je tedy analogií pro ticho. Pomlky by tak mohly být indikovány jako příliš tmavé regiony barev.

Ve studii se však našli i jednotlivci, kteří naopak prezentovali snižující se hlasitost jako zvyšující se jas. To plynule přechází k myšlence ohledně intensity hlasitosti jako o intenzitě (saturaci) barvy. Barvy v gradientu černé, přes šedou, až po bílou mají společný prvek, a to je nulová saturace v modelech `HSV`/`HSL`. Je jedno, jaký barevný odstín (hue) nastavíme, ani v jednom případě nebudou příslušet barvy[8] ze zmíněného gradientu k odstínu konkrétnímu. Zároveň se díky hodnotě světlosti[9] lze v gradientu posunovat.

To se zdá být analogií k tomu, že být hrán jakýkoliv tón na hudební nástroj, ale když není hrán dostatečně hlasitě, posluchač ho nerozezná. Tedy i barevné regiony s nízkou saturací by měly být počítány jako pomlky.

Výhody	Nevýhody
⊕ je vycházeno ze studie	
⊕ bere v úvahu více veličin	

6.3 Tvorba doprovodu

Tvorbou doprovodu se rozumí hudební linka, která doprovází hlavní melodii. Většinou je tvořen akordy. Na doprovod se lze dívat z hlediska vertikální harmonie (souladnost a nesouladnost), ale také z hlediska harmonie funkční (funkce akordů).

6.3.1 Rytmus

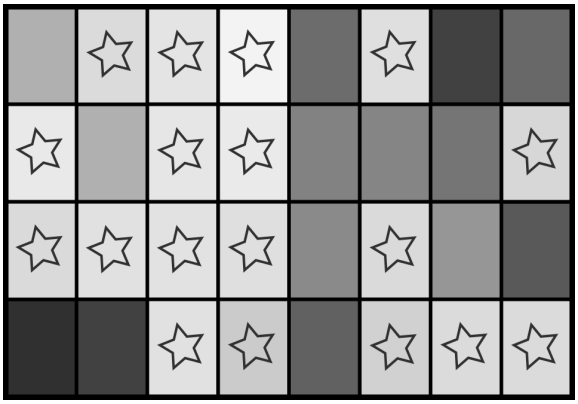
Rytmus souvisí s pomlkami, uvedenými v předchozí kapitole, velmi úzce. *Luděk Zenkl* ve své knize *ABC Hudební nauky* [1] poznamenává, že „Metrum a rytmus se v hudbě často doplňují, a proto se při jejich zkoumání často mluví o metrorytmických vztazích v hudbě.“

„Melodie bez rytmu neexistuje...“ říká v publikaci *ABC Hudební nauky* [1] *Luděk Zenkl*. Zatímco rytmus melodie je určen střídáním tónů a pomlk na základě barevných regionů v segmentu, pro akordy v doprovodu není žádný definován.

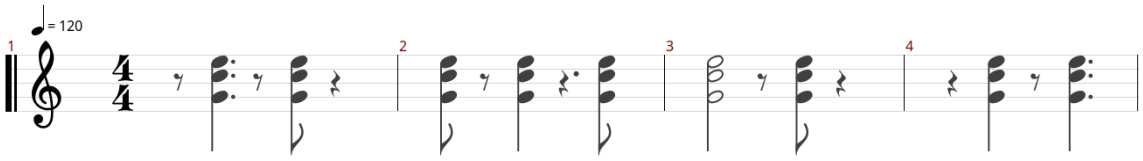
Světlost segmentů

Pokud jeden segment představuje takt v melodii, pak ho lze vnímat i jako jednotku obrázku. Podobně jako u oktávy tónu, i zde bude hrát roli svělost obrázku, jejíž průměr bude představovat

[7] Příkladem může být zvukotěsná komora. Po chvíli začne člověk slyšet i tlukot vlastního srdce.
[8] Černá, bílá i šedá jsou dle studie [37] také barvami.
[9] lightness v `HSL`, value v `HSV`



Obrázek 6.3 Ukázka generování rytmu – světlé a tmavé segmenty.



Obrázek 6.4 Ukázka generování rytmu – výsledek. Snímek obrazovky pochází z open-source editoru *TuxGuitar*.

jakýsi práh. Segmenty budou rozděleny na dvě skupiny, dle vztahu k tomuto prahu: segmenty světlejší než práh a segmenty tmavší než práh. Analogicky ke kapitole o hlasitosti představují tyto dvě třídy akordové pomlky a zahrané akordy. Otázkou zůstává, které jsou které.

Protože rytmus je klíčovou složkou nejen doprovodu, bylo by zajímavé vzít do úvahy obě možnosti a následně srovnávat výsledky. Příklad mapování lze vidět na obrázku 6.3. Označeny hvězdičkou jsou segmenty, které jsou světlejší než je zmíněný práh, a tedy jsou zdrojem not. Výsledek rytmu generovaného z výše zmíněného obrázku lze spatřit v hudební notaci na obrázku 6.4.

Výhody	Nevýhody
⊕ variace možností jako proměnné	
⊕ vztah k celému obrázku	

6.3.2 Tvorba akordu

Stejně jako melodie

Pokud by se generovaly akordy na základě stejného postupu jako v melodii (délka dle rozlohy, stupeň základního tónu dle spektrální barvy), pak by to znamenalo, že by jejich základní tón odpovídal právě hranému tónu v melodii. Tedy jejich vztah by byl vždy vztah konzonantní. Skladba by ale měnila akordy stejně často jako tóny melodie, což by mohlo působit nudně. Navíc se tento postup ve skladbách mnoho nevyužívá.

Výhody	Nevýhody
⊕ vždy souladné	⊖ mohlo by působit nudně a nezvykle

Změna akordu při pomlce

Aby nebyl akord tak často měněn, změna nastane, jen když v se v rytmickém předpisu vyskytne pomlka. Tím se oproti předchozímu postupu oživí střídání akordů a skladba nebude působit tolik nudně. Tento postup by však měl tu nevýhodu, že by výsledný hudební artefakt byl častokrát disharmonický, což by nemuselo působit na posluchače esteticky správně.

Za konsonantní jsou považovány pouze durový a mollový kvintakord (a jejich obraty sextakord a kvartsextakord), ostatní jsou považovány za disonantní [1]. Tedy pravděpodobnost, že tón bude znít disonantně[10] s právě hraným akordem, není malá.

Výhody	Nevýhody
⊕ vztah k obrázku	⊖ artefakt by často působil disharmonicky
⊕ není tolik nudné	

Harmonizace melodie

Akordy by byly generovány na základě tónu v melodii, jako u prvního postupu. Nejprve by však bylo zkontrolováno, zda tón není již obsažen v předchozím akordu. Pokud by tomu tak bylo, předchozí akord by byl prodloužen o délku tónu. To zaručuje méně časté střídání akordů, a přitom harmonickou konsonanci s tóny.

[10] Pojmy konsonance a disonance akordů se rozlišují zejména mimo hudební souvislost. Ve skladbách jsou často disonance hrány záměrně k vytvoření napětí a následně rozváděny zpět do konsonancí [1]. To však není tento případ, protože není řízeno jak bude disonance rozvedena.

Akordy budou pro zjednodušení postaveny na kvintakordech a septakordech, a to při inspiraci z tabulek na webu *Kurz harmonie bez not* [38]. V souladu s čtyřhlasou úpravou hudebních linek [2] by se základní tón akordu zdvojil a posunul o oktávu níže, čímž se k akordu přidá dojem basového prvku. Tón, na kterém byl akord postaven, by následně byl z akordu vyhozen (bude hrán nějaký čas v melodii). Je-li akord tvořen na bázi `VII.` stupně stupnice a jedná se o *septakord*, nic se nevynechává ani nezdvojuje (pravidla čtyřhlasé úpravy hlasů [2]).

Výhody	Nevýhody
⊕ souladné	
⊕ akordy nejsou střídány zbytečně často	

Výsledné mapování

Vzhledem k převažujícím plusům u mapování *harmonizace melodie* se bude práce ubírat touto cestou.

6.3.3 Změny ve funkční harmonii

Z hlediska funkční harmonie je na akord pohlíženo jako na nositele pohybové energie [2]. Tu určuje jeho harmonická funkce. Rozlišujeme tři základní harmonické funkce: `tóniku`, `dominantu` a `subdominantu`. Jednotlivé harmonické funkce bývají označovány i svým stupněm[11]. Zmíněné funkce se nacházejí postupně na `I.`, `V.` a `IV.` stupnice. Na ostatních stupních stupnice jsou vedlejší harmonické funkce.

Detekce pohybu v texturách

Jednou z možností, jak v malbě vzbudit pocit pohybu, je umístit tam úsečky, které nejsou na okraj obrázku kolmé. Horizontální a vertikální úsečky naopak budí pocit klidu a jistoty [39].

Jak už bylo zmíněno, v hudbě, přesněji v tonální harmonii, nalezneme podobnou analogii v harmonických funkcích akordů. Akord na `V.` stupni (dominanta) má velice silnou pohybovou energii a budí pocit neklidu, stejně jako šikmé úsečky. Subdominanta, akord postavený na `IV.` stupni stupnice, je podobně neklidný, ovšem v menší míře. Z analogie k úsečkám si jej tedy lze představit jako úsečku odkloněnou od roviny jen mírně. Tónika, hlavní kvintakord tóniny, pak svou paralelu nalezne právě v horizontálních a svislých čarách.

Tento postup tvoří most mezi uměleckým vyjádřením pohybu v obraze a pohybovou energií ve výkladech funkční harmonie, což je relace, která se zdá být intuitivní, má však tu nevýhodu, že pokud nejsou detekovány žádné šikmé čáry nebo je jich detekováno málo, hudba bude příliš nudná.

Výhody	Nevýhody
⊕ přenesení pohybu z obrazu do funkční harmonie	⊖ v případě obrázku bez šikmých čar nudné

Komplementárnost barev

A. Wells ve své práci *Music and Visual color: A proposed correlation* [40] zmiňuje podobnost dominantního, subdominantního a tónického akordu k systému komplementárních barev. Uvádí, že komplementární barvy v chromatickém kruhu vytvářejí kontrast stejný jako interval *tritonus*

[11] Stupeň označuje, na kterém stupni stupnice je akord postaven.

Obrázek 6.5 Mapování barev na harmonické funkce

vytváří v hudbě. Dále, že tento kontrast je zmenšený, pokud jsou vedle sebe postaveny barvy vedlejší k těmto komplementárním barvám.

Protože v práci je využito spektrálních barev, bude se aplikace výše uvedenou myšlenkou pouze inspirovat. Namísto chromatického kruhu se bude používat kruh, ve kterém je sedm spektrálních barev. Naproti každé barvě jsou dvě barvy. K tomu, aby se člověk dostal na každou z nich, potřebuje přejít přes dvě jiné barvy.

Vzhledem k tomu, že chromatický kruh vychází ze spektrálních barev, lze předpokládat, že na opačné straně kruhu se nachází nejvíce cizí barvy (neklid, velká pohybová síla), a naopak barvy blíže jsou původní barvě více blízké (klid). Na základě rodin harmonických funkcí akordů: *tónické, dominantní* a *subdominantní* popsaných na webu *Simplifying Theory* [41] bylo přiřazeno mapování přechodu mezi barvami dle jejich vzájemné pozice na harmonickou funkci. Na obrázku 6.5 jsou zobrazeny vztahy mezi harmonickými funkcemi a barvami.

Dominantní funkce (`V.`) a neúplná dominanta (`VII.`) v sobě mají největší neklid. Proto jsou nejdále. Trochu blíže jsou subdominanta (`IV.`) a střídavá dominanta (`IV.`). Poslední stupeň před tónikou (`I.`) tvoří dvojice vrchní medianty (`III.`) a spodní medianty (`VI.`). V této relaci jsou tak obsaženy všechny harmonické funkce, hudební skladba je tak pestrá co se funkční harmonie týče. Nevýhodou je, že opět zanedbává purpurovou barvu.

Výhody	Nevýhody
⊕ analogie ke komplementárnosti barev	⊖ purpurová barva zanedbána
⊕ využívá všechny harmonické funkce	

Výsledné mapování

Z výše uvedených postupů byl zvolen ten, který mapuje přechody mezi barevnými regiony na harmonické funkce. Výsledné mapování se tedy bude řídit podle principu *Komplementárnosti barev*, a to zejména pro svou pestrost.

6.4 Obecné vlastnosti písně

Výše již bylo určeno mapování pro prvky typické pro každou z hudebních linek. Na píseň, resp. hudební artefakt, je však nutné se dívat i z hlediska celku. To zahrnuje *pořadí not, tempo, strukturu písně* a *volbu hudebních nástrojů*.

6.4.1 Pořadí not

Náhodně

Jedna z nejjednodušších metod na implementaci je generované noty náhodně zamíchat, podobně jako to udělal projekt *Cross-Domain Analogy*. Tato volba však nejen, že je nedeterministická, a nezaručuje tak možnost vygenerovat totožný hudební artefakt, ani při zachování stejných vstupních parametrů, ale také vůbec neodráží vlastnosti obrázku.

Výhody	Nevýhody
⊕ jednoduché na implementaci	⊖ neodráží vlastnosti obrázku
	⊖ nedeterministické

Zleva doprava, Shora dolů

V evropské kultuře je naprosto přirozené sledovat věci na obraze zleva doprava a shora dolů, ať už jde o text [42] nebo uspořádání hodnot [43]. Ačkoliv tento postup může působit podivně vůči jiným kulturám, které to mají opačně, či dokonce směry čtení střídají, jak se jim hodí [44], tato práce staví na základech evropské hudby, a jak už to u umění bývá, počítá s tím, že pro ostatní kultury může být příležitostí k poznání něčeho nového. Z pohledu programátora není těžké takový algoritmus naimplementovat, stačí jednotlivé zdroje not seřadit podle jejich počátečních bodů v obraze.

Výhody	Nevýhody
⊕ přirozené pro evropskou kulturu	⊖ nemusí být přirozené pro jiné kultury
⊕ jednoduché na implementaci	

Výsledné mapování

Pořadí not v melodii i doprovodu bude řazeno dle výchozí pozice obrazových dat, které jsou zdrojem pro noty. Bude tedy kopírovat princip *Zleva doprava, Shora dolů*, a to hlavně pro převažující výhody oproti druhému postupu.

6.4.2 Tempo

Rychlost skladby, tempo, se obvykle určuje v jednotkách úderů za minutu (`BPM`). Jedná se o prvek, který do skladby přináší energii, a to hlavně ve spojení s kratšími délkami not.

Dle průměrné světlosti obrázku

Díky studiím *Schloss et al.* [45] a *Tsang et al.* [46], zabývajícími se spojením barvy a hudebních ukázek, bylo zaznamenáno přidružení světlejších barev k hudbě hrané v durových tóninách a rychlém tempu. Díky tomuto poznatku lze uvažovat o mapování rychlosti generované hudby[12] na průměrnou světlost obrázku.

Nejprve je nutno připomenout, jak člověk vnímá barvu. Světlo dopadne na objekt, který má určité vlastnosti. Objekt část světla pohltí, část světla nechá projít a část odrazí. Světlo odražené od předmětu zachytí oko pozorovatele. Dle vlnových délek záření určí jeho odstín. Intenzita má za následek světlost barvy.

[12] Aplikace počítá s omezením tónin pouze na durové.

Je známo, že oči člověka jsou silně adaptivní na změnu světelných podmínek. Ve tmě jsou zornice očí větší, čímž se mění množství světla, které do oka pouští [47]. Vnímání světlosti scény tedy závisí na více faktorech, a u každého jedince se ještě mírně liší.

O světlosti obrazu lze tedy uvažovat ve smyslu jakési energie, kterou objekt vyzařuje. To se shoduje s popisem tempa, uvedeného výše. Jak bylo uvedeno v kapitole *Barevné modely a prostory*, při analyzování světlosti obrázku, je důležité vědět, že ne každý barevný model odpovídá vnímání člověka. Pro sledování světlosti obrázku je vhodné použít barevný prostor `L*a*b*`, který zobrazuje světlost obrázku tak, jak ho vnímají průměrné lidské oči.

Výhody	Nevýhody
⊕ vztah energie fotonu a energie skladby	
⊕ podloženo studiemi	

6.4.3 Struktura písně

Uspořádání částí písně je jedna z nejdůležitějších věcí v hudební skladbě. *Luděk Zenkl* dokonce vydal knihu [48], která se zabývá jednotlivými hudebními formami. Poznávání hudební formy a analýza její struktury se celkem běžně vyučuje na konzervatořích. Z toho plyne, že by se tím měla aplikace, která převádí obraz na hudební artefakt, nějakou chvíli zabývat.

Heat mapa

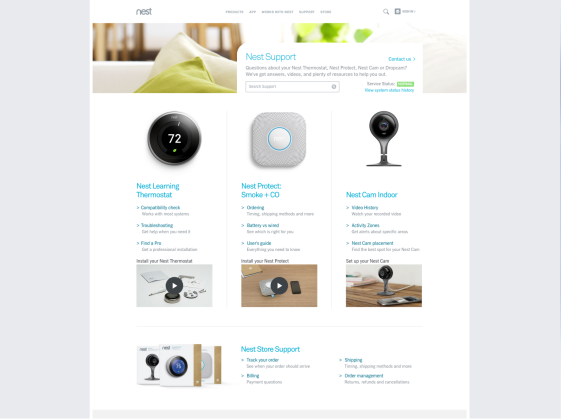
Heat mapy slouží k vyhodnocení chování potencionálních zákazníků určitého produktu. Využívají se např. u webových stránek, kde se sleduje, která tlačítka jsou mačkána nejčastěji, kam se uživatel pohybuje kurzorem a někdy dokonce i to, jak se pohybují jeho oči. Všechna tato data jsou posléze převedena do tzv. *heat mapy*. Nejčastěji se zobrazuje jako 2D obrázek zkoumaného subjektu, který je částečnou průhlenností překryt barevnými oblastmi. Barva oblasti je určena počtem očních fixací průměrného návštěvníka a jejich délkou. Červená značí oblast největší pozornosti a fialová naopak nejmenší. Úplně průhledné oblasti pozornosti v testu nebyly navštíveny žádným účastníkem. Studie společnosti *Nielsen Norman Group* [49] sledovala fixaci očí na designech webových stránek, kdy testerům byly zadány konkrétní úkoly. K návrhu na obrázku 6.6 byla získána heatmapa 6.7. Na základě těchto dat bývá vyhodnoceno uživatelské rozhraní webové stránky či aplikace a předesignováno, aby lépe odpovídalo svému účelu a záměrům vlastníka.

Pokud by se podařilo získat heatmapu s využitím metody sledování očí [50] pro vstupní obrázek, aplikace by dostala možnost zaměřit se na části, které jsou pro průměrného člověka nejvíce atraktivní a reflektovala by tak přirozenou cestu vnímání obrazu. To by následně mohla využít pro generování struktury písně.

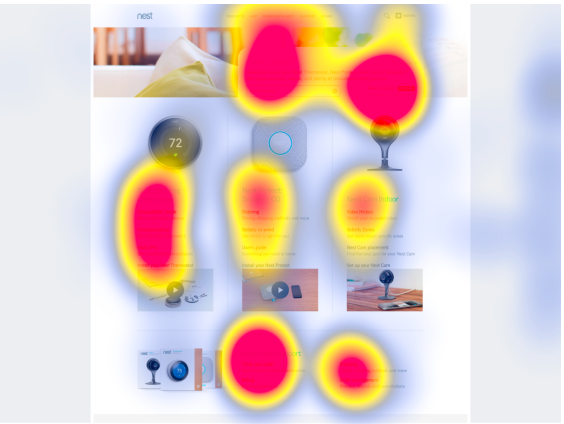
Velmi významnou nevýhodou tohoto postupu, je nutnost použití předem analyzovaných obrázků, časová zátěž pro získání dat[13] a drahé vybavení. Toto lze částečně obejít zapojením AI místo živých uživatelů. Bohužel se nepodařilo najít žádné bezplatné API na získávání těchto dat.

Výhody	Nevýhody
⊕ přirozená důležitost elementů v obrázku	⊖ nutnost mít předem analyzované obrázky
	⊖ placené API

[13] *Nielsen Norman Group* zmiňuje okolo 30 účastníků testování.



Obrázek 6.6 Zdrojová webová stránka



Obrázek 6.7 Vytvořená heat mapa

Objekt jako motiv

Jak již bylo zmíněno v projektu *Generate expressive music*, divák se při prvním setkání s obrázkem soustředí na pouhých 10 % jeho obsahu, což bývají většinou objekty. I ve fotografování tvoří často hlavní téma scény nějaký objekt. Stává se tak hlavním středem pozornosti, k němuž se při prohlídce obrazu člověk častokrát vrací, jak lze spatřit i v předchozí sekci s ukázkou heatmapy. Objekt přitom budí kontrast s pozadím, na kterém je položen. Při každém návratu k objektu od pozadí, dostává trošku jiný rozměr. Známe více informace z pozadí. Stejně tak je motiv v písni věc, která se skladbou prolíná několikrát, pokaždé trochu jinak.

Tuto spojitost lze zvednout o úroveň výše, pokud uvážíme tzv. *příznačný motiv*, který je dle hudební teorie [48] významnou hudební myšlenkou. Slouží jako symbol určité osoby, věci nebo ideje v operách, melodramech, atd. Dále je zde zmíněno, že se tento motiv vrací buď ve stejné podobě nebo v různých obměnách. Když budou obrazová data detekovaného objektu převedeny do hudebního motivu, lze s ním pak pracovat právě jako s *příznačným motivem*. Každému motivu lze určit jeho důležitost v hudební skladbě. Tuto relaci lze zvolit jako velikost objektu v pixelech. Větší objekty, které vidíme jsou buď velmi velké, anebo jsou velmi blízko. V obou případech upoutají naši pozornost dříve než ty menší.

Téma v hudební teorii je výrazná myšlenka, která se někdy člení na motivy [48]. O detekovaných objektech lze tedy uvažovat i jako o tématu obsahujícím několik motivů. Na pozadí obrázku (bez objektů) lze oproti tomu pohlížet jako na doplňující myšlenky, které spojují jednotlivé motivy/témata, a to buď *úvodem, mezivětou* nebo *kodou*. Hlavní myšlenky pak můžou lépe vyznít [48]. Z výše uvedených poznatků lze navrhnout dva způsoby přenesení motivů z obrázku do hudební skladby.

Intro – Hlavní téma – Sloka – Hlavní téma – *Hlavní téma – Outro. *Hlavní téma* je složeno ze všech zřetězených motivů seřazením dle důležitosti. Oproti tomu *intro, sloka* a *outro* vznikly rozložením notového materiálu generovaného z pozadí obrázku na tři části. Představují tak výše uvedené vedlejší myšlenky. Hlavní téma plní ve skladbě funkci refrénu. Při posledním opakování je hlavní téma mírně pozměněno, a to zvýšením tónů melodie o oktávu.

Intro – Hlavní téma – Sloka – *Hlavní téma – Bridge – Hlavní téma – Outro. Forma vypadá podobně jako v předchozím případě. Je zde však využit další prvek, a to rozdělení motivů

na dvě poloviny dle důležitosti. Zatímco *hlavní téma* tvoří motivy z první poloviny, *bridge* je utvořené z nejméně důležitých motivů. Protože jsou objekty seřazené dle své velikosti, představuje téma uspořádané z nejmenších objektů jakési překvapení ve scéně, které divák najednou objevil, čímž obohacuje skladbu, a přitom nechává zachovány relace mezi obrázkem a hudebním artefaktem. Předposlední opakování hlavního tématu má podobně jako v předchozím postupu posunutou melodii o oktávu výš.

Výhodou této metody je jeho shoda s myšlenkovým postupem člověka, tedy dá se říci, že se jedná o relaci velmi blízkou realitě. Pro jeho úspěšnou aplikaci je však třeba mít technologii, která dokáže spolehlivě určit objekt v obrázku. Přestože implementace této technologie je nad rámec rozsahu práce, nabízí se využít již hotového řešení v podobě API. Možno je využít například umělé inteligence, pro kterou je detekce objektů denním chlebem. Nutností však zůstává nalezení kompatibilního API, pomocí kterého bude aplikace s AI komunikovat.

Výhody	Nevýhody
⊕ shoda s myšlenkovým pochodem člověka	⊖ nutnost kompatibilního API
⊕ jednoduchost detekce objektů pomocí AI	⊖ problém harmonizace více melodií

Simulace objektu

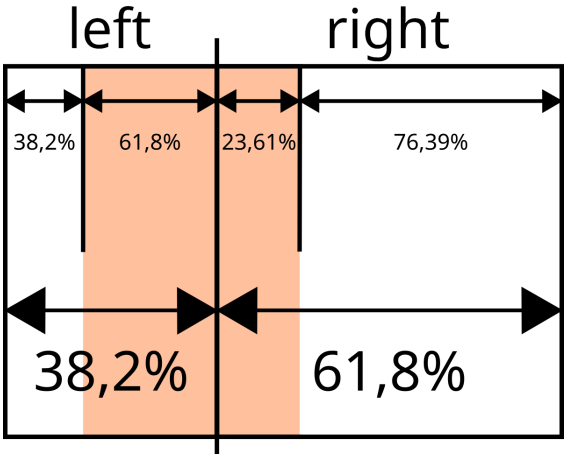
Alternativně lze postupovat cestou, že pokud objekt v obrázku není nebo není možné ho pomocí dostupných prostředků rozpoznat, lze nalezení objektu simulovat. Využit při tom může být princip *zlatého řezu*. Zlatý řez je rozdělení celku na dvě části tak, že větší část má k menší části stejný poměr, jako původní celek k větší části. Tento poměr je iracionální číslo a činí cca `1.618` [51]. Princip zlatého řezu najdeme všude v přírodě, mimo to se využívá se právě u fotografování. Fotografové se snaží umístit předměty na průsečíky různě poskládaných z proporcí zlatých řezů. Simululaci objektu, jako hlavního motivu scény, tak můžeme zasadit právě do lokace určené zlatým řezem.

Když si části rozdělené zlatým řezem přepočítáme na procenta, získáme po zaokrouhlení hodnoty `38,2 %` (menší část) a `61,8 %` (větší část). Pokud jednotlivými procenty vynásobíme konstantní délku, poměr se nezmění. Díky tomuto výpočtu lze algoritmicky rozdělit jakoukoliv úsečku v přibližném poměru zlatého řezu.

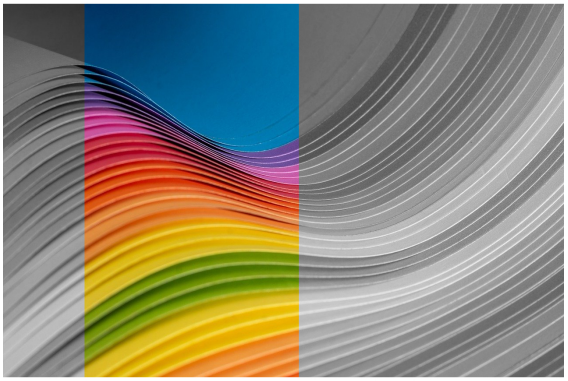
Simulovaný objekt zasadíme do lokace určené zlatým řezem. Situaci ilustrují obrázky 6.8 a 6.9. Pro větší estetičnost bude poměr šířky obrázku a šířky simulovaného objektu taktéž v poměru zlatého řezu. Levý okraj objektu získáme, když menší část zlatého řezu zlatým řezem znovu rozřízneme, a to tak, aby se průsečík nacházel co nejvíce vlevo. Nový průsečík se nachází v 38,2 % levé části obrázku a v cca 14,6 % celého obrázku. Pravý okraj objektu získáme tak, že od původního průsečíku postupujeme doprava právě o 14,6 % celého obrázku (což tvoří 23,61 % z pravé části obrázku). Tímto objekt získá šířku 38,2 % vzhledem k obrázku[14] [15], což je jedna z částí zlatého řezu.

Výsledné mapování

Vzhledem k tomu, že na mapování pomocí *Heat mapy* nebyly nalezeny potřebné prostředky, vypadává ze hry. Obě mapování se zdají být z hlediska principu relací a svých výhod velmi vhodné, navíc jedno může fungovat bez problému vedle druhého. Aplikace je proto implementuje obě.



Obrázek 6.8 Přehled výřezu dle zlatého řezu



Obrázek 6.9 Simulace v obrázku [52] (*edit.*)

6.4.4 Hudební nástroje

Zatímco barva v obrázku je analogie pro vlnovou délku světla, které se od plátna odráží, v hudbě tento pojem označuje spektrum tónu. To je tvořeno z různých částkových tónů, jejichž poměr a síla určuje líbivost zvuku. Líbivost zvuku má v hudební skladbě dokreslovat její atmosféru. Přiřazení hudebních nástrojů je tedy velmi důležitá položka.

Průměrná saturace

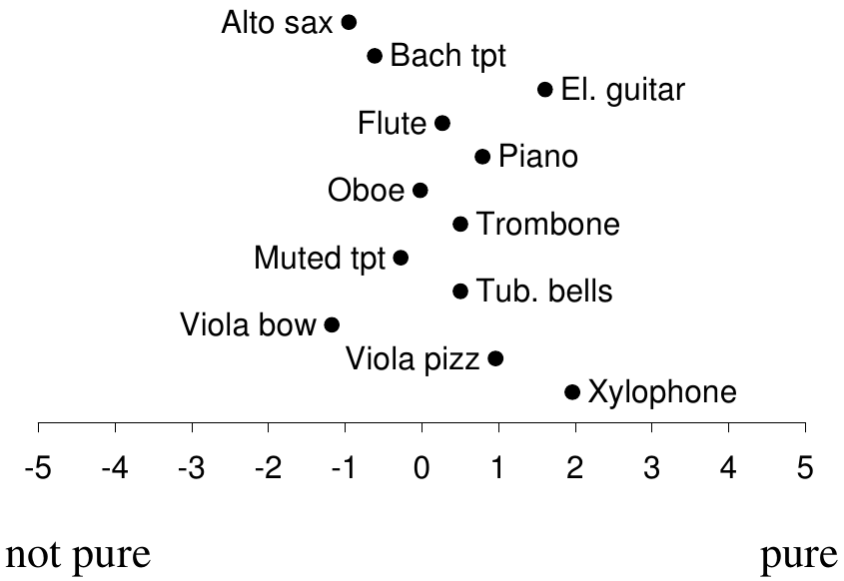
Ve své studii zabývající se vztahy mezi zvukem a hudbou [17] J. L. Caivano zmiňuje, že o barvě tónu, tedy o použitém hudebním nástroji, lze uvažovat jako o čistotě zvuku. Stejnou jednotkou se zdá být saturace u barvy. Přílišná míra šedé barvu dělá víceméně špinavější. Pokud má hudební nástroj příliš mnoho nepravidelných částkových tónů, zní méně čistě.

Ve studii *Timbral description of musical instruments* [53] měli její účastníci přiřadit k několika hudebním nástrojům hodnotu na škále dvou vzájemně opačných slov, kterými se hudební nástroje běžně popisují (dřevěný, kovový, drsný, atd.). Součástí tohoto testu bylo i zařazení nástroje dle čistoty jeho zvuku. Nástroje byly přiřazeny dle grafu na obrázku 6.10.

Šlo by vybrat některé z nástrojů a dle průměrné saturace obrázku přiřadit k hudební lince. Když je bráno v úvahu, že hlavním motivem v obrázku jsou objekty, lze pohlížet na melodii jako hlavní linku a přiřadit jí hudební nástroj dle průměrné saturace největšího detekovaného objektu. Pro zajištění propojení jednotky (objektu) s celkem (celá scéna) by měl dostat doprovod hudební nástroj dle průměrné saturace celého obrázku.

Výhody	Nevýhody
⊕ vztah jednotky a celku napříc modalitami	
⊕ založeno na studii	

[14] Relativně vůči obrázku: 38,2 % (velikost levé části) − 14,6 % (velikost části před objektem) = 23.61 %.
[15] 23,61 % (velikost objektu před hlavním průsečíkem) + 14,6 % (velikost objektu za hlavním průsečíkem) = 38,2 %.



Obrázek 6.10 Výsledek zařazení hudebních nástrojů dle čistoty zvuku [53]

Harmonická analýza

Jak již bylo řečeno, barva tónu hudebního nástroje závisí na jeho tónovém spektru a na poměru a síle frekvencí v něm obsažených. I z obrázku lze generovat spektrum frekvencí pomocí *Fourierovy transformace*.

Obrázek by následně byl vyhodnocen pomocí harmonické analýzy a poté by bylo detekováno harmonické spektrum frekvencí. To by bylo dále porovnáno s jednotlivými tónovými spektry hudebních nástrojů.

Nevýhodou tohoto postupu je velmi přibližná podoba tónového spektra a harmonického spektra frekvencí. Protože tato spektra si nejsou moc podobná, bude třeba je jistým způsobem zjednodušit. Dále se spektrum tónu vyvíjí i v čase. K vytvoření této vazby by tak bylo potřeba sehnat vzorky tónového spektra z různých nástrojů.

Výhody	Nevýhody
⊕ vztah mezi podobnými veličinami napříč zvukem a obrazem	⊖ nákladné na sbírání hudebních vzorků

Výsledné mapování

Vzhledem k rozsahu práce bylo rozhodnuto o přičlenění hudebního nástroje k hudební lince pomocí vazby na průměrnou saturaci obrazových dat. Ač se zdá být druhé mapování zajímavé, nezbyl by na něj čas a prostor.

Kapitola 7

Návrh technologických možností

V této kapitole bude provedena analýza rozhraní aplikace, představen programovací jazyk a technologie, použitelné při implementaci. Dále bude nastíněna podoba vstupu, podoba výsledku a podporované operační systémy.

7.1 Analýza rozhraní

V následující kapitole bude analyzováno možné rozhraní pro budoucí aplikaci. Posuzovány jsou dvě hlavní kategorie: *konzolová aplikace* a *grafické uživatelské rozhraní*

Konzolová aplikace

Konzolové rozhraní lze implementovat dvěma způsoby. Buď se parametry generování zadají jako argumenty při spuštění aplikace nebo je bude aplikace po svém spuštění postupně přijímat na vstupu (nemusí být nutně dodrženo pořadí).

Jasnou nevýhodou tohoto postupu je nevelká přívětivost programu pro uživatele. Většina uživatelů je totiž zvyklá používat grafické uživatelské rozhraní. Jen malá část[1] z nich by uvítala, kdyby pro používání aplikace musela sáhnout po příkazovém řádku. Konzolové grafické rozhraní je na druhé straně jednodušší na implementaci. Uživatel z principu nepočítá s tím, že by rozhraní mělo nějak reagovat na jeho vstup poté, co spustil výpočet.

Výhody	Nevýhody
⊕ jednodušší na implementaci	⊖ nepříliš přívětivé pro většinu uživatelů

GUI

Grafické uživatelské rozhraní[2] nepředstavuje výhody jen pro uživatele v podobě jednodušší interakce, ale také pro vývojáře, ve smyslu sbírání a třízení jeho vstupu. Lze také jednodušeji předdefinovat, co uživatel bude moci aplikaci říct, a k čemu už ho aplikace nepustí.

[1] Pozor, jsou tu i výjimky. Zejména uživatelé linuxových operačních systémů.
[2] GUI

Pro dobré uživatelské rozhraní je důležité přichystat vizuální návrh a následně ho implementovat. Tomu pomáhá spoustu knihoven na podporu tvorby dostupných napříč programovacími jazyky. Dále je třeba počítat s tím, že když budou probíhat výpočty na pozadí, aplikace musí stále reagovat[3] na vstup uživatele. Tedy je nutno zajistit chod více vláken/procesů programu a s tím spojené zabezpečení sdílených prostředků mezi nimi.

Výhody	Nevýhody
⊕ jednodušší na ovládání pro uživatele	⊖ nutnost synchronizace vláken/procesů
⊕ uživatel nerozbije tolik věcí	⊖ práce na návrhu rozhraní
⊕ jednodušší zpracovávání vstupu	

Vyhodnocení

Vhledem k tomu, že cílová skupina budou uživatelé, kteří mají vztah k hudbě, a také jiní umělci, používání příkazového řádku by pro většinu těchto lidí mohlo být zbytečně demotivující. Z tohoto důvodu a z vypsaných výhod, bude použito grafického uživatelského rozhraní.

7.2 Programovací jazyk

Python je jazyk, s nímž má autorka zkušenosti z předchozích projektů. Je textově úsporný a poskytuje množství knihoven, které jsou k dispozici v podobě balíčků, lehce stažitelné a integrovatelné do aplikace.

SCAMP, knihovna, umožňující generování hudební notace, je implementována v *Pythonu*. Knihovna *OpenCV*, zajišťující rychlou a efektivní práci s obrazem je *Pythonem* podporovaná. *OpenCV* navíc zajišťuje převod do barevného prostoru *L*a*b**, na který je v práci častokrát odkazováno. Modul *python-colormath* implementuje vzorec pro porovnávání barev – *Delta E CIE 2000* [10]. Stejně tak *ImageAI* [54], knihovna detekující objekty, a *PySimpleGUI* [55], poskytující možnost zjednodušení implementace grafického uživatelského rozhraní.

Vzhledem k výše uvedeným výhodám, bude aplikace používat programovací jazyk *Python*. Protože knihovna *ImageAI* je v současnosti implementována pro nejvyšší verzi *Pythonu* – `Python 3.7.6`, bude i aplikace používat tuto verzi.

7.3 Podoba vstupu

Zdrojový obrázek, ze kterého bude probíhat generování hudební skladby, by měl být ve formátu, který knihovna používaná pro načítání dat, bude podporovat. `PNG` a `JPEG` jsou obrazové formáty standardně používané napříc počítačovým světem.

Na systému *Windows* podporuje knihovna *OpenCV* oba z nich. Na linuxových operačních systémech je knihovna *OpenCV* závislá na podpoře formátů systémem. Linuxová distribuce tedy bude zvolená tak, aby knihovně *openCV* načítání formátů *PNG* a *JPEG* umožnila.

7.4 Podoba výsledku

Výsledek generované hudební skladby může být pojat mnoha způsoby. V následujících odstavcích budou představeny výstupní formáty a důvody, proč byly zvoleny.

[3] Jinak se při spuštění výpočtu objeví známá hláška *Program neodpovídá.*

Grafické znázornění

Aplikace během svého zpracovávání obrázku s ním provádí různé operace, které se dají graficky znázornit. Například pro zjištění délky tónu bude potřeba nejprve detekovat regiony s podobnou barvou. Dále bude třeba provést rozdělení obrázku do segmentů.

To všechno jsou věci, které by mohly být pro uživatele zajímavé. V případě změny nastavení parametrů by viděl, jak se změnilo chování algoritmu, a to v grafické[4] podobě. Obrázkový výstup by aplikaci mohl také přidat jistý umělecký faktor (pozměněná podoba obrázku). Pokud by byl tento grafický výstup reflektován v reálném čase v GUI, mohl by sloužit uživateli i jako ukazatel progresu aplikace.

V kapitole *Mapování modalit* je řečeno, že aplikace bude detekovat objekty. S využitím knihovny `ImageAI` [54], která poskytuje grafické znázornění detekovaných objektů pomocí ohraničení rámečky, by mohl uživatel zjistit, které objekty byly při generování hudební skladby detekovány.

Hudební notace

Při použití knihovny SCAMP [56] je generování hudební notace velmi snadné. Navíc notový zápis oproti zvukové nahrávce mnoho paměti nezabírá. Nevýhodou je žádost uživatele o instalování některých dalších programů, které převod do člověkem čitelných formátů zaručují.

Formáty hudební notace mohou být různé. Určitě by nemělo chybět `PDF`, jeden z nejpoužívanějších formátů dnešních dob. Jsou v něm zapsané dokumenty, nákresy určené k tisku nebo vektorové obrázky. Je tedy velká pravděpodobnost, že uživatel nebude mít problémy hudební notaci zobrazit.

Knihovna SCAMP generuje `PDF` pomocí programu *Lilypond*. Když už tento program bude muset mít uživatel nainstalovaný, nebylo by marné vygenerovat hudební notaci taktéž v něm. Formát `Lilypond` je založen na jednoduchém vytváření notového zápisu pomocí vlastního programovacího jazyka. Uživatelé, kteří budou chtít hudební notaci později upravovat (což formát `PDF` nedovoluje), ho ocení.

Přesto, že formát `Lilypond` zaručuje (dle svých tvůrců) snadnou úpravu notace, mnoho grafických editorů hudební notace[5] nepodporuje jeho import. Uživatel ho tedy musí nejdříve převést pomocí příkazu nebo grafického rozhraní programu *Lilipond* do jiného formátu nebo se naučit výše zmíněný jazyk. Aby aplikace uživateli zbytečně nepřidělávala práci a nenutila ho stát se programátorem, poskytne mu i více standardní formát notového zápisu, `MusicXML`.

Výhody	Nevýhody
⊕ úsporné řešení z hlediska paměti	⊖ potřeba externího programu

7.4.1 Zvukový výstup

Hudební notace generované skladby se zdá být dobrým nápadem, co však budou dělat uživatelé, kteří neumí noty, a zároveň běžně nepoužívají žádný editor hudební notace, který by jim mohl skladbu přehrát?

Existuje spousta zvukových formátů, do kterých by šlo skladbu konvertovat. Protože však není v rámci rozsahu práce možné implementovat všechy, uživatel by měl mít na výběr především ty, které pro něj představují nějakou výhodu.

[4] Příklad: zvětšila se šířka segmentu, zvětšila se i ve výstupním obrázku.
[5] Guitar pro 7, TuxGuitar, MuseScore

`SMF`[6] je standardizovaný datový formát, který využívá protokolu *MIDI*[7]. Právě kvůli tomuto známému protokolu a díky své příponě bude v práci dále označován jako *`MIDI` formát*. `MIDI` neobsahuje zvuková data, ale sekvenci příkazů zmíněného protokolu. Jde tedy o vcelku malý soubor, který umí přehrát i *Windows Media Player*, obsažený v systému *Windows*. Nevýhodou a zároveň výhodou `MIDI` je, že zvuk použitých nástrojů nemá pevně dán. Při přehrávání jsou hudební tóny generované v reálném čase, a to za použití tzv. *soundfontů* (hudebních fontů). Pokud to hudební editor[8] dovoluje, může uživatel přiřadit zvolené hudební fonty k jednotlivým hudebním linkám, a tím upravit zvuk. Výše uvedené přehrávače `MIDI` většinou mívají výchozí soundfont, takže se uživatel nemusí o nic starat.

Dalším formátem, pomocí kterého by si mohl uživatel poslechnout generovanou hudební skladbu, je `WAV`. Zatímco `MIDI` je z hlediska velikosti malé a snadno uskladnitelné, formát `WAV` bývá velký. Zato díky své nekomprimované (a tudíž bezztrátové) podobě poskytuje uživateli nahrávku ve vysoké kvalitě. Při generování `WAV` je třeba použít vlastní hudební font. Aplikace tak může zvolit ten, který se zdá mít dobré hudební vlastnosti.

7.5 Podporované operační systémy

Většina uživatelů používá systém *Microsoft Windows*. Je tedy potřeba zajistit, aby byla aplikace spustitelná na něm. *Windows* je však placený. Uživatelé by měli dostat šanci spustit aplikaci i na systému, který je zdarma. V úvahu tak přichází *Linux*. Pokud bude program implementován alespoň pro jednu linuxovou distribuci, uživatel si ji může nainstalovat, a tak bude moci aplikaci používat bez nutnosti platit za operační systém.

Z výše uvedené části je patrné, že linuxová distribuce musí knihovně *OpenCV* zajistit načítání obrázkových formátů. Z tohoto důvodu bude vybrána distribuce *Manjaro*, založená na *Arch linuxu*.

[6] Standard MIDI File
[7] Musical Instrument Digital Interface
[8] Guitar Pro, MuseScore

Kapitola 8

Návrh uživatelského rozhraní

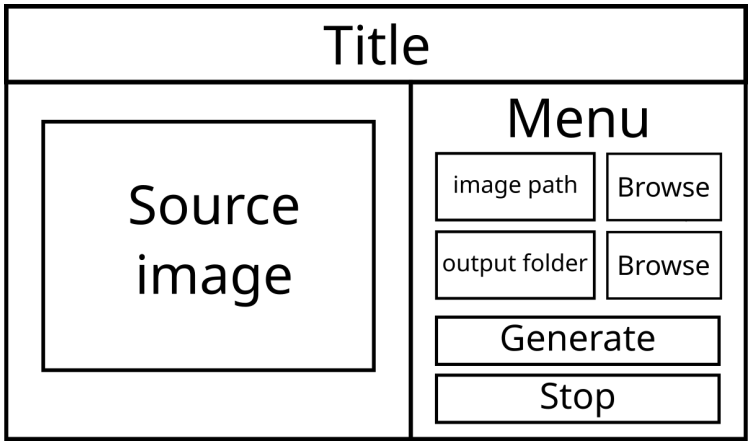
V aplikaci vytvářené v rámci bakalářské práce bude uživateli k dispozici grafické uživatelské rozhraní. K jeho implementaci je však nejdříve potřeba toto rozhraní graficky navrhnout. V následujících sekcích bude po jednotlivých krocích rozebrán postup návrhu. S odkazem na předchozí prototyp MusicWave, přes základní rozložení oddílů a ovládacích tlačítek, po finální podobu aplikace.

8.1 Rozhraní MusicWave

Aplikace je volným pokračováním autorčina projektu *MusicWave*. Tento projekt poskytoval uživateli rozhraní pouze přes příkazový řádek a informace o svém stavu vypisoval taktéž do konzole, jak lze vidět na ukázce 8.1.

```
################################################################################
################################### Music Wave #################################
Displaying '/home/radka/bridge.jpg' ...
Using preset Acoustic Bass for Acoustic Bass
Using preset Acoustic Bass for Acoustic Bass
################################## Informations ################################
Title: bridge.jpg
Composer: The image
Instruments: Acoustic Bass, Acoustic Bass
Tempo: 120 BPM
Source image: /home/radka/bridge.jpg
Exported music score: /home/radka/bridge.jpg_song.pdf
################################### Progress ###################################
Transcribing has started...
Playing...
```

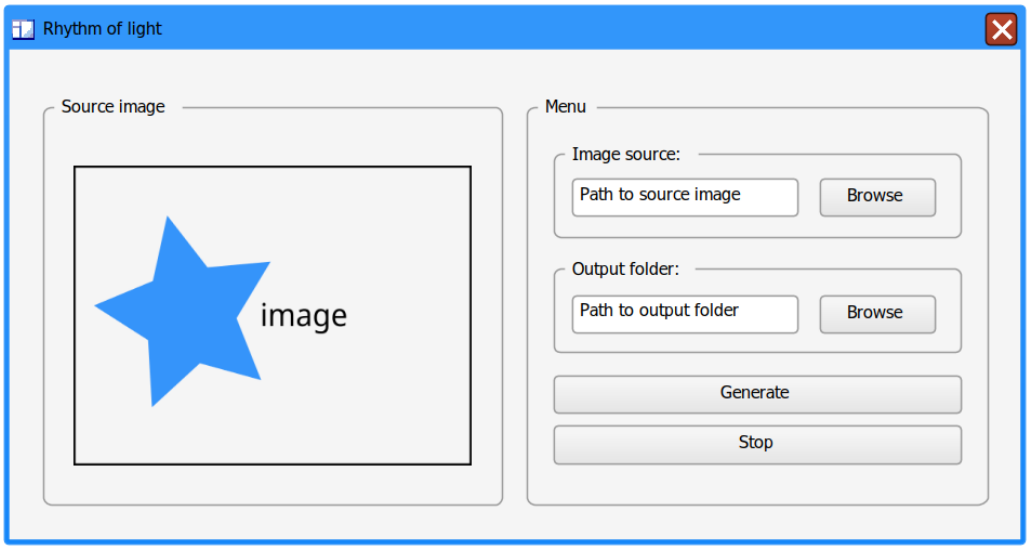
Obrázek 8.1 GUI projektu MusicWave



Obrázek 8.2 Prvotní návrh GUI

8.2 Prvotní návrh

Na obrázku 8.2 je ukázáno základní rozložení grafického rozhraní aplikace a jeho následný wireframe 8.3. Aplikace se dělí na dvě hlavní části: *Zobrazování operací nad zdrojovým obrázkem* a *Menu*. V návrhu jsou také zachyceny nejpodstatnější ovládací prvky, a to: *cesta k zdrojovému obrázku, složka určená k uložení výstupu* a *tlačítko pro spuštění generování*.



Obrázek 8.3 Wireframe prvotního návrhu GUI

Tabulka 8.1 Detaily vstupních a výstupních polí

Datový typ	Jméno vstupního pole	Rozsah hodnot
`string`	Cesta k zdrojovému obrázku	
`string`	Cesta k výstupní složce	
`int`	Procentuální přesnost detekce objektů	1%–99%
`int`	Tolerance barev	1–100
výčet hodnot	Zdroj pomlk v rytmu	tmavý segment, světlý segment
`int`	Šířka segmentu v % šířky obrázku	1%–100%
`int`	Výška segmentu v % výšky obrázku	1%–100%
výčet hodnot	Tónina hudebního artefaktu	`C–H dur`
výčet hodnot	Minimální délka noty v rytmu	dvaatřicetinová–celá nota
výčet hodnot	Minimální délka noty v melodii	dvaatřicetinová–celá nota
`int`	Maximální tempo	dvaatřicetinová–celá nota
výčet hodnot	Kompozice skladby	Intro-Middle-Outro, Motive-Driven
`bool`	Exportovat jako PDF	
`bool`	Exportovat jako Lilypond	
`bool`	Exportovat jako MusicXML	
`bool`	Exportovat jako MIDI	
`bool`	Exportovat jako WAV	
tlačítko	Generovat	
tlačítko	Stop	

Datový typ	Jméno výstupního pole	Rozsah hodnot
`string`	Výpis informací o stavu programu	
ukazatel průběhu	Ukazatel průběhu generování	
`string`	Výpis chybových hlášení	
obrázek	Zdrojový obrázek	
obrázek	Obrázek s detekovanými objekty	
obrázek	Zobrazení průběžné segmentace obrázku	

8.3 Přidání dalších prvků

Na základě vstupních parametrů, které potřebuje aplikace ke svému chodu, byly do uživatelského rozhraní přidány další ovládací prvky. Detaily o jejich datových typech je možno nalézt v tabulce 8.1. Aplikace bude používat jedno okno, ale zároveň v něm bude zobrazovat několik svých částí. Bude využívat tzv. karet, ve kterých bude ukryty další ovládací a zobrazovací prvky. Celé schéma rozšířeného návrhu ukazuje obrázek 8.4 a wireframe rozšířeného návrhu 8.5.

8.3.1 Menu

Menu aplikace dostane šest karet: *Hlavní nastavení*, *Nastavení segmentace*, *Nastavení notace*, *Nastavení exportu*, *O aplikaci* a *Zobrazování chyb*. Poslední karta bude neviditelná a zobrazí se pouze v případě, kdy bude nutné uživatele informovat o určité chybě - například pokud byl zadán neplatný vstup. Každá karta bude představovat ucelenou jednotku nastavení, které bude uživateli k dispozici.

Zároveň se na levé straně aplikace budou nacházet i hlavní kontrolní tlačítka *Generovat* a *Stop*.

Pod nimi bude ukazatel průběhu generování[1] a textový popis aktuálního stavu. Tento blok se bude zobrazovat pod kartami jednotlivých nastavení, takže bude viditelný při výběru libovolné z karet. Díky tomu uživatel nikdy neztratí pojem o postupu generování.

Přehled karet menu

Hlavní nastavení bude sloužit jako most mezi vstupem a výstupem. Uživatel bude mít k dispozici pole pro zapsání *cesty zdrojového obrázku* a *výstupní složky*.

Nastavení segmentace určí vstupní údaje k *přesnosti detekce objektů, jemnosti segmentace barev, významu světlosti segmentu v rytmu, šířce segmentu v procentech, výšce segmentu v procentech*.

Nastavení notace souvisí s generováním not. Uživatel dostane na výběr z *tónin*, bude moci zvolit *minimální délku noty v rytmu* a *minimální délku noty v melodii*. Dále určit *maximální tempo skladby* a *způsob její kompozice*.

Nastavení exportu provede uživatele možnými formáty, ve kterých je možné vygenerovanou skladbu uložit. To zahrnuje *formát* `PDF`, `Lilypond`, `MusicXML`, `MIDI` *a* `WAV`.

O aplikaci zobrazí informace o použitých balíčcích a technologií a uvede jméno autorky aplikace.

Zobrazování chyb bude sloužit k vypsání vyskytnutých chybových hlášení aplikace, ať už se bude jednat o neplatný vstup či neexistující zdrojový obrázek. Karta bude schována do doby, než bude potřeba poprvé uživateli ohlásit nějaký nedostatek.

8.3.2 Zobrazovací část

Stejně jako pro vstup, i pro výstup bude definováno několik přepínatelných oddílů: *Originální obrázek*, *Zobrazení detekovaných objektů* a *Segmentace dle barev*. V každé z karet se bude zobrazovat podoba obrázku v určité fázi výpočtu aplikace. Díky přepínání karet bude moci uživatel jednotlivé verze porovnávat.

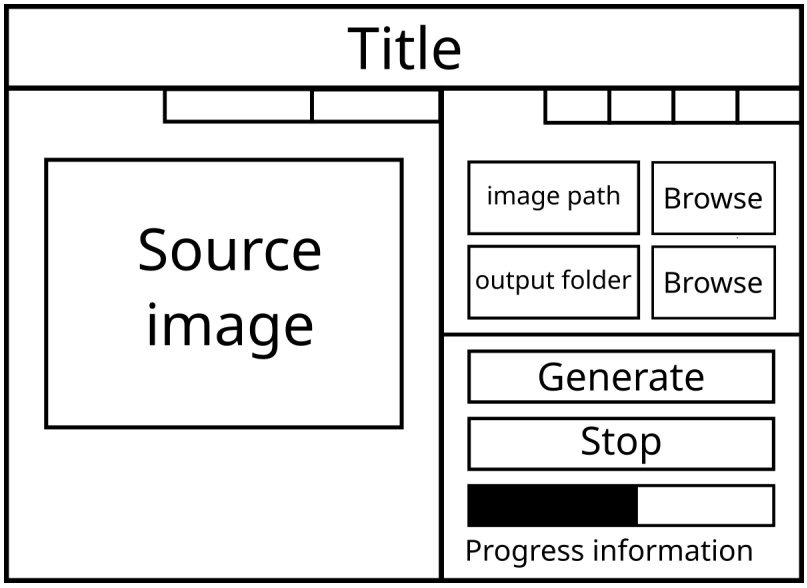
Přehled karet zobrazovací části

Originální obrázek bude po celou dobu zobrazovat zdrojový obrázek, aby měl uživatel možnost srovnání s jeho upravenými verzemi.

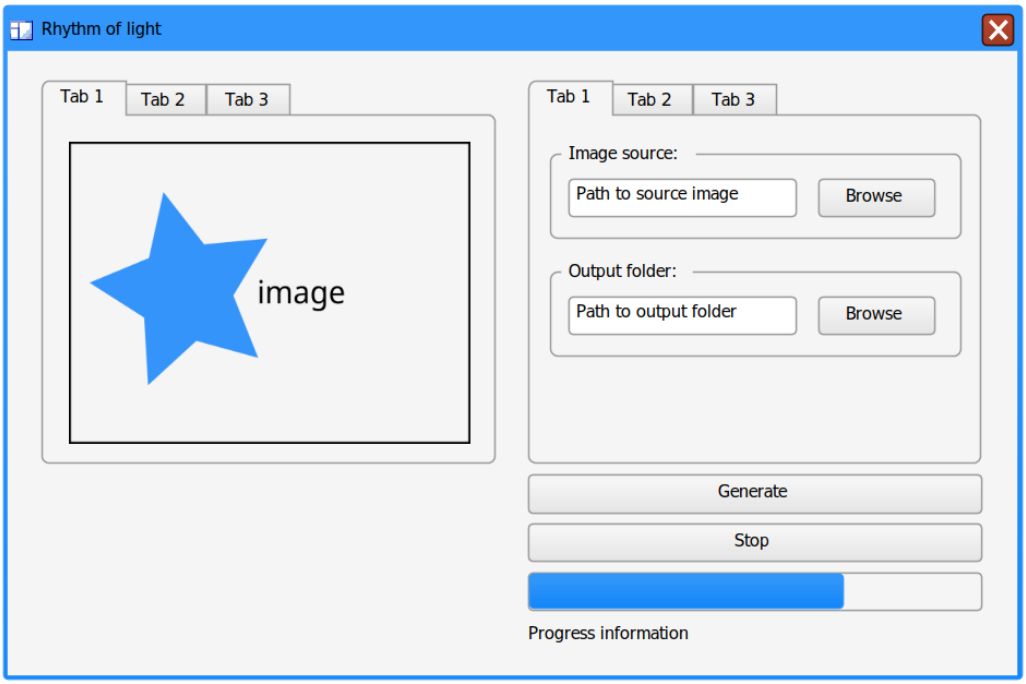
Zobrazení detekovaných objektů si vezme na starost výstup z detektoru objektů.

Segmentace dle barev bude ukazovat obrázek s vyznačenými barevnými regiony v jednotlivých segmentech.

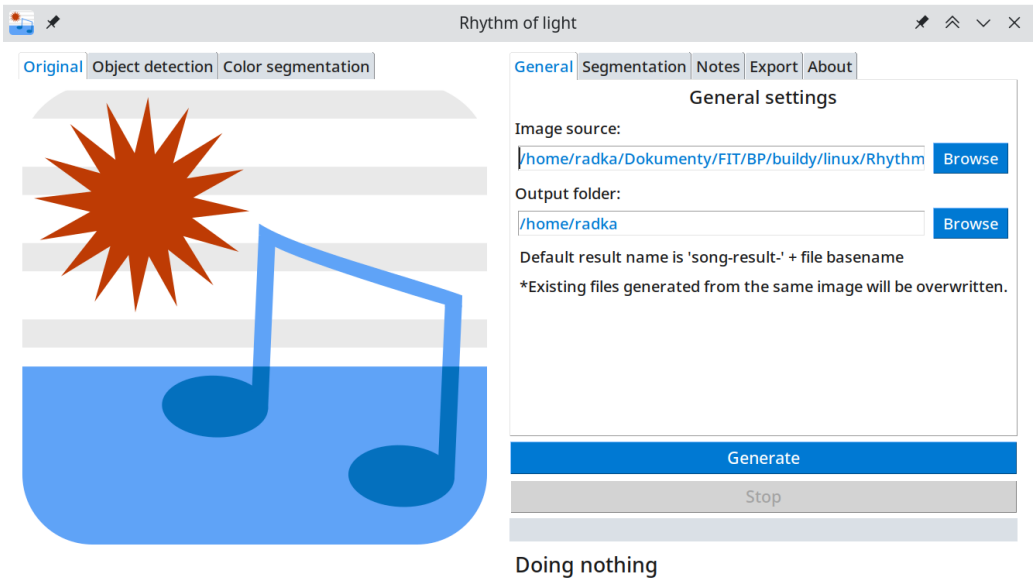
[1] progress bar



Obrázek 8.4 Rozšířený návrh GUI



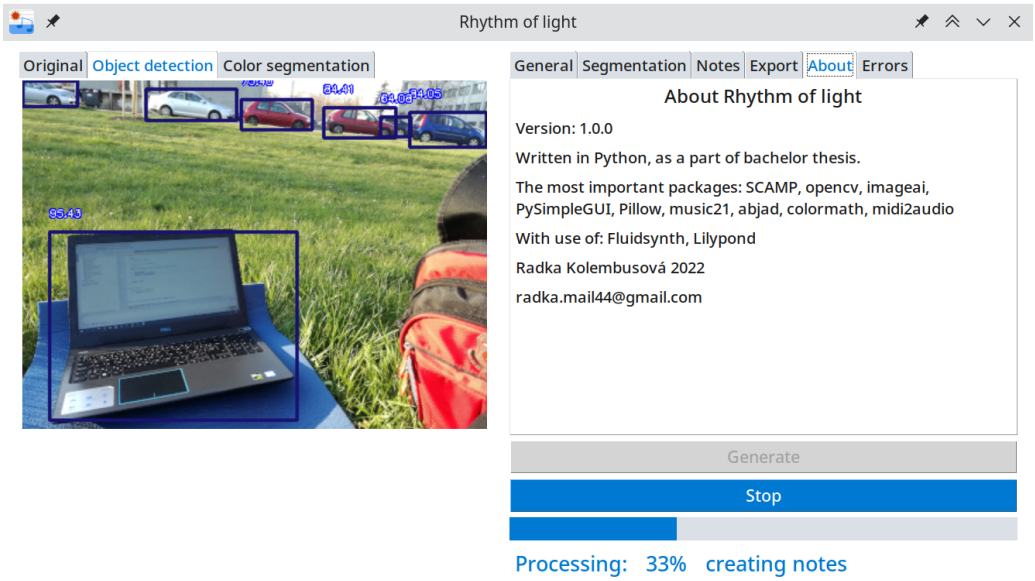
Obrázek 8.5 Wireframe rozšířeného návrhu GUI



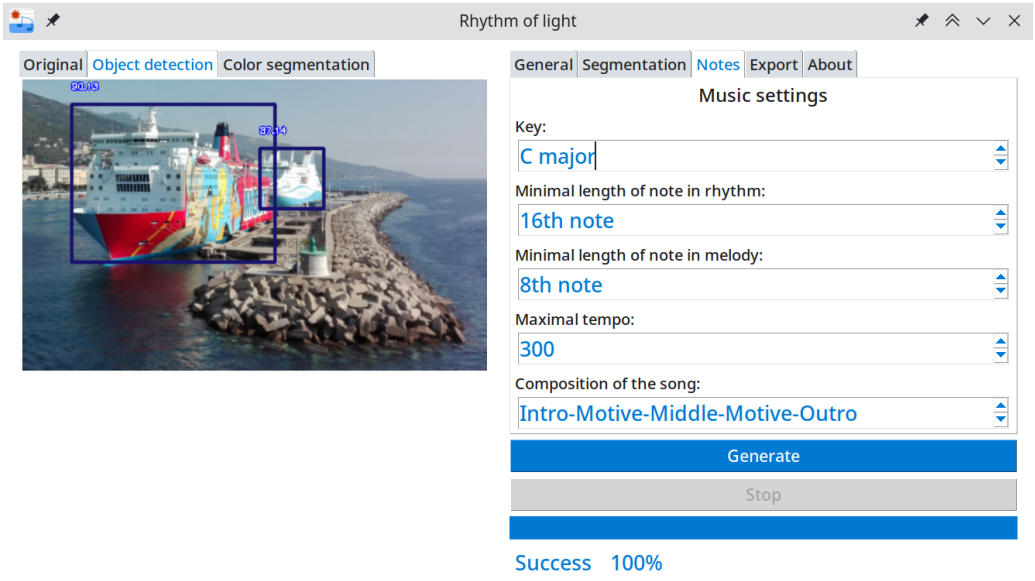
Obrázek 8.6 Konečná podoba GUI: Prvotní setkání aplikace s uživatelem. Vlevo: Úvodní obrázek aplikace v kartě originální obrázek. Vpravo: Hlavní nastavení.

8.4 Konečná podoba

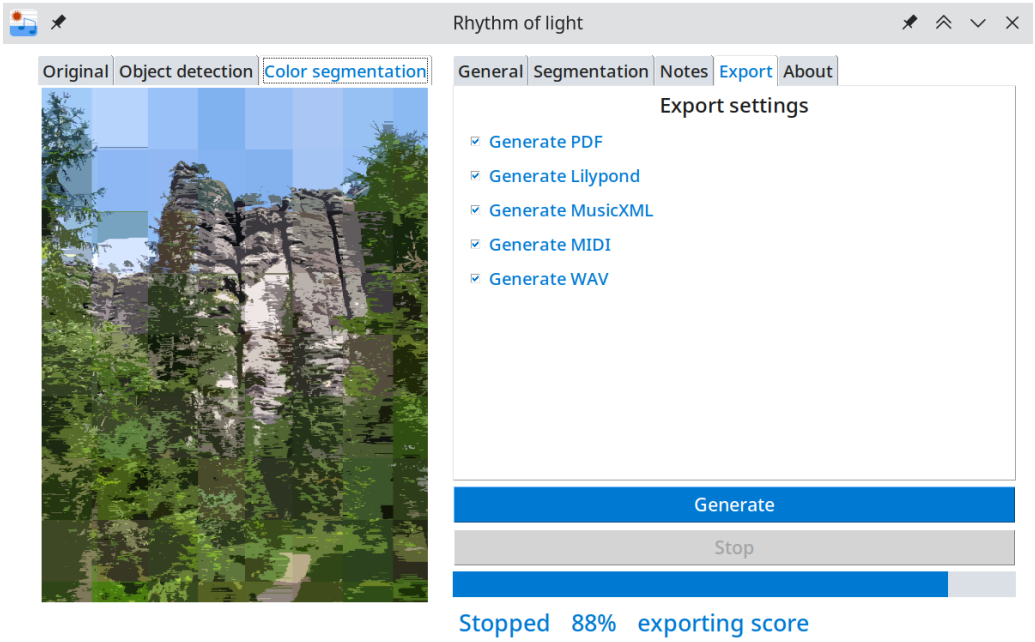
Obrázky 8.6, 8.7, 8.8, 8.9, 8.11 zobrazují finální podobu grafického uživatelského rozhraní. Ke každému obrázku je popisek, který upřesňuje, jaké části aplikace jsou na něm viditelné. Dále pojmenovává fázi či stav, ve které se aplikace nachází. Screenshoty byly pořízeny na linuxové distribuci Manjaro.



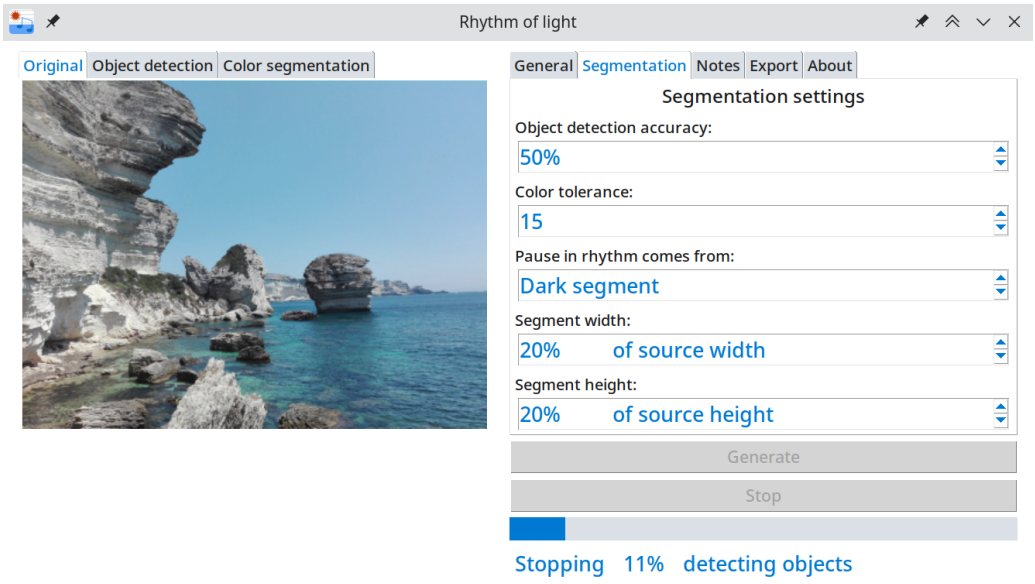
Obrázek 8.7 Aplikace zachycena v průběhu generování. Stav: generování not. Celkový průběh generování: 33%. Vlevo: Zobrazení detekovaných objektů. Vpravo: O aplikaci.



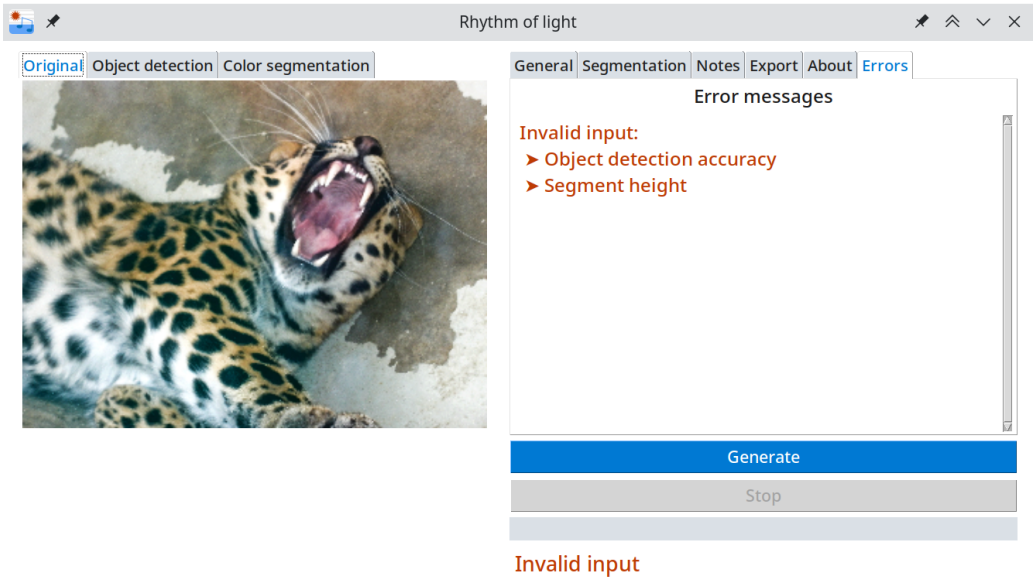
Obrázek 8.8 Aplikace dokončila v pořádku generování hudby. Vlevo: detekované objekty v obrázku. Vpravo: Nastavení notace.



Obrázek 8.9 Aplikace zastavena. Stav: Export notového zápisu. Celkový průběh generování: 88%. Vlevo: Dokončená segmentace zdrojového obrázku. Vpravo: Nastavení exportu.



Obrázek 8.10 Aplikace zachycena v průběhu zastavování procesu. Stav: detekování objektů. Celkový průběh generování: 11%. Vlevo: Originální obrázek. Vpravo: Nastavení segmentace.



Obrázek 8.11 Aplikace skončila s chybou: Neplatná přesnost detekce objektů a výška segmentu. Vlevo: Zdrojový obrázek z předešlého běhu aplikace. Vpravo: Zobrazení skryté karty *Zobrazování chyb* spolu s hláškou pro uživatele.

Kapitola 9

Softwarový návrh

V této kapitole bude rozebrán softwarový návrh aplikace. První částí je analýza *funkčních a nefunkčních požadavků*, dále bude představen *Model případů užití* a jako poslední sekce bude následovat *Architektura*.

9.1 Funkční a nefunkční požadavky

Cílem analýzy funkčních a nefunkčních požadavků, je především *vymezit hranice systému* a *zachytit omezení, která jsou na systém kladena*. Hranice systému vymezují *Funkční požadavky* a omezení kladená na systém *Nefunkční požadavky*. U každého požadavku je jeho popis a přiřazená zkratka.

9.1.1 Funkční požadavky

F1 – Sonifikace obrazu. Systém bude provádět sonifikaci obrázku zadaného na vstupu. Podporované formáty obrázku jsou `JPG` a `PNG`. Alfa kanál[1] v obrázku není žádoucí a aplikace informaci o něm bude při zpracování zahazovat.

F2 – Generování notového zápisu. Aplikace bude umět generovat notový zápis hudebního artefaktu, a to ve formátech `MusicXML`, `Lilypond` a `PDF`. Hra na hudební nástroj dle notového zápisu nemusí být technicky proveditelná lidmi.

F3 – Generování `MIDI` souboru. Aplikace bude umět z obrázku vygenerovat `MIDI` soubor. `MIDI` bude obsahovat popis hudebních linek (názvy hudebních nástrojů), které skladba používá, a informaci o hudebních tónech v nich (časové zařazení, výšku, délku, hlasitost, atd.)

F4 – Generování audio souboru. Uživatel bude moci pomocí aplikace převést hudební artefakt na zvukový soubor ve formátu `WAV`. Předpokládá se, že tento soubor bude generován syntézou nebo jiným digitálním procesem. Nepředpokládá se, že by soubor byl generován jako záznam živého vystoupení hudebníků. Převod do jiných formátu nebude aplikace zajišťovat.

[1] kanál průhlednosti

F5 – Informování o stavu. Systém bude během svého působení informovat uživatele o svém stavu, a to zobrazením příslušných informací ve svém okně. Není v plánu uživatele upozorňovat pomocí notifikací v operačním systému nebo jiným podobným způsobem.

F6 – Zachycení výsledku segmentace. Aplikace jako součást výsledku vygeneruje obrázek, který bude zachycovat výsledek segmentace barev. Tento obrázek uloží společně s ostatními výsledky ve formátu `PNG` do výstupní složky.

F7 – Determinismus. Ze stejného obrázku se stejnými vstupními parametry bude pokaždé vygenerován stejný notový zápis, stejný audio soubor a stejný segmentovaný obrázek.

F8 – Zastavení generování. Uživatel bude mít možnost zastavit generování hudebního artefaktu v každé fázi. Nejsou kladeny nároky na délku ukončování procesu.

F9 – Nastavitelnost parametrů. Uživatel bude moci před generováním nastavit v aplikaci parametry generování. Není vyžadováno ukládání parametrů zadaných během minulého běhu aplikace nebo synchronizace nastavení mezi jednotlivými instancemi aplikace v systémech. Nastavení parametrů bude možné před zahájením procesu generování, nikoliv v jeho průběhu.

F10 – Mapování modalit. Sonifikace obrázku bude prováděna na základě mapování modalit popsaného v kapitole *Mapování modalit*.

9.1.2 Nefunkční požadavky

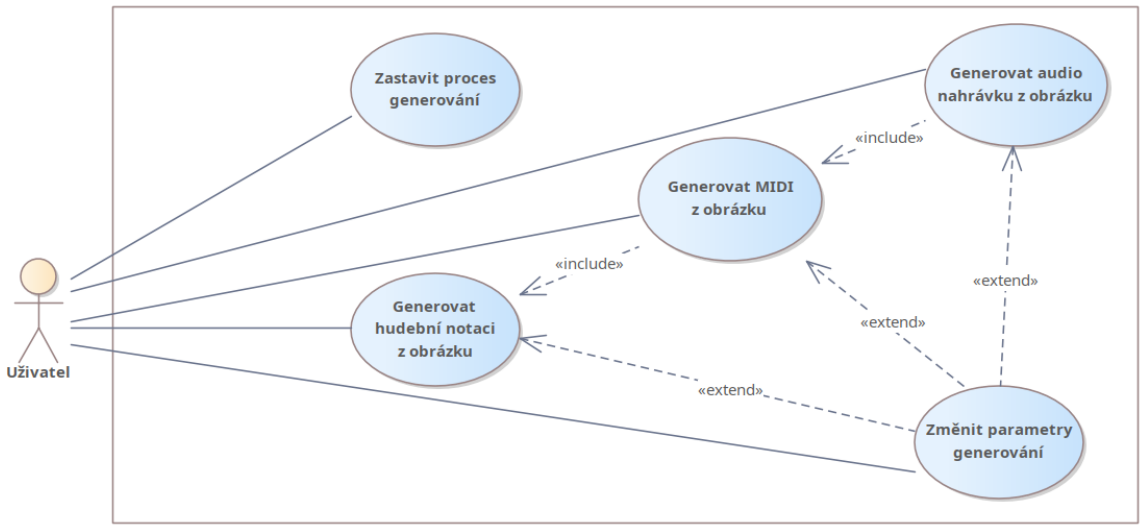
N1 – Dostupnost přes desktopovou aplikaci. Aplikace bude fungovat jako desktopová aplikace. Není vyžadována funkčnost na mobilních a podobných zařízeních (chytrý telefon, chytrý tablet, atd.) nebo na webu.

N2 – Přenositelnost. Aplikace bude fungovat na operačních systémech linuxové distribuce *Manjaro* a *Microsoft Windows 10*. Není v plánu zajišťovat přenositelnost a synchronizaci generovaných dat a nastavení mezi systémy.

N3 – Interakce přes grafické uživatelské rozhraní. Uživatel nebude muset s aplikací komunikovat přes terminál. Aplikace dovolí zadat vstupní parametry přes grafické uživatelské rozhraní, a to pomocí tlačítek, vstupních polí a dalších standardních kontrolních prvků.

N4 – Zobrazení informací na grafické uživatelské rozhraní. Uživatel bude dostávat veškeré informace o aplikaci na grafické uživatelské rozhraní. Příklad informace: obrázek, text, ukazatel průběhu. Generované soubory budou poskytovány přes souborovéh rozhraní operačního systému.

N5 – Rychlost zpracování. Na rychlost zpracování vstupu a generování výstupu nejsou u aplikace kladeny žádné nároky. Je dána především velikostí vstupu a hodnotami nastavených parametrů generování.



Obrázek 9.1 Model případů užití

9.2 Model případů užití

V této kapitole je popsán a vysvětlen *Model případů užití* (*Use Case Model*). Nejprve jsou označeni *aktéři*, kteří v aplikaci působí. Následuje výčet *případů užití* se zkratkou a popisem každého z nich. Na obrázku 9.1 je zachycen *diagram případů užití* (*Use Case Diagram*). Spolu s tabulkou *plnění funčkních požadavků případy užití* 9.1 tak poskytuje stručný vhled do funkcionality aplikace.

9.2.1 Seznam aktérů

Uživatel je jediným aktérem v systému. Může si nechat s pomocí aplikace nechat vygenerovat hudební notaci a audio nahrávku sonifikace zdrojového obrázku. Dále může upravovat parametry generování, prohlížet si informace o stavu aplikace a generování.

9.2.2 Případy užití

UC1 – Generovat hudební notaci z obrázku. Umožňuje uživateli vygenerovat hudební notaci z obrázku za použití nastavených parametrů. Hudební notací se rozumí soubor ve formátu `PDF`, `Lilypond` nebo `MusicXML`, který obsahuje informace o hudební skladbě (obsazení linek, tempo, předznamenání, uspořádání tónů, výšky tónů, délky tónů, rozdělení do taktů atd.).

UC2 – Generovat audio nahrávku z obrázku. Poskytuje uživateli možnost vygenerovat audio nahrávku z obrázku za použití nastavených parametrů. Audio nahrávka je soubor ve formátu `WAV`, který vznikl syntézou či jiným digitálním způsobem.

UC3 – Generovat `MIDI` z obrázku. Umožňuje uživateli generování `MIDI` souboru za použití nastavených parametrů. `MIDI` soubor obsahuje popis hudebních linek (názvy hudebních nástrojů),

Tabulka 9.1 Tabulka plnění požadavků případy užití

	Požadavky									
Případy užití	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
UC1	⊕	⊕			⊕	⊕	⊕			⊕
UC2	⊕			⊕	⊕	⊕	⊕			⊕
UC3	⊕		⊕		⊕	⊕	⊕			⊕
UC4					⊕				⊕	
UC5					⊕			⊕		

které skladba používá, a informaci o hudebních tónech v nich (časové zařazení, výšku, délku, hlasitost, atd.).

UC4 – Změnit parametry generování. Dává uživateli možnost změnit veškeré parametry generování uvedené jako vstupní pole v tabulce 8.1. Změna parametrů probíhá před sputěním generování.

UC5 – Zastavit proces generování. Dovoluje uživateli zastavit již spuštěné generování hudební notace nebo audio nahrávky z obrázku.

9.3 Architektura

Tato sekce poskytuje vhled do softwarového pojetí architektury. V této části je zmíněn způsob, jakým budou navrženy jednotlivé *vrstvy architektury*, a také, jak bude probíhat komunikace při *zasílání informací o stavu* aplikace během procesu generování hudebního artefaktu.

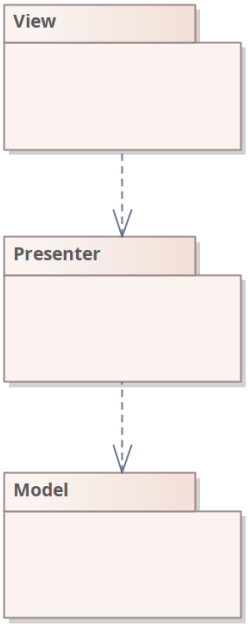
9.3.1 Vrstvy

Protože aplikace počítá s tím, že by ji někdo mohl v budoucnu rozšiřovat, její logická architektura bude navržena tak, aby bylo možné její celistvé části co nejjednodušeji vyměnit. K tomu dobře slouží třívrstvá architektura designovaná dle vzoru `Model-View-Presenter`. Vzor `Model-View-Controller`, který používá třívrstvou architekturu relaxovanou, není použit právě z důvodu zachování co nejméně závislostí mezi vrstvami a také zdůvodu

Dle výše zmíněného vzoru jsou utvořeny vrstvy, jejichž vazby zachycuje obrázek 9.2. Vrstvy představují balíčky aplikace: `view`, `presenter` a `model`. Lze na ně pohlížet[2] jako na funkční celky. Úplně nejvyssší vrstva, `View`, představuje *GUI aplikace*. S nejspodnější vrstvou, `Modelem`, komunikuje přes rozhraní, které poskytuje `Presenter`. `Model` má na starosti výpočty, načítání a ukládání dat do souborového systému. `View` se stará o zpracování uživatelského vstupu a interakci směrem na obrazovku.

V momentě, kdy uživatel interaguje s aplikací, `View` vyhodnotí jeho akci. Dle jejího typu spustí určitou metodu vrstvy `Presenter`. Ta vyhodnotí vstupní data, zkontroluje, jestli jsou validní. V případě, že nejsou, vrátí chybový kód zpět na `View`. Ten ho v podobě chybového hlášení předá uživateli. Je-li všechno v pořádku, `Presenter` spustí příslušnou metodu `Modelu`. `Model` data zpracuje a když je hotov, vrátí výsledek operace `Presenteru`. `Presenter` je převede do vhodné podoby pro `View`, a tu vrátí nejvyšší vrstvě jako výsledek.

[2] I když jsou to balíčky, lze se k nim pro účely popisu chovat jako k třídám, které mají vyhraněnou funkci v systému.



Obrázek 9.2 Architektura založená na vzoru Model-View-Presenter

9.3.2 Zaslání informací o stavu vyšším vrstvám

Vzhledem k funkčním požadavkům *(F5 – Informování o stavu)* bude nutné posílat informace z nejspodnější vrstvy do vrstev vyšších nejen při dokončení výpočtu, ale i v průběhu generování. Je tedy třeba navrhnout takový postup, který nedá vzniknout cyklické závislosti. K tomu se hodí návrhový vzor chování `Observer and subject` *(Pozorovatel a předmět pozorování)*.

Funguje na principu notifikací. *Observer* a *Subject* jsou rozhraní. *Observer* poskytuje předpis metody pro *registraci subjektů*[3] a *Subject* zase předpis metody pro *dostávání notifikací*. Při nějaké události pošle třída implementující rozhraní *Subject* notifikaci všem *pozorovatelům*, kteří jsou u ní zaregistrováni. A to tak, že na nich zavolá metodu pro *dostávání notifikací* a jako argument předá data. Na toto volání reaguje každý pozorovatel dle své povahy, a to díky různé implementaci funkce pro *dostávání notifikací*, vyhodovující účelu třídy pozorovatele. Tímto způsobem lze obejít vznik cyklické závislosti.

Nechť *Presenter* je hlavní třída[4] vrstvy `Presenter`, *GUI* hlavní třída vrstvy *View* a *BasicModel* hlavní třída vrstvy `Model`. Pokud bude třída *Presenter* implementovat rozhraní *Observer* a třída *BasicModel* rozhraní *Subject*, může *BasicModel* na základě výše uvedeného principu posílat *Presenteru* notifikace se zprávou o stavu výpočtu, případně daty, aniž by musel předávat kontrolu někomu jinému. Stejný postup bude použit pro informování *GUI* z místa *Presenteru*.

[3] Případné odstranění pozorovatele z registrovaných subjektů se pro zjednodušení výkladu nebude řešit.
[4] Hlavní třída je v tomto případě popis třídy v balíčku, která zaručuje komunikaci s třídami z jiných balíčků.

Kapitola 10
Implementace

Aplikace byla implementována v programovacím jazyku *Python*. Byl zvolen především díky snadné přenositelnosti mezi operačními systémy a velkému množství knihoven, pro něj dostupných. V této kapitole jsou popsány nejdůležitější části implementace, a to *hlavní fáze výpočtu*, *propojení GUI a výpočetní části* a nakonec *zastavení generování*.

10.1 Hlavní fáze výpočtu

Od okamžiku, kdy dá uživatel pokyn ke generování, proběhne v aplikaci několik hlavních fází výpočtu: *Načtení obrázku, detekce objektů, extrakce objektů, tvorba not, kompozice písněm přiřazení hudebních nástrojů, přiřazení tempa, tvorba notového zápisu, export notového zápisu.* Pořadí jednotlivých fází je stručně zobrazeno na diagramu 10.1.

10.1.1 Načtení obrázku

S využitím knihovny *OpenCV* [57] je obrázek načten ze zdrojového souboru zadaného uživatelem. Data jsou načtena do `numpy array` ve formátu odpovídajícím barevnému modelu `RGB`[1] a následně pomocí metody `cv2.cvtColor` transformována do barevného prostoru `L*a*b*`. Tuto práci má v aplikaci na starosti třída *LabImageLoader*.

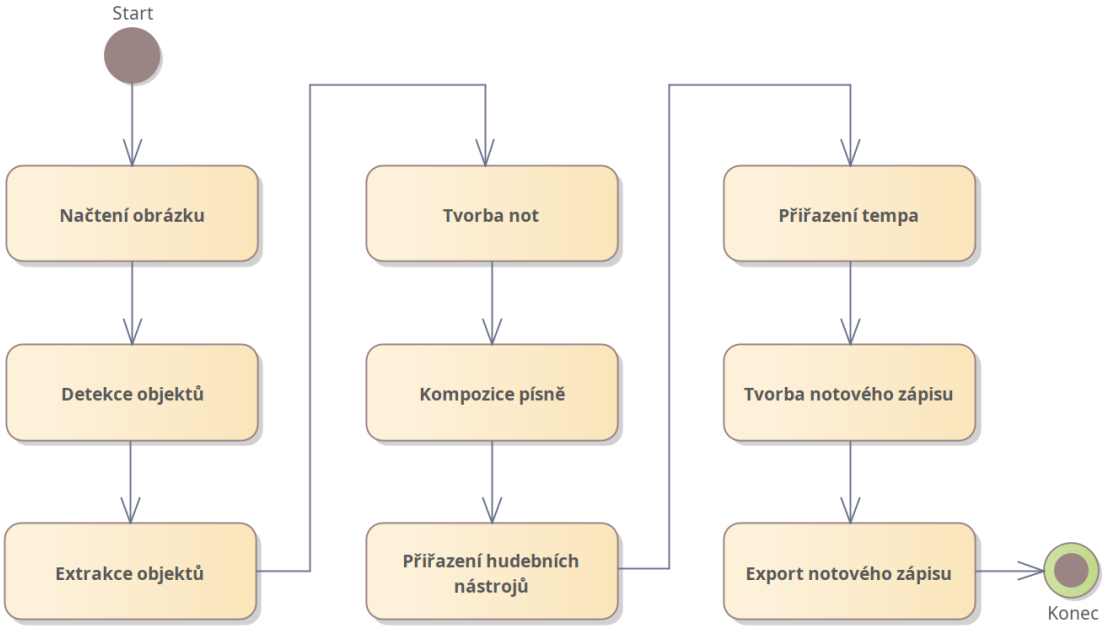
Poté je obrázek předán do prezenteru aplikace, kde je převeden nazpět do formátu `RGB`. Presenter předá obrázková data grafickému rozhraní (instance třídy *GUI*), které změní velikost obrázku tak, aby odpovídala velikosti okna. Následně je na kartě *Original image* ukázán uživateli.

10.1.2 Detekce objektů

Detekce objektů v aplikaci probíhá za pomoci open-source knihovny `ImageAI` [54]. Je používán předtrénovaný model *RetinaNet* [58]. Detektor očekává cestu k obrázku, ze kterého následně detekuje objekty. Zároveň zakreslí do obrázku místo výskytu objektů pomocí modrých obdelníků. Tento výstup je předán uživateli (více v kapitole 10.2).

V případě, že detektor nenalezne žádný objekt, je výskyt objektu simulován. Simulovaný objekt se nachází v místě vycházejícím ze zlatého řezu, blíže popsaného v kapitole *Mapování modalit*. Se

[1] *OpenCV* ukládá obrazová data v pořadí `BGR` namísto `RGB`



Obrázek 10.1 Stručný diagram fází výpočtu

simulovaným objektem je dále nakládáno úplně stejně jako s objektem, který byl detekován pomocí umělé inteligence.

10.1.3 Extrakce objektů

Souřadnice objektů jsou transformovány do podoby vhodné pro ořezávání `numpy array`. Každý objekt je pak na základě svých souřadnic nalezen a jeho obrazová data zkopírována do seznamu obrazových dat objektů. O tuto práci se stará instance třídy *ObjectExtractor*.

V tomto kroku je také vytvořena maska. Místa v obrázku, kde jsou detekované objekty jsou označena výrazem `True` a ostatní výrazem `False`. Tato maska slouží v dalším zpracování jako klíč k rozpoznání co je objekt a co pozadí. Výsledky jsou předány k dalšímu zpracování.

10.1.4 Tvorba not

Tvorba not je jeden z větších procesů, který si zaslouží být popsán podrobněji. Na diagramu 10.4 lze vidět zjednodušené schéma jeho jednotlivých fází: *Výpočet souřadnic mřížky, výpočet minimální rozlohy regionu, přiřazení dat z obrázku a masky, výpočet důležitost objektu, výpočet průměrné světlosti objektu, přiřazení počáteční oktávy, tvorba rytmu, zpracování segmentů, přenesení rytmu na generované akordy, výpočet celkové délky not, uložení not do objektu.* O většinu těchto fází se stará instance třídy *BasicMotiveFactory*.

Výpočet velikosti segmentů je proces, díky němuž aplikace spočítá, jak velké mají být jednotlivé segmenty. Kalkulace vychází ze vstupu uživatele (šířka a výška segmentu v procentech vůči

velikosti zdrojového obrázku).

Výpočet minimální rozlohy regionu určuje na základě minimální délky noty zvolené uživatelem, minimální rozlohu regionu v segmentu. Postup je založený na stejném principu, který byl popsán v kapitole *Mapování modalit* v sekci *Délka tónu*, tedy nejdříve je spočítán poměr nejmenší noty k celému taktu a následně je z tohoto poměru vyvozena minimální rozloha regionu v procentech.

Přiřazení dat z obrázku a masky zaručuje propojení výstupních dat (`numpy array` segmentovaného obrázku) a aktuálním zpracovávaným objektem. Metodě, která se stará o zapsání výstupních dat do obrázku, je tak přiřazeno správné místo v obrázkum, a to pomocí indexace `numpy array`. Díky tomu může být každý objekt zpracováván bez závislosti na pozadí. Vycházeno je ze souřadnic spočítaných v kroku *Extrakce objektů*. V tomto kroku jsou dále přiřazena obrazová data zpracovávaného objektu a pokud je zpracováváno pozadí, je k němu přidána jako parametr vytvořená maska. Vzájemné překrytí objektů není nijak řešeno.

Výpočet důležitost objektu určuje význam objektu na základě jeho velikosti. Čím větší objekt je, tím větší má hudební motiv, který z nej vznikne, důležitost. Velikost je dána počtem pixelů.

Výpočet průměrné světlosti objektu probíhá za přičinění třídy *AggregateImageFunctions*. Jedna z jejích statických funkcí vrací průměrnou světlost vstupních dat, se znalostí, že pixely dat jsou uloženy v souladu s barevným prostorem `L*a*b*`.

Přiřazení počáteční oktávy zajišťuje, že počáteční oktáva hudebního motivu bude reflektovat jeho průměrnou světlost. Ta byla spočítána v předchozím kroku. V této části je tedy lineárně mapována světlost na oktávu. Minimální oktávou je *kontra oktáva* a maximální *čtyřčárkovaná oktáva*. Je vycházeno z toho, že klavír zahraje škálu tónů $\mathtt{A}_2$–$\mathtt{c}^5$ [1], což je bez tří krajních tónů právě mapovaný rozsah.

Tvorba rytmu je proces, jehož výstupem je sekvence tzv. *not abstraktního rytmu* s určenými délkami a informací o tom, zda představují pomlku. Tento proces je řízen instancí třídy *DarkLightRhythmCreator*, která při své práci vychází ze vstupu uživatele. Díky němu ví, zda přiřadit pomlku segmentům tmavším, než je průměrná světlost obrázku, nebo naopak těm světlejším. Tvorba rytmu implementuje princip zmíněný v kapitole *Mapování modalit* v sekci . Z každého obrázku[2] je generován rytmus o 4 taktech.

Zpracování segmentů je klíčová fáze, která je více rozvedena na schématu 10.5. Po každém jednom zpracovaném segmentu jsou data o detekovaných regionech posílána do grafického rozhraní aplikace a zobrazována na obrazovku (více v sekci *Propojení GUI a výpočetní části*).

Melodie je vzniká jako sekvence tzv. *abstraktních tónů*, kdy je jejich výška určena jen formálně (oktávou). Hudební tón kromě toho získá *zařazení do chromatické řady, délku* a *hlasitost*. Přiřazení stupňů durové stupnice spektrálním barvám je zachyceno na obrázku 10.2 a kopíruje zákonitosti popsané v kapitole *Mapování modalit* v části *Zařazení tónu do chromatické řady*. Přiřazení odstínů jednotlivým spektrálním barvám vychází z rozmístění 24 odstínů barev v kruhu `HSV` [59]. Na základě

[2] Jehož výška je větší než 3 pixely. Pokud tomu tak není, rytmus je složen z méně taktů.

Obrázek 10.2 Mapování barev na stupně

Stupeň	Odstín v `HSV`	Jméno barvy
I.	0°–15° a 330°–360°	Červená
II.	15°–44°	Oranžová
III.	44°–75°	Žlutá
IV.	75°–150°	Zelená
V.	150°–195°	Tyrkysová
VI.	195°–252°	Indigo
VII.	252°–330°	Fialová

Obrázek 10.3 Škála odstínů spektrálních barev

mapování je zmenšen rozsah oranžové na 96 % a barvy indigo na 95 %. Rozšířen byl rozsah červené a fialové[3] barvy.

Délka tónu je určená procentuální rozlohou regionu v segmentu. Více je toto přiřazení popsáno v části *Délka tónu*. Hlasitost je přiřazena konstantně. Konkrétní podobu *pro výšku not* stanoví až instance třídy *AbstractionDestroyer*, a to vymezením konstanty *pitch*, určené dle oktávy z `MIDI` protokolu.

Doprovod je tvořen též abstraktní formou. Na základě stupně hudebního tónu a tónické funkce vzniklé ze dvou sousedních regionů, ze kterých byly taktéž vytvořeny tóny melodie, je generován akord. Algoritmus postupuje dle výsledného mapování modalit ze sekce Tvorba akordu a Změny ve funkční harmonii.

Konkretizace doprovodu se skládá ze dvou hlavních kroků: *přenesení rytmu na generované akordy* a *odstranění abstrakce z akordů*. Přenesení rytmu na generované akordy odebírá jednu z úrovní abstrakce generovaných not. Proces si lze představit jako aplikaci jakési masky. V místech, kde jsou v rytmickém předpisu pomlky, objeví se i ve výsledném doprovodu. Je třeba zmínit, že generovaný doprovod předtím žádné pomlky neobsahoval. Pokud je rytmický předpis menší délky, než generovaný doprovod, opakuje se. Odstranění abstrakce z akordů pak probíhá převodem tónů akordů z abstraktní formy do formy specifické.

Výpočet celkové délky not slouží nejen k spočítání, jak dlouho takt hraje, ale jednotlivé noty jsou také obalovány tzv. *notovou jednotkou*. V ní je uložena informace o tom, zda nota představuje akord nebo samostatný hudební tón. Toho je využito později, zejména při tvorbě hudební notace (*score objektu* v knihovně SCAMP).

Uložení do objektu je poslední fází tvorby not. Vygenerované notové jednotky v seznamu jsou zabaleny dle své příslušnosti k hudební lince (doprovod nebo melodie) do objektu třídy *MusicalPart* a uloženy do objektu *hudebního motivu*.

10.1.5 Kompozice písně

Na základě volby uživatele je vybrán jeden ze dvou kompozitorů a na něm zavolána metoda `compose`. Pokud kompozitor nedostal na vstupu žádné motivy, vrátí prázdný seznam hudebních linek. Jednotlivé motivy jsou seřazeny dle své důležitosti (dle velikosti ohraničujícího obdelníku objektu ve scéně).

[3] *purpurová barva* nemůže vzniknout bez spojení červené a fialové, což se u dvou vzdájemně opačných spektrálních barev v barevném spektru nikdy nestane.

Obrázek 10.4 Stručný diagram fází procesu tvorby not

Prvním motivem se vždy stává ten, který vychází z pozadí obrázku. Následně je motiv pozadí rozdělen do tří sekcí: *úvod, spojovací část* a *závěr*. Se zbývajícími motivy je nakládáno dle volby kompozitoru. Práci jednotlivých kompozitorů zajišťují instance tříd *RepeatingMotiveCompositor* a *MotiveDrivenCompositor*. Algoritmus postupuje dle instrukcí uvedených v sekci Struktura písně v kapitole *Mapování modalit*.

10.1.6 Přiřazení hudebních nástrojů

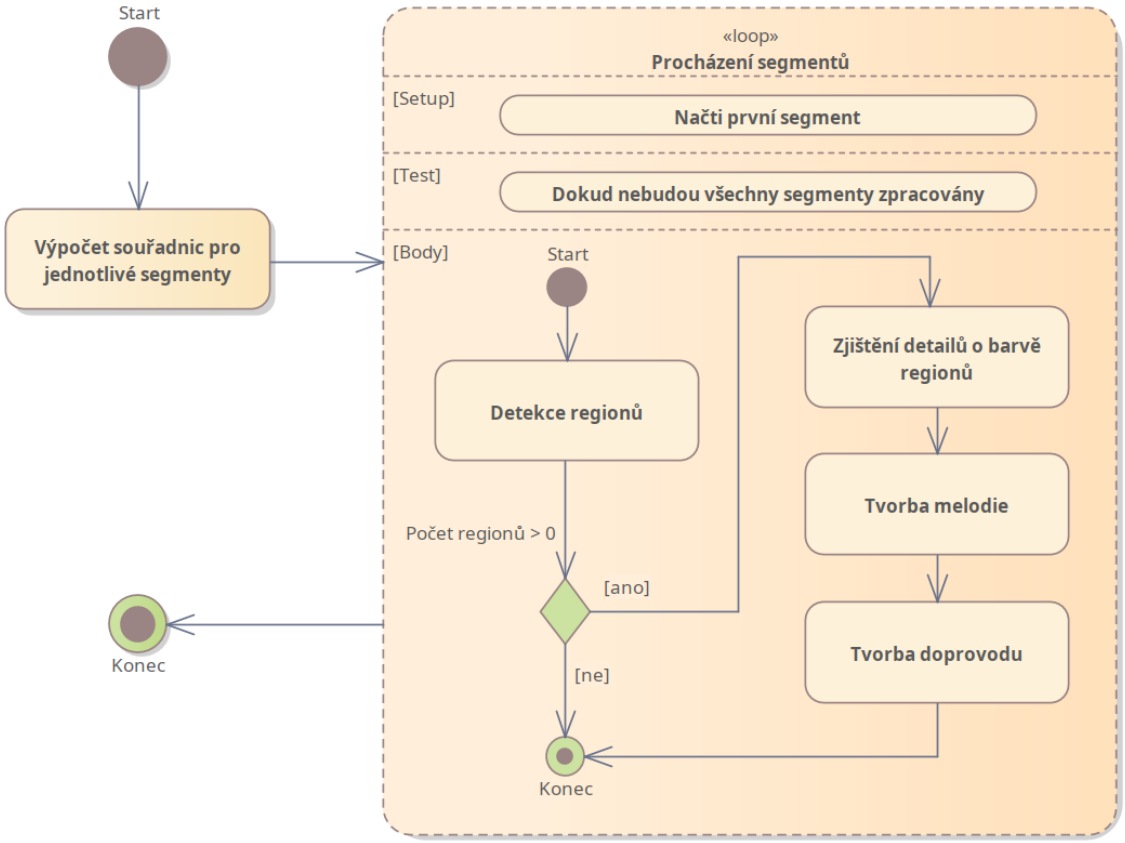
Nástrojová linka doprovodu je zajištěna ve vztahu k celému obrázku, zatímco hudební nástroj melodie ve vztahu k největšímu detekovanému objektu. Určení nástroje probíhá na základě průměrné saturace obrazových dat. Ta jsou z `L*a*b*` převedena do barevného prostoru `HSV`.

Přiřazení se řídí zákonitostmi popsanými v kapitole *Mapování modalit* v části *Hudební nástroje*, se zúžením na nástroje: *viola, flétna, piano* a *elektrická kytara*. Toto zúžení bylo aplikováno po prozkoumání zvuku jednotlivých hudebních nástrojů v volně dostupných zvukových bankách. Mapování je prováděno tak, aby každý z nástrojů dostal přidělený stejně velký rozsah saturace.

10.1.7 Přiřazení tempa

Určení tempa pro hudební skladbu kopíruje princip popsaný v sekci *Tempo* z kapitoly *Mapování modalit*. Implementací se zabývá instance třídy *LightnessTempoDetector*. V závislosti na vstupu uživatele stanoví maximální tempo. To se může pohybovat od hodnoty `40 BPM` do `400 BPM`.

LightnessTempoDetector předpokládá na svém vstupu hodnotu průměrnou svítivost obrázku. Nedostane-li ji, zjistí ji sám, a to pomocí již zmíněné třídy *AggregateImageFunctions*. Průměrná svítivost je na základě hodnot svítivosti jednotlivých pixelů, které jsou ukládány dle předpisu barevném prostoru `L*a*b*`. Výsledek je poté lineárně namapována na rozsah `1`–`maximální tempo`.



Obrázek 10.5 Stručný diagram fází zpracování segmentů

10.1.8 Tvorba notového zápisu

V této části jsou zúročeny výsledky všech předchozích operací. Objekt třídy *ScoreCreator* nejprve určí absolutní cestu k hudebnímu fontu (více v sekci Export notového zápisu). Po přidělení jména aplikace jako hudebního skladatele, určí z *jména obrázku bez přípony* titul skladby:

$$\text{titul skladby} = \texttt{song-result-} + \textit{jméno zdrojového obrázku bez přípony}$$

Následně je vytvořena tzv. `Session`, hlavní řídící objekt knihovny SCAMP [56], a inicializovány hudební linky. Každé lince je přiřazen hudební nástroj a objekt typu `MusicalPart`. Na řídícím objektu jsou posléze spouštěny asynchronní procesy, které představují hrané hudební linky jednotlivých hudebních nástrojů. Zde jsou noty z objektů *MusicalPart* přehrávány buď jako akord, nebo samostatný tón, dle informace v *notové jednotky*.

Když je tzv. *transkripce* dokončena, je její výsledek transformován do tzv. `score objektu`. Tento objekt je výchozím bodem pro export notového zápisu do čitelné podoby, ale také pro generování zvukových souborů.

10.1.9 Export notového zápisu

Skladbu jde exportovat do pěti různých formátů: `Lilypond`, `PDF`, `MusicXML`, `MIDI` a `WAV`. Které formáty skutečně vzniknou, závisí na volbě uživatele, který tento údaj zadá na vstupu. Výsledné soubory jsou pojmenovány dle následujícího vzoru:

$$\text{jméno souboru} = \texttt{song-result-} + \textit{jméno zdrojového obrázku bez přípony} + \textit{přípona}$$

Soubory stejného jména, které již ve výstupní složce existují, jsou při generování přepsány novější verzí. Některé formáty požadují na svém vstupu nejprve formát jiný. Diagram 10.6 tyto závislosti zobrazuje a taktéž naznačuje pořadí generování jednotlivých formátů. Pokud uživatel nechce vygenerovat některou ze závislostí, je generována do složky `.tmp_results` a následně smazána.

Počáteční bodem pro export je `score object`. Z něj jsou pomocí knihovny SCAMP [56] exportován soubory ve formátu `Lilypond` a `MusicXML`. Ze souboru typu `Lilypond` je následně pomocí programu *Lilypond* vyroben soubor ve formátu `PDF`. K vytvoření `MIDI` souboru je využito knihovny *music21* [60]. Pro syntézu a vytvoření zvukové nahrávky ve formátu `WAV` se využívá projektu *Fluidsynth* [61]. O tyto exporty se starají instance tříd implementující rozhraní *ExporterInterface*.

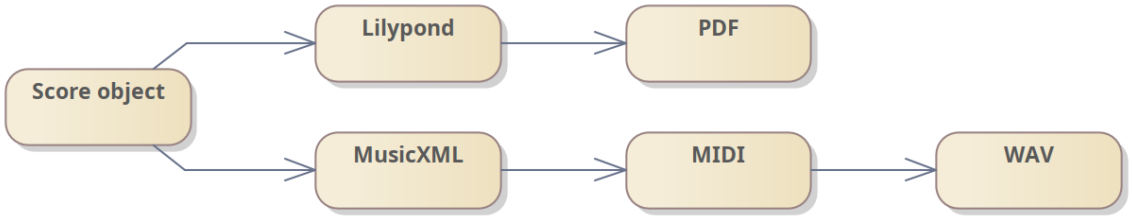
Pro vznik souboru je potřeba soubor hudebních fontů, tzv. *soundfont*, z nějž se následně pomocí syntézy z formátu `MIDI` stane `WAV`. Aplikace používá soundfonty `MuseScore_General.sf3` [62] a `MuseScore_General.sf2` [63], publikované pod `MIT`[4] licencí.

10.2 Propojení GUI a výpočetní části

Jak lze vidět na sekvenčním diagramu 10.7, komunikace zbytku aplikace s GUI probíhá na principu *Subject and observer*[5] za spolupráce dvou vláken. První z vláken pečuje o grafické uživatelské rozhraní a reaguje na vstupy uživatele. Po stisknutí tlačítka *Generate* je vytvořeno druhé vlákno a jeho život začíná validací vstupu uživatele. Následně kopíruje hlavní výpočetní fáze uvedené ve schématu 10.1.

[4] svobodná licence, která vznikla na Massachusettském technologickém institutu (MIT)

[5] *Předmět a pozorovatel* je návrhový vzor chování ze softwarového inženýrství, ve kterém *předmět* při změně svého stavu dá vědět všem *pozorovatelům*, kteří jsou u něj registrováni.



Obrázek 10.6 Pořadí a závislosti exportu

Jak už bylo zmíněno výše, aplikace průběžně posílá do uživatelského rozhraní k zobrazení data. Jakmile jsou data připravena, volá model v roli *předmětu* metodu `notify` svých pozorovatelů, v tomto případě objekt Jakmile výpočetní část dokončí výpočet a data jsou připravená, rozešle je pomocí funkce `notify` svým pozorovatelům.

K přístupu k datům v GUI je implementován zámek, který hlídá, aby data nebyla zaráz v rukou více vláken. V případě zamčeného zámku je volání o přístup k datům blokující. V případě odemčeného zámku jsou data nahrazena novými a zámek odemčen. Zároveň je k událostem připsána událost o připravenosti dat, společně s potřebnými informacemi o datech (která data byla aktualizována). Výpočetní část se po návratu z metody `notify` vrací ke své obvyklé činnosti.

O tom, že byla data aktualizována, se vlákno starající se o GUI aplikace dozví pomocí skenování událostí, které probíhá pravidelně již na úrovni používané knihovny *PySimpleGUI* [55]. Když GUI zjistí, že byla vyvolána událost s aktualizací dat, pokusí se získat přístup k zámku, stejně jako předtím výpočetní vlákno aplikace. V případě úspěšného získání zámku data překopíruje do dočasné proměnné a odemkne zámek. Následně zobrazí data na obrazovku dle jejich povahy. Podobně funguje i výpis informace o aktuálním stavu aplikace. Sekvenční diagram 10.7 zachycuje zobrazení výsledku detekovaných objektů do levé části GUI aplikace.

10.3 Zastavení generování

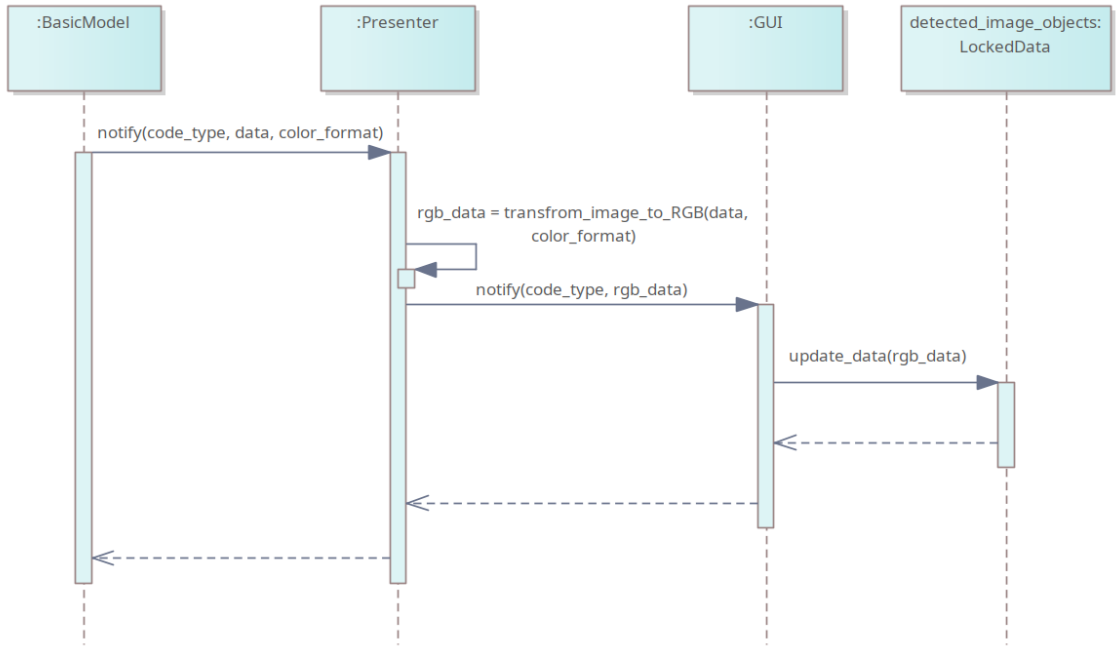
Jedním z funkčních požadavků aplikace bylo, aby se dala zastavit v jakékoliv stadiu generování. Tento problém je v implementaci řešen pomocí nastavování příznaků přerušení napříč vlákny a vyvolání výjimky ve výpočetním vláknu.

Od momentu, kdy uživatel stiskne tlačítko *Stop*, proběhne několik věcí. První, kdo zareaguje, je vlákno starající se o GUI. Vkročí do zóny chráněné zámkem (neúspěch je blokující jako v předchozí kapitole) a nastaví příznak přerušení na `True`. Zámek přenechá odemčený.

Výpočetní vlákno zatím běží, o nastaveném příznaku neví. To se změní v momentě, kdy na své pravidelné obhlídce po každém výpočetním celku příznak zkontroluje. Zjistí-li výpočetní vlákno, že je příznak nastaven, s okamžitou platností vyvolá výjimku typu *InterruptException*. Výjimka probublá až do metody třídy *GUI*, odkud byl výpočet spuštěn. Poté je zachycena a výpočetní vlákno ukončeno.

10.4 Použité technologie a jejich licence

Aplikace vzniklá z této práce, *Rhythm of light*, používá různé části, které implementoval někdo jiný. Součástí těchto knihoven bývají licence, které je autorka povinna uvést, spolu se jmény jejich



Obrázek 10.7 Sekvenční diagram zachycující zobrazení výsledku detekovaných objektů na GUI

autorů. Zároveň oznamuje, že ani jedna ze zmíněných knihoven není její prací. Přesný text licencí je možno nalézt v adresářové struktuře aplikace nebo na webových stránkách autorů.

Numpy

Licence: BSD License (BSD)

Autoři: Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke a Travis E. Oliphant [64]

Využití: Efektivní rozhraní pro práci s poli.

OpenCV python bindings

Licence: MIT License (MIT)

Autoři: Olli-Pekka Heinisuo a další [57]

Využití: Poskytuje rozhraní jazyku *Python* do *OpenCV*.

Midi2audio

Licence: MIT License (MIT)

Autor: Bohumír Zámečník[6]

Využití: Poskytuje jazyku *Python* rozhraní do *FluidSynth* pro systém *Linux*.

Music21

Licence: BSD License (BSD)

Autoři: Michael Scott Cuthbert, The music21 project a další [60]

Využití: Poskytuje konverzi datových formátů. Aplikace *Rhythm of light* využívá převod z formátu `MusicXML` do `MIDI`.

Colormath

Licence: BSD License (BSD)

Autor: Gregory Taylor[7]

Využití: Implementace vzorce o barevném rozdílu *CIE DELTA E* [10].

SCAMP (Suite for Computer-Assisted Music in Python)

Licence: GNU General Public License v3 (GPLv3)

Autor: Marc Evanstein [56]

Využití: Tvorba a úpravy `score objektu`, jednoduché generování hudební notace na základě vstupních dat ve formátu *délka noty, MIDI pitch, hlasitost a hudební nástroj*, případně *seskupení tónů do akordu*.

Clockblocks

Licence: GNU General Public License v3 (GPLv3)

Autor: Marc Evanstein[8]

Využití: Implementace časové osy a synchronizace času generování `score objektu`, využívaného knihovnou SCAMP.

[6] https://github.com/bzamecnik/midi2audio
[7] https://python-colormath.readthedocs.io/en/latest/
[8] https://github.com/MarcTheSpark/clockblocks

Abjad

Licence: GNU General Public License (GPL) (MIT)

Autoři: Trevor Bača, Josiah Wolf Oberholtzer[9]

Využití: Udržuje a dále převádí data ze `score objektu`, generovaného knihovnou SCAMP.

ImageAI

Licence: MIT License (MIT)

Autoři: Moses Olafenwa and John Olafenwa [54]

Využití:

TensorFlow

Licence: Apache Software License (Apache 2.0)

Autor: Google Inc.

Využití: Závislost pro knihovnu *ImageAI*.

PySimpleGUI

Licence: GNU Lesser General Public License v3 or later (LGPLv3+)

Autoři: PySimpleGUI (není uvedeno jméno) [55]

Využití: Snadné a rychlá tvorba grafického uživatelského rozhraní.

Pillow

Licence: Historical Permission Notice and Disclaimer (HPND) (HPND)

Autoři: Alex Clark, Fredrik Lundh a další [10]

Využití: Práce s obrázky (zvětšování, zmenšování) v grafickém uživatelském rozhraní (udržuje si data ve stejném formátu jako knihovna *PySimpleGUI*).

[9] https://abjad.github.io/
[10] https://pillow.readthedocs.io/en/stable/index.html

Fluidsynth

Licence: GNU Lesser General Public License v2.1 (LGPLv2.1)

Autoři: Peter Hanappe, Josh Green, Pedro Lopez-Cabanillas, David Henningsson a další [61]

Použití: Dynamická knihovna je dodávána s programem pro operační systém *Windows*. Její soubory se nachází ve složce `_thirdparty`. Uživatel ji tak může může nahradit požadovanou verzí. Knihovna *SCAMP* požaduje mít pro případ systému *Windows* podobu této knihovny v upravené formě. Jsou to další soubory, tentokráte je okolnosti volání z knihovny SCAMP nutí nacházet se hned vedle spustitelného souboru aplikace *Rhythm of light*. Ani v jednom případě není uživateli omezována výměna těchto souborů a nahrazení jejich upravnou verzí.

Kapitola 11
Testování

Hlavním účelem testování bylo zjistit, zda je návod k instalaci aplikace a jejímu spuštění dostatečný. Dále, jak uživatelé reagují na grafické uživatelské rozhraní aplikace, a otestování funkčnosti aplikace v systémech `Windows 10` a `Manjaro Linux`. Tato kapitola je členěna na několik částí. První z nich popisuje testovací *persony*. Dále se nachází *testovací scénáře* a poslední sekcí je *vyhodnocení testování*.

11.1 Persony

Persony mají za cíl modelovat cílovou skupinu uživatelů, která bude aplikaci používat. Při testování jsou tito uživatelé simulováni pomocí testerů, kteří se chovají dle popisu person.

Nehudebník. Persona představuje muže středního věku, který se tvorbě hudby nikdy nijak nevěnoval. Počítač využívá pouze na čtení elektronických článků a sledování videí na sociálních sítích. Dokáže rozlišit spustitelný soubor od obrázek nebo zvukového souboru.

Hudebnice. Jedná se o ženu ve středním věku, která se v hudbě aktivně angažuje. Zpívá a tancuje. Nezná však noty ani hudební zápis. Počítač používá velmi často, hlavně ke zpracování elektronických dokumentů a brouzdání po internetu. Dokáže rozlišit přípony jednotlivých souborů.

Výtvarnice. Jedná se o mladou ženu, která má zálibu v kreslení obrazů (na počítači i na papír). Počítač dále používá zejména ke komunikaci s vrstevníky, poslouchání hudby a sledování videí. Dokáže rozlišit přípony jednotlivých souborů.

11.2 Scénáře testování

Za účelem otestování aplikace byly vytvořeny tři testovací scénáře: *Instalace a spuštění (Windows 10), Změna parametrů (Windows 10)* a *Zastavení generování (Manjaro)*. U každého ze scénářů se nachází sekce *Odhadovaný čas, účel testování, počáteční bod, koncový bod, instrukce pro testera, očekávané kroky* a *zpráva z testování*, rozdělena dle jednotlivých person. Kvůli velmi dlouhým zprávám z testování budou uvedeni jen některé z nich.

11.2.1 Testovací scénář – Instalace a spuštění (Windows 10)

Odhadovaný čas: 10 min

Účel testování: Otestování, zda návod v souboru `README.txt` dostatečně připraví uživatele, k tomu, aby nainstaloval a spustil aplikaci *Rhythm of light*. Součástí scénáře je i instalace pomocných aplikací, které *Rhythm of light* potřebuje ke svému chodu.

Počáteční bod

Uživatel se nachází v operačním systému *Windows 10*, se spuštěným *průzkumníkem souborů*. K dispozici má soubory aplikace ve složce `Rhythm_of_light` a soubor `README.txt`. Má přístup k internetu a operační systém mu garantuje práva k instalaci aplikací. V systému není nainstalován program *Lilypond*.

Adresářová struktura

```
README.txt
Rhythm_of_light
    Rhythm_of_light.exe
    resources
    další soubory, které jsou pro účely testování nepodstatné
```

Koncový bod

Uživatel se nachází ve spuštěné aplikaci *Rhythm of light*. V operačním systému je nainstalovaná aplikace *Lilypond*.

Instrukce pro testera

1. Přečtěte si návod k instalaci, který se nachází v souboru `README.txt`.
2. Nainstalujte aplikaci *Rhythm of light*. To zahrnuje veškeré pomocné aplikace, které jsou aplikací vyžadovány. Instalační soubory ostatních aplikací můžete uložit do libovolné lokace. Postupujte dle návodu v `README.txt`.
3. Spusťe aplikaci *Rhythm of light*, která se nachází ve složce `Rhythm_of_light`.

Očekávané kroky

1. Uživatel otevře soubor `README.txt`.
2. Přečte si návod v `README.txt`.
3. Otevře webový prohlížeč.
4. Do hledáčku vloží odkaz pro stažení aplikace *Lilypond*, který nalezne v návodu, a potvrdí klávesou `Enter`.
5. Uživatel po načtení stránky klikne na odkaz pro stažení aplikace.
6. V dialogovém okně vybere lokaci uložení instalačního souboru a potvrdí.

7. Uživatel spustí instalační soubor.
8. Potvrdí, že tento soubor není nebezpečím pro systém.
9. Uživatel je proveden instalací programu *Lilypond*.
10. Uživatel restartuje počítač.
11. Po opětovném přihlášení si uživatel nechá zobrazit složku s daty aplikace.
12. Přepne se do složky `Rhythm_of_light`.
13. Uživatel najde spustitelný soubor aplikace s názvem `Rhythm_of_light.exe`.
14. Na tento spustitelný soubor dvakrát poklepe, čímž ho spustí.

Průběh testování

Nehudebník

Uživatel bez problému našel soubor `README.txt` a otevřel ho. Nebyl si jist, co je zač program *Lilypond*. Zkopíroval z `README.txt` název programu a následně ho zkoušel vyhledávat pomocí kláves *CTRL* + *F*. To mělo za následek označení slova v textovém editoru. Když to uživatel viděl, zmáčkl enter. Myslel, že to program spustí. Namísto toho se však smazalo slovo. Uživatel poté pochopil, že to je špatná cesta, a usoudil, že program na počítači zřejmě není.

Zkopíroval si tak odkaz na stažení aplikace do schránky a následně se přepnul na plochu, kde našel odkaz na spuštění webového prohlížeče *Microsoft Edge*. Otevřel ho a odkaz vložil do hledáčku a potvrdil.

Stáhnutí instalačního souboru z oficiálních stránek *Lilypond* bylo velmi velkou zátěží. Uživatel nemohl najít odkaz na stažení na stránce. Stěžoval si na to, že je stránka příliš složitá. Dívá se znovu do `REAMDE.txt`. Po několika neúspěšných pokusech, zahrnujících hledání na stránce, procházení mnoha podsekcí webu, byl moderátorem naveden na instalační soubor.

Protože zvolený prohlížeč *Microsoft Edge* nedůvěřoval instalačnímu souboru, nestáhl ho. Uživatel tuto informaci obdržel a byl z ní nešťastný. Na odkaz klikal několikrát, ale soubor byl pokaždé zablokován. Po několika nesúpěšných pokusech otevřít soubor, požádal o pomoc moderátora. Ten mu ukázal, jak instalační soubor správně stáhnout a ukázal mu lokaci, kde se soubor nachází.

Uživatel soubor s instalací otevřel. Protože systém *Windows* nedůvěroval staženému instalačnímu programu, upozornil uživatele, že ho právě ochránil před hrozbou. Uživatele to nepřekvapilo. Jen poznamenal, že se zřejmě nepůjde spustit. Když byl moderátorem naveden, aby nechal systém instalačnímu programu důvěřovat, spustila se instalace programu *Lilypond*.

Uživatel bez problému, až automaticky, kliká na tlačítka *Next* v instalaci. Po kliknutí na tlačítko *Finish* a uzavření instalačního okna usoudí, že se program nainstaloval. Znovu se podívá do `README.txt` a poznamenává, že je příliš dlouhé. Nevyzná se totiž, u kterého kroku skončil. Když mu moderátor poradí, který krok právě dokončil, bez dalších okolků přejede myší na položku *Start* systému *Windows 10* a spustí restart počítače.

Po přihlášení do systému *Windows 10* a zobrazení příslušných složek v *průzkumníku souborů* moderátorem, pokračuje testování. Uživatel si znovu zobrazí soubor `README.txt`. Poznamená, že je zřejmě připraven pustit aplikaci.

Přepne se do složky `Rhythm_of_light` a říká, že zřejmě hledá nějaký `exe` soubor. Protože je ve složce příliš mnoho souborů, nemůže spustitelný soubor aplikace najít. Tvrdí, že soubor aplikace se ve složce nenachází. Když je moderátorem dotázan, zda opravdu prošel všechny soubory ve složce, znovu se dá do hledání. Tentokrát soubor `Rhythm_of_light.exe` nalezne.

Dvakrát na spustitelný soubor pokliká. Za chvíli se na obrazovce objeví černé okno. Uživatel říká, že se aplikace zřejmě spustila. Po chvíli poznamenává, že se aplikace spouští velmi dlouho. Když se aplikace spustí a uživatel uvidí její grafické rozhraní, poznamená, že je aplikace spuštěna.

Hudebnice

Uživatel si bez problému otevře soubor `README.txt`. Po přečtení kroků předpokládá, že na počítači nainstalovaný program *Lilypond* není. Zkopíruje adresu odkazu na stažení jeho instalace. Klikne na tlačítko *Start* v systému *Windows 10* a vybere prohlížeč *Microsoft Edge*, který se zobrazuje v oblíbených aplikacích. Prohlížeč se spustí.

Uživatel vloží do adresového hledáčku zkopírovaný odkaz. Po otevření webové stránky klikne na modře označený odkaz na stažení souboru. Protože však *webový prohlížeč* nedůvěřuje stahovanému souboru, zablokuje jeho stahování. Uživatel se podivuje chybové hlášce a neví co má dělat dál. Požádá o pomoc moderátora. Ten přesvědčí prohlížeč o tom, že stažení souboru je bezpečné a instalační soubor se stáhne.

Uživatel klikne na stažený soubor v informačním okně *webového prohlížeče*. Objeví se hlášení systému, že právě ochránil uživatele před hrozbou. Uživatel okno chvíli pozoruje a pak klikne na tlačítko *Nespouštět*. Okno se zavře. Uživatel si uvědomí, že to takto zřejmě nemělo být a znovu spouští instalační soubor *Lilypondu*. Tentokrát při systémovém hlášení klikne na prvek *Další informace*. Následně dovolí systému instalaci spustit.

Po objevení *instalačního průvodce* programu *Lilypond* uživatel bez problému následuje instalační instrukce. Po dokončení instalace uživatel zkontroluje soubor `README.txt` a následně za kliknutí na systémové tlačítko *Start* spouští restart systému.

Po přihlášení do systému *Windows 10* a zobrazení příslušných složek v *průzkumníku souborů* moderátorem, pokračuje testování. Uživatel se přepne do složky `Rhythm_of_light` a automaticky hledá `exe` soubor. V mnoha souborech ve složce ho však nemůže najít. Otevírá a zavírá soubor `README.txt`. Znovu se dívá do složky a konečně spustitelný soubor aplikace *Rhythm_of_light* nalezne.

Jednou na něj klikne. Když se nic nestane, zkusí kliknout dvakrát. Poté se objeví konzolové okno. Uživatel vyčkává, co se bude dít. Když se zobrazí grafické rozhraní aplikace, moderátor mu oznámí, že testování je u konce.

Výtvarnice

Uživatel si bez váhání otevře soubor `README.txt`. Není si jist, jak by měl otestovat, zda se aplikace *Lilypond* v systému nachází. Hledá v aktuálně zobrazené složce, to nic nenachází. Nakonec stiskne klávesu *Start* a do vyhledávacího pole v systému napíše *ly*. Zobrazí se soubor `ly.py`. Uživatel přemýšlí, zda našel, to co hledal. Dle přípony však po chvíli usoudí, že to nenašel.

Vrátí se do souboru `README.txt` a tam zkopíruje odkaz na stažení instalačního souboru. Minimalizuje všechna okna a zobrazí plochu. Na ploše spatří odkaz na webový prohlížeč *Google Chrome*, dvakrát na něj klikne. Když se prohlížeč otevře, vloží do URL pole zkopírovaný odkaz a potvrdí.

Na oficiální stránce programu *Lilypond* klikne na modře označený soubor. Soubor se začne stahovat. Uživatel pozoroje stahování dole na liště. Když je stahování souboru dokončeno, uživatel klikne na ikonu se souborem na liště.

Objeví se varování systému *Windows*, že spuštěný program může představovat hrozbu. Uživatel toto okno automaticky zavírá kliknutím na tlačítko *Nespouštět*. Poznamená, že má aplikaci staženou, takže může přistoupit k dalšímu kroku.

Dle scénáře se v *systémovém průzkumníku* přepne do složky `Rhythm_of_light` a hledá `exe` soubor. Když ho nemůže najít, spustí vyhledávání pomocí hledáčku v okně *průzkumníku*. napíše první písmena souboru a systém aplikaci najde. Uživatel na ni pokliká. Na krátkou dobu se otevře

černé konzolové okno a pak zase zmizí. Uživatel poznamenává, že ho to vylekalo.

Následně se rozhodne, že soubor najde ručně. Když ho v adresáři konečně objeví, dvakrát na něj poklepe. Tentokrát se černé konzolové okno objeví na delší dobu. Uživatel říká, že neví, co to má znamenat. Když se po chvíli objeví GUI aplikace, oznámí, že dokončil scénář.

Moderátor se uživatele zeptá, zda si myslí, že všechno, co udělal bylo přesně dle kroků v `README.txt`. Uživatel odpovídá, že zřejmě ano, jinak by se aplikace nespustila. Moderátor uživatele upozorní, že aplikace *Lilypond* nebyla nainstalována a pobídne ho, ať ji zkusí nainstalovat.

Uživatel se tedy znovu dívá do `README.txt`. Dodává, že „tu aplikaci přece stahoval“. Přepne se do okna *webového prohlížeče*. Prochází si text na stránce programu *Lilypond*. Po chvíli mu dojde, že měl aplikaci nainstalovat. Znovu kliká na instalační soubor. Když se objeví hláška o hrozbě, uživatel si ji přečte, chvíli se zamyslí a následně si nechá zobrazit další informace. Klikne na *Přesto spustit*.

Spustí se instalace aplikace, kterou uživatel ochotně řídí tlačítkem *Next*. Když dojde instalace ke konci, uživatel potvrdí tlačítkem *Finish*. Následně chce spouštět aplikaci, když tu si uvědomí, že měl ještě něco udělat. Podívá se do `README.txt` a následně restartuje počítač.

Po přihlášení do systému *Windows 10* a zobrazení příslušných složek v *průzkumníku souborů* moderátorem, pokračuje testování. Uživatel se přepíná do složky spustitelnými soubory aplikace a dvakrát poklepává na `Rhythm_of_light.exe`. Poznamenává, že se aplikace spouští velmi dlouho. Když se aplikace zobrazí v podobě GUI, moderátor ukončí test.

11.2.2 Testovací scénář – Změna parametrů (Windows 10)

Odhadovaný čas: 10 min

Účel testování: Testování srozumitelnosti grafického uživatelského rozhraní při volbě parametrů generování a funkčnost aplikace při jejich různých hodnotách. Dále je testováno prohlížení výsledků generování.

Počáteční bod

Uživatel se nachází v operačním systému *Windows 10*. Otevřené aplikace jsou: *Průzkumník souborů* se dvěma okny. První z oken se nachází v adresáři se spustitelným souborem aplikace. Druhé okno zobrazuje adresářovou strukturu testovacích dat.

Adresářová struktura testovacích dat

```
zdrojove_obrazky
    chleba_stredni.jpg
    skaly_male.jpg
    kone.jpg
vystup
```

Adresářová struktura spustitelných souborů aplikace

```
Rhythm_of_light
    Rhythm_of_light.exe
    resources
    další soubory, které jsou pro účely testování nepodstatné
```

Koncový bod

Ve složce `vystup` se nachází výsledek generování (viz adresářová struktura). Obsah složky, kterou měl zvolit uživatel jako výstupní v instrukci č. 7, zůstává nezměněna. Aplikace je ukončena.

Adresářová struktura testovacích dat

```
zdrojove_obrazky
    chleba_stredni.jpg
    skaly_male.jpg
    kone.jpg
vystup
    song-result-chleba_stredni.png
    song-result-chleba_stredni.pdf
    song-result-chleba_stredni.ly
    song-result-chleba_stredni.xml
    song-result-chleba_stredni.midi
    song-result-chleba_stredni.wav
    song-result-skaly_male.pdf
    song-result-skaly_male.png
```

Instrukce pro testera

1. Spuste aplikaci *Rhythm of light*, která se nachází ve složce `Rhythm_of_light`.
2. Vygenerujte s pomocí aplikace hudební artefakt z obrázku `chleba_stredni.jpg`, nacházející se ve složce `zdrojove_obrazky` tak, aby tónina skladby byla `E dur`, velikost segmentu 15 % šířky a 25 % výšky zdrojového obrázku a maximální tempo 180 `BPM`. Výsledek uložte do složky `vystup`.
3. Zobrazte si výsledek generování.
4. Slovně popište, zda byly při generování detekovány v obrázku nějaké objekty a které to byly.
5. Vygenerujte `PDF` s notovým zápisem z obrázku `skaly_male.jpg`, který se nachází ve složce `zdrojove_obrazky`. Parametry generování mohou být libovolné. Nic dalšího kromě `PDF` souboru (případně segmentovaného obrázku) negenerujte.
6. Zobrazte si výsledek generování a přesvědčte se, že je vygenerováno pouze `PDF` a segmentovaný obrázek.
7. Nechte aplikaci vygenerovat `MusicXML` notaci, a to při nulové přesnosti detekování objektů. Zdrojový obrázek zvolte libovolný, výstupní složku taktéž.
8. Ukončete aplikaci.

Očekávané kroky

1. Uživatel se přepne do složky `Rhythm_of_light`.
2. Klikne na spustitelný soubor aplikace – `Rhythm_of_light.exe`.
3. Uživatel v GUI stiskne tlačítko *Browse* u pole *Source image*.

4. V dialogovém okně vybere lokaci obrázku `chleba_stredni.jpg` a potvrdí.
5. Uživatel klikne na kartu *Notes*.
6. Ve vstupním poli pod popiskem *Key* mačká tlačítko nahoru do doby, než se v poli objeví nápis *E major*.
7. Uživatel klikne na kartu *Segmentation*.
8. Přepne se kurzorem do vstupního pole pod popiskem *Segment width*.
9. Pomocí kláves změní výchozí hodnotu na *15% of source width*
10. Uživatel se přepne kurzorem do vstupního pole pod popiskem *Segment height*.
11. Pomocí kláves změní výchozí hodnotu na *25% of source height*.
12. Uživatel klikne na kartu *Notes*.
13. Přepne se kurzorem do vstupního pole pod popiskem *Maximal tempo*.
14. Pomocí kláves změní výchozí hodnotu na *180*.
15. Uživatel klikne na kartu *General*.
16. Stiskne tlačítko *Browse* u pole *Output folder*.
17. V dialogovém okně vybere lokaci složky `vystup` a potvrdí.
18. Uživatel stiskne tlačítko *Generate*.
19. Počká, až program zobrazí informaci o dokončení generování (*Success 100%*).
20. Přepne se do průzkumníku souborů, kde se nachází složka `vystup`.
21. Otevře ji.
22. Vidí, že se vygenerovalo 5 souborů:

    ```
    song-result-chleba_stredni.png
    song-result-chleba_stredni.pdf
    song-result-chleba_stredni.ly
    song-result-chleba_stredni.xml
    song-result-chleba_stredni.midi
    song-result-chleba_stredni.wav
    ```

23. Vyhodnotí, že generování proběhlo správně.
24. Uživatel se přepne zpět do GUI aplikace.
25. Klikne na kartu *Object detection*.
26. Pojmenuje objekty ukázané v modrých rámečcích.
27. Uživatel stiskne tlačítko *Browse* u pole *Source image*.
28. V dialogovém okně vybere lokaci obrázku `skaly_male.jpg` a potvrdí.

29. Klikne na kartu *Export*.
30. Klikne na všechny zaškrtávací pole kromě pole u popisku *Generate PDF*.
31. Stiskne tlačítko *Generate*.
32. Počká, až program zobrazí informaci o dokončení generování (*Success 100%*).
33. Přepne se do průzkumníku souborů, kde se nachází složka `vystup`.
34. Otevře ji.
35. Vidí, že se vygenerovaly 2 nové soubory:

    ```
    song-result-skaly_male.pdf
    song-result-skaly_male.png
    ```

36. Vyhodnotí, že generování proběhlo správně.
37. Uživatel se přepne zpět do GUI aplikace.
38. Klikne na kartu *Segmentation*.
39. Přepne se kurzorem do vstupního pole pod popiskem *Object detection accuracy*.
40. Pomocí přepsání textu pomocí klávesnice změní výchozí hodnotu na *0%*.
41. Uživatel klikne na kartu *Export*.
42. Klikne na zaškrtávací políčko u popisku *Generate MusicXML*, čímž ho označí jako aktivní.
43. Stiskne tlačítko *Generate*.
44. Na základě zobrazené chybové hlášky vyhodnotí, že aplikace nepodporuje nulovou přesnost u pole *Object detection accuracy*.
45. Pomocí kliknutí na křížek v pravém horním rohu aplikace ukončí program.

Průběh testování

Uživatelé byli informováni o tom, že v případě nejasnosti pokynů nebo překladu anglických výrazů mohou požádat moderátora o pomoc.

Hudebnice

Po přepnutí do složky `Rhythm_of_light` uživatel najde spustitelný soubor aplikace *Rhythm of light* a dvakrát na něj poklepe. Když se načte grafické rozhraní aplikace, přepne se v *průzkumníku souborů* do složky `zdrojove_obrazky`. Pravým tlačítkem myši klikne na soubor obrázku `chleba_stredni.jpg`. Očekává, že tam bude volba *otevřít v programu Rhythm of light*, to však nenastane.

Následně otevře obrázek v systémovém *prohlížeči obrázků*. Přiblížuje a oddaluje obrázek. Poznamenává, že neví, jak dostat obrázek do aplikace. Moderátor uživatele navede zpět do aplikace. Uživatel klikne na tlačítko *Browse* u položky *Source image*. Objeví se dialogové okno *průzkumníku souborů* s pohledem do složky `samples` (součást složky `resources` v adresářové struktuře souborů aplikace). Uživatel klikne nedávná místa v navigačním panelu, je zde zobrazená složka

`testovaci_data`. Přepne se do ní a následně i do složky `zdrojove_obrazky`. Zde vybere soubor `chleba_stredni.jpg` a potvrdí.

V aplikaci klikne uživatel na kartu *Segmentation*. Chce nastavit šířku segmentu v procentech, ale jakmile spatří políčko s popiskem *Object detection accuracy* s údajem v procentech, usoudí, že zřejmě hledá toto pole. Jeho hodnotu nastaví pomocí šipek v GUI na *15 %*. Následuje dotaz na moderátora, jak se anglicky řekne *výška*. Když moderátor odpoví, uživatel se přepne do políčka s popiskem *Segment height* a pomocí šipek nastaví jeho hodnotu na *25 % of source width*.

Uživatel zmíní, že nastavení *tóniny* skladby bude zřejmě umístěno u not. Klikne na kartu *Notes* a následně se přepne do políčka s popiskem *Key*. Zeptá se moderátora, jak se řekne *durová tónina* anglicky. Po jeho odpovědi nejprve stiskne šipku dolů v grafickém rozhraní, když se nic nestane, stiskne tlačítko s šipkou nahoru. Kliká na něj až do doby, než se v poli objeví hodnota *E dur*.

Následně očima hledá políčko pro nastavení *maximálního tempa*. Když narazí na pole s popisem *Maximal tempo*, znovu začne měnit jeho obsah pomocí šipky dolů v GUI. Když je hodnota nastavena na *180*, Přepne se na kartu *General*. Tam stiskne tlačítko *Browse* s popiskem *Output folder*. Zobrazí se dialogové okno *systémového průzkumníku souborů*. Při této příležitosti uživatel hlavní okno aplikace omylem přesune do dolní části, tak, že poslední, co je vidět, je tlačítko *Generate*. Podobně jako u vybírání vstupního souboru se v dialogovém okně dostane do složky `testovaci_data`. Zvolí složku výstup a následně chce pojmenovat výstupní soubor. To mu však systém nedovolí. Upozorní ho, že soubor názvu, který si zvolil, ve složce neexistuje. Uživatele to překvapí a ptá se moderátora, co se stalo. Ten mu vysvětlí, že nevybírá soubor, ale složku.

Uživatel následně potvrdí výběr složky `vystup`. Poté stiskne v aplikaci tlačítko *Generate*. Aplikace spustí proces generování, ale uživatel nevidí ukazatel průběhu generování, takže si myslí, že se nic nestalo. Pak ho napadne uchopit okno aplikace a přesunout ho výše tak, aby viděl na celý program.

To zapříčiní mimo jiné i ukázání ukazatele průběhu a uživatel pochopí, že se něco přece jen děje. Vyčká dokud aplikace na dolní liště neoznámí, že generování dosáhlo *100 %*. Následně si chce zobrazit výsledek, mačká na různé karty v pravé části aplikace. To považuje za zobrazení výsledku. Když moderátor uživatele upozorní, aby se podíval i do výstupní složky, uživatel je překvapen, že se něco do složky vygenerovalo.

V *systémovém prohlížeči souborů* se přepne do složky `testovaci_data` a následně do složky `vystup`. Spatří vygenerované soubory a po jednom je otevírá. U souboru formátu `MusicXML` je překvapen, že se mu otevřel webový prohlížeč „s jakýmsi kódem“. Při zavírání generovaných souborů omylem zavře aplikaci.

Při popisu detekovaných objektů poznamenává, že neví, jestli byly nějaké detekovány. Místo toho, aby se podíval na kartu *Object detection*, začne vyjmenovávat všechny objekty, které sám viděl na obrázku

U dalšího kroku scénáře si uživatel uvědomí, že zavřel aplikaci. Bez jakýchkoliv problémů ji znovu spustí. Zdrojový obrázek `skaly_male.jpg` vybere jako v předchozím případě v dialogovém okně po stisknutí tlačítka *Browse* ze složky `zdrojove_obrazky`. I když nemusí, pomocí dialogového okna volí i výstupní složku, a to stejnou, kterou již v předchozím kroku zadal.

Aniž by nastavil, které formáty má aplikace generovat, stiskne uživatel tlačítko *Generate*. Vyčkává, než aplikace oznámí konec generování. Následně si v *systémovém průzkumníku souborů* zobrazí obsah složky `vystup`. Jsou tam kromě souboru ve formátu `PDF` i další nově vygenerované soubory. Uživatel to nepovažuje za překážku a po jednom si je zobrazuje. Na dotaz moderátora, zda je tedy všechno v pořádku, odpovídá, že ano, načež mu moderátor připomene, že se měl vygenerovat pouze `PDF` soubor a segmentovaný obrázek.

Uživatel pochopí, že zřejmě měl nastavit ještě něco. Vrací se zpět do aplikace. I když nemusí, znovu pomocí dialogového okna vybírá zdrojový obrázek a následně proklikává všechny karty v le-

vém menu, než se dostane ke kartě *Export*. Pochopí, že se jedná o typy souborů, které aplikace generuje. Podiví se, že zde není volba vygenerovat *segmentovaný obrázek*. Postupně kliká na zaškrtávací pole s popiskem formátů, čímž je deaktivuje. Aktivní nechá jen zašktrvávací pole u formátu `PDF`.

Následně stiskne tlačítko *Generate*. Po dokončení generování se podívá do složky `vystup` a oznámí, že tentokrát se vygenerovalo vše správně. Když se ho moderátor dotáže, jak to poznal, když ve složce jsou i soubory jiných typů, odpovídá, že dle času vytvoření.

Při dalším kroku uživatel tentokrát nechává parametry *zdrojového obrázku* a *výstupní složky* tak, jak jsou. Pole pro změnu *přesnosti detekce objektů* nemůže nalézt. Když spatří pole s popiskem *Object detection accuracy*, poznamenává, že to zřejmě není ono, protože tam je procento. Pomocí tlačítek nakonec nastaví toto pole na hodnotu *1 %*, nižší hodnotu pomocí tlačítek už zadat nemůže.

Následně se přepne kartu *Export*. Zde deaktivuje zaškrtávací políčko u popisku *Generovat PDF* a naopak zaktivuje zaškrtávací políčko u popisku *Generovat MusicXML*. Pomocí tlačítka *Generate* spustí generování.

Uživatel si postěžuje, že se aplikace zřejmě sekla (generování trvá příliš dlouho). Moderátor se ho zeptá, z čeho tak soudí. Uživatel odpovídá, že ukazatel průběhu je již delší dobu na *33 %*. Když aplikace dokončí výsledek, uživatel ji chce ukončit. Nejprve klikne na tlačítko *Stop*, a teprve pak na křížek v pravém horním rohu. Aplikace je ukončena.

Výtvarnice

Uživatel dvakrát poklepe na `Rhythm_of_light.exe`, spustí se aplikace *Rhythm_of_light*. Následně si upraví velikost oken *průzkumníku souborů* tak, aby viděl na obě složky. V testovacích souborech najde soubor `chleba_stredni.jpg` a otevře ho v *prohlížeči obrázků*. Zkouší na obrázek klikat pravým tlačítkem myši a hledá možnost, jak ho otevřít v aplikaci.

Nakonec se vrací do okna aplikace a kliká na tlačítko *Browse* u popisku *Source image*. Otevře se dialogové okno *průzkumníku souborů* s lokací ukázkových obrázků aplikace. Uživatel se podívá, v jaké složce se nachází testovací data a následně se tam v dialogovém okně dokliká. Vybere požadovaný soubor a výběr potvrdí.

Uživatel kliká na karty v levém menu aplikace, než se dostane na kartu *Notes*. Následuje dotaz na moderátora, jak se řekne *tónina* anglicky. Po odpovědi se uživatel přepne do políčka s popiskem *Key*. Pomocí klávesnice tam vpíše `E dur`.

Následně se přepne na kartu *Segmentation*. U první položky se zastaví, protože vidí v hodnotě procento. Do pole vpíše pomocí klávesnice hodnotu *15 %*. Když chce zadat výšku segmentu, zarazí se. Zeptá se moderátora, jak se řekne anglicky *výška*. Po odpovědi nastaví hodnotu v poli u *Object detection accuracy* na *20 %*, s poznámkou, že „to dává zpátky“.

Poté se přepne do pole *Segment width* a pomocí klávesnice změní hodnotu na *15 % of source width*. Podobně postupuje u pole *Segment width*, tentokrát s hodnotou *25 % of source height*. Uživatel se dále přepne do pole s popiskem *Maximal tempo* a vpíše hodnotu *180 BPM*.

Následně se přepne zpátky na kartu *General* a ujistí se, jestli *output folder* znamená *výstupní složka*. Poté klikne na tlačítko *Browse* pod popiskem. Stejným způsobem jako při výběru vstupního obrázku se přepne do složky `vystup`.

Poté zmáčkne tlačítko *Generate*. Program ohlásí chybu – nevalidní vstup. Uživatel se podiví, co zadal špatně a pak proklikává karty *About, Export*, než se dostane na kartu *Notes*. Tam smaže slovo *dur* u tóniny a nechá tam pouze písmeno *E*. Znovu stiskne tlačítko pro generování. Program ohlásí stejnou chybu. Uživatel se následně přepne do pole *Key* na kartě *Notes*. Tentokrát zkusí měnit hodnotu šipkami. Dokliká až na hodnotu *E dur* a poznamenává, že by hlášení chyb mohlo být přesnější.

Stiskne tlačítko *Generate*. Tentokrát se ohlásí pouze jeden nevalidní vstup. Uživatel se vrací

na kartu *Notes* a maže slovo *BPM* v políčku *Maximal tempo*. Poté znovu spustí generování.

Zdá se, že si oddechl, že program konečně něco dělá. Trpělivě čeká, až program dokončí generování. Po chvíli však poznamenává, že generování trvá příliš dlouho. Když se na panelu ukáže hláška *100 % Success*, uživatel si chce prohlédnout výsledek programu. V levém části programu kliká postupně na jednotlivé karty. Když se moderátor dotáže, zda si uživatel prohlédl všechny výsledky generování, tvrdí, že ano. Moderátor mu připomene, že při generování zadával výstupní složku, ať se tam podívá. Uživatel se přepne do okna *průzkumníku souborů* a vzpomíná, jakou výstupní složku tam zadával. Podívá se do scénáře a následně se v *průzkumníku* dostane do složky `vystup`. Po jednom otevírá generované soubory.

Na otázku, které objekty byly v obrázku detekované pouze krčí rameny. Dále dle scénáře mačká tlačítko *Browse* u políčka *Source image* a přepíná se do složky `zdrojove_obrazky`. Zde vybere `skaly_male.jpg` a potvrdí. Protože je ve scénáři zmíněno, že parametry generování mohou být libovolné, uživatel proklikává karty a šipkami náhodně mění hodnoty. Následně se přepíná na kartu *About* a následně *Export*. Odšrtává všechna políčka, až na to s popisem *Generate PDF*. Následně spouští generování.

Když se po dokončení generování dívá do složky `vystup`, je překvapen, že je zde kromě `PDF` také segmentovaný obrázek. Říká moderátorovi, že je to zřejmě špatně, protože se mělo vygenerovat pouze `PDF`. Moderátor mu ukáže pasáž ze scénáře a vysvětlí mu, že je vše v pořádku. Uživatel tedy pokračuje dál.

Uživatel v kartě *Export* odznačí generování *PDF* a místo toho zašktrne generování *MusicXML*. Nechápe, co se myslí *nulovou přesností detekování objektů*. Zeptá se moderátora, jak se tato vlastnost nazývá anglicky. Po odpovědi se pomalu přepíná mezi kartami *Notes* a *Segmentation* a hledá políčko s uvedeným popisem. Následně do něj píše hodnotu *0 %*. Spouští generování.

Když program skončí s chybou, přepne se nazpátek na kartu *Segmentation* a nastavuje políčko *Object detection accuracy* pomocí šipky dolů. Nejmenší hodnota, která lze zadat, je *1 %*. Uživatel vyhodnotí, že aplikace nulovou přesnost detekce objektů zřejmě nedovoluje. Moderátor to potvrdí a uživatel následně kliká na křížek v pravém rohu aplikace.

11.2.3 Testovací scénář – Zastavení generování (Manjaro)

Odhadovaný čas: 10 min

Účel testování: Testování zastavení procesu generování a toho, zda uživatel rozumí tomu, že to může udělat.

Počáteční bod

Uživatel se nachází v operačním systému *Manjaro Linux*. Otevřené aplikace jsou: *Dolphin* (*souborový prohlížeč*) se dvěma okny. První z oken se nachází v adresáři se spustitelným souborem aplikace. Druhé okno zobrazuje adresářovou strukturu testovacích dat.

Adresářová struktura testovacích dat

```
zdrojove_obrazky
    chleba_stredni.jpg
    skaly_male.jpg
    kone.jpg
vystup
```

Adresářová struktura spustitelných souborů aplikace

```
Rhythm_of_light
    Rhythm_of_light
    resources
    další soubory, které jsou pro účely testování nepodstatné
```

Koncový bod

Ve složce `vystup` se nachází výsledek generování (viz adresářová struktura). Aplikace je ukončena.

Adresářová struktura testovacích dat

```
zdrojove_obrazky
    chleba_stredni.jpg
    skaly_male.jpg
    kone.jpg
vystup
    song-result-kone.png
    song-result-kone.midi
```

Instrukce pro testera

1. Spusťe aplikaci *Rhythm of light*, která se nachází ve složce `Rhythm_of_light`.
2. Ze složky testovacích dat (`zdrojove_obrazky`) vyberte obrázek `kone.jpg` a nechejte aplikaci vygenerovat pouze `MIDI` soubor (a segmentovaný obrázek). Parametry volte libovolně. Výstupní složku zvolte složku `vystup`.
3. Zastavte generování.
4. Slovně popište, v jaké fázi jste aplikaci zastavil(a).
5. Upravte parametry: *šířka segmentu* bude 52 % a *výška segmentu* 42 %.
6. Znovu generování spusťte.
7. Zobrazte si výsledek generování.
8. Ukončete aplikaci.

Očekávané kroky

1. Uživatel se přepne do složky `Rhythm_of_light`.
2. Klikne na spustitelný soubor aplikace – `Rhythm_of_light`.
3. Uživatel v GUI stiskne tlačítko *Browse* u pole *Source image*.
4. V dialogovém okně vybere lokaci obrázku `kone.jpg` a potvrdí.
5. Uživatel klikne na kartu *Export*.
6. Klikne na všechny zaškrtávací pole kromě pole u popisku *Generate MIDI*.
7. Uživatel klikne na kartu *General*.

8. Stiskne tlačítko *Browse* u pole *Output folder*.
9. V dialogovém okně vybere lokaci složky `vystup` a potvrdí.
10. Zmáčkne tlačítko *Generate*.
11. Uživatel stiskne tlačítko *Stop*.
12. Přečte z informační hlášky (dole pod tlačítkami *Generate* a *Stop*), ve kterém kroku generování proces zastavil.
13. Uživatel klikne na kartu *Segmentation*.
14. Přepne se kurzorem do vstupního pole pod popiskem *Segment width*.
15. Pomocí kláves změní výchozí hodnotu na *52% of source width*
16. Uživatel se přepne kurzorem do vstupního pole pod popiskem *Segment height*.
17. Pomocí kláves změní výchozí hodnotu na *42% of source height*.
18. Uživatel znovu stiskne tlačítko *Generate*.
19. Počká, až program zobrazí informaci o dokončení generování (*Success 100%*).
20. Přepne se do průzkumníku souborů, kde se nachází složka `vystup`.
21. Otevře ji.
22. Vidí, že se vygenerovaly 2 soubory:

```
song-result-kone.png
song-result-kone.midi
```

23. Vyhodnotí, že generování proběhlo správně.
24. Uživatel se přepne zpět do GUI aplikace.
25. Pomocí kliknutí na křížek v pravém horním rohu aplikace ukončí program.

Průběh testování

Hudebnice

Uživatel nalezl spustitelný soubor aplikace `Rhythm_of_light` a jednou na něj klikl. Na obrazovce se zatím nic nedělo, takže měl pocit, že aplikaci nespoustil. Poklikal na aplikaci tedy dvakrát. Vzápětí se spustilo grafické rozhraní aplikace z prvního kliknutí. Následovalo spuštění dalších dvou instancí. To uživatele překvapilo. Přebytečná okna aplikace zavřel a nechal si jen jedno.

V grafickém rozhraní kliknul na tlačítko *Browse* u popisku *Source image*. Zobrazilo se dialogové okno *systémového průzkumníku souborů* ve složce `samples` (součást složky `resources` v adresářové struktuře souborů aplikace). Tentokrát se zde nenachází žádný navigační panel, který by ukazoval *poslední navštívená místa*. Uživatel je zmatený, kde má testovací data nalézt. Různě se přepíná ve složkách. Nakonec dialogové okno zavře.

Přepne se do okna *adresářového průzkumníku*, kde má otevřenou složku s testovacími daty. Přepne se do složky `zdrojove_obrazky` a pravým tlačítkem myši klikne na soubor `kone.jpg`. Zobrazí se nabídka. Uživatel vybere volbu *Zkopírovat*.

Následně se vrátí do okna aplikace, vybere obsah pole u popisku *Source image* a pomocí klávesy *Delete* jej smaže. Následně pomocí kláves `CTRL + V` vloží obsah schránky. V poli se objeví cesta k požadovanému obrázku.

Uživatel se přepne na kartu *Export* a deaktivuje všechna zaškrtávací políčka, až na to, u kterého se nachází nápis *Generate MIDI*. Následně se vrátí na kartu *General* a stiskne tlačítko *Browse* u popisku *Output folder*. Složku `vystup` opět nemůže najít, a tak volí stejný postup jako při zadávání zdrojového obrázku (kopírováním a vložením cesty).

Stiskne tlačítko *Generate*. Po přečtení dalšího kroku ve scénáři, mačká v aplikaci tlačítko *Stop*. Podívá se a ukáže na *spodní lištu* aplikace, kde visí popis *stavu* aplikace. Komentuje jej slovy *„33 procent, creating notes“*.

Přepne se na kartu *Segmentation*, kde se přepne do pole s popiskem *Segment width*, s dotazem na moderátora, zda slovo *„width“* znamená *šířka*. Po odpovědi použije šipky v grafickém uživatelském rozhraní k nastavení hodnoty na *52 % of source width*. Následně se přepne do pole popiskem *Segment height* a opět pomocí šipek v GUI změní jeho hodnotu na *42 % of source height*.

Stiskne tlačítko *Generate*. Vzhledem k tomu, že obrázek `kone.jpg` je relativně velký, provádí aplikace výpočet dlouho. Uživatel znovu poznamenává, že se aplikace zřejmě zasekla. Po dokončení generování se přepne do složky `vystup` v *systémovém průzkumníku souborů* a poklepe na vygenerovaný soubor `song-result-kone.midi`. Otevře se nástroj *MuseScore*. Uživatel chvíli hledá, čím by zobrazené noty přehrál, až nalezne symbol trojůhleníku, běžně používaný pro přehrávání. Klikne na něj. Aplikace začne přehrávat generovaný hudební artefakt. Po chvíli aplikaci zavře. Vrací se do aplikace *Rhythm of light*, kterou ukončuje křížkem v pravém rohu okna.

11.3 Vyhodnocení testování

Během testování prototypu byla objevena řada nedostatků aplikace. V následující sekci budou po jednom rozebrány a přidán návrh, jak by je šlo vyřešit. Nedostatky jsou řazeny do dvou kategorií: *Instalace a spuštění* a *Funkčnost aplikace*.

11.3.1 Instalace a spuštění

- Pro všechny uživatele bylo velmi náročné stáhnout instalační soubor programu *Lilypond*. Nejen, že webový prohlížeč *Microsoft Edge* stažení tohoto souboru blokoval, ale také systém *Windows* označil instalaci programu *Lilypond* za hrozbu. Uživatelé se tak mohli cítit být kroky v souboru `README.txt` ohroženi. Těmto situacím by šlo předejít přidáním dodatečné informace o bezpečnosti těchto souborů a upozornění, že zmíněné situace mohou nastat. Dále by chtělo rozšířit instalační návod, aby uživatele přesně navedl, který ze souborů na stránce *Lilypond* si má stáhnout.
- V systému linux je spuštění aplikace problematické. Uživatelé po spuštění souboru nebyli dostatečně informováni o průběhu spuštění aplikace, takže si mysleli, že se nespustila nebo že nefunguje. V systému windows byl problém podobný, jen se alespoň zobrazilo konzolové okno, takže uživatel nabyl pocitu, že se děje alespoň něco. Tuto situaci by mohl grafický ukazatel průběhu spouštění, který by se objevil ihned po kliknutí na spustitelný soubor aplikace. dlouhou dobu se spouští
- Všichni testeři si stěžovali, že položek ve složce, kde se nachází spustitelný soubor aplikace, je příliš mnoho. To jim zhoršovalo orientaci a snižovalo úspěšnost spuštění aplikace. Tento problém by šel vyřešit vytvořením zástupce na ploše, který by odkazoval na spustitelný soubor aplikace, v rámci instalace. Další možností je přidat zástupce na aplikaci do složky vyšší úrovně.

11.3.2 Funkčnost aplikace

- Mnozí uživatelé jsou zvyklí na chování *drag & drop*. Usnadnilo by používání aplikace, kdyby tento způsob výběru souborů také uměla.
- Uživatele mátlo, že po vybrání vstupního obrázku nebyl rovnou zobrazen v aplikaci. Kdyby aplikace reagovala na vstup uživatele v reálném čase, zřejmě by to ulehčilo čitelnost její komunikace.
- Nastavování vstupních hodnot pomocí šipek, a ne klávesnice, může být pomalé. Uživatelé by tak měli být upozorněni, že při zadávání vstupu mohou používat jak tlačítka na grafické prvku, tak vpisování hodnot pomocí klávesnice.
- Uživatelé si v aplikaci často pletli výstupní složku se souborem. Hodilo by se, kdyby uživatel mohl určit jméno výstupního souboru.
- Uživatelé většinou vůbec nevěděli, kde najít výsledek generování. Aplikaci by tak pomohlo tlačítko *zobrazení výsledku*.
- Ani jeden z uživatelů nepoznal, jakým způsobem aplikace informuje o detekovaných objektech. Hodilo by se přidat popis, že aplikace označuje detekované objekty v obraze pomocí modrých obdelníků.
- Jeden z uživatelů při téměř každém generování nastavoval výstupní složku a obrázek na stejnou hodnotu, jaká v aplikaci již byla zadaná. To může symbolizovat dvě věci. Buď to, že nevěděl, jaká hodnota tam je zadaná. V druhém případě si uživatel nebyl jist, zda aplikace uchovává jeho minulý vstup. V obou případech by měla aplikace zlepšit svůj způsob informování uživatele.
- Uživatelé byli překvapeni, že v nastavení exportu není možné nastavit všechny typy souborů, které aplikace generuje. Řešením je přidat možnost (ne)generovat segmentovaný obrázek.
- U nastavení parametru *Object detection accuracy* uživatele mátlo procento, které bylo za číselnou hodnotou. Tento parametr by si zřejmě zasloužil lepší popis. V celé aplikaci by bylo dobré přidat delší vysvětlivky jednotlivých polí.
- V případě delšího času generování měli uživatelé pocit, že aplikace přestala odpovídat. To lze vyřešit buď přidáním upozornění, že generování může trvat i déle. Další možností je přidání dalšího ukazatele průběhu, který bude ilustrovat průběh dílčích úloh a uživatel tak bude dostávat informaci o stavu aplikace častěji.

Kapitola 12
Budoucí rozšíření

Python je jazyk, který v dnešní době používá mnoho vývojářů. Už z tohoto důvodu, a také z důvodu třívrstvé architektury a používání abstrakce v návrhu tříd, by rozšíření aplikace nemělo být příliš težké. Při implementaci a designu aplikace přicházelo mnoho nápadů, na které v práci nezbyl čas a prostor. Zde je krátký výčet těch nejzajímavějších.

12.1 Po hudební stránce

Bicí linka by byla vhodným doplněním skladby. Upevnila by se tak ještě více rytmická funkce doprovodu. Navíc by tím skladba dostala větší hloubku a zdála by se být pestřejší.

Zlepšení kompozice. Prosté opakování motivů je ve skladbě málo časté. S motivy je běžné si hrát, různě je přetvářet. Dále lze například přenášet rytmus či dokonce jednotlivé témata z melodie do doprovodu.

Variabilní počet taktů v rytmu. Rytmus nemá vždy jen 4 takty. Především by se mohl lišit počet taktů v jednotlivých motivech, což by vyzdvihlo jejich různorodost a přiblížilo je tak více k smyslu *příznačného motivu*.

Vyzkoušení ostatních relací. Až budou dostupné technologie nebo prostředky na implementaci dalších vazeb mezi obrazem a zvukem, uvedených v kapitole *Mapování modalit*, bylo by dobré tyto relace otestovat.

12.2 Po stránce zpracování

Průhlednost obrázku je jednou z věcí, kterou aplikace neřeší. Namísto rozpoznání průhlednosti, překryje průhledné oblasti černou barvou. To jistým způsobem zkresluje výsledek (průměrná světlost, nechtěná detekce objektu).

Paralelizace by byla vhodným zrychlením celého procesu generování. Program by mohl zpracovávat více segmentů naráz, například pomocí vláken, nebo paralelizaci využít při exportu do více souborů.

Přesnější detekce objektů s využitím vlastního natrénovaného modelu, je dalším možným rozšířením aplikace. Předtrénováno by bylo několik modelů, dle různých katerogií trénovacích dat. Uživatel by následně mohl vybírat, která data nejlépe odpovídají obrázku na vstupu.

Odstranění pozadí z detekovaného objektu by přidalo algoritmu větší cit pro přiblížení sémantiky obrázku. Objekty by tak získaly namísto ohraničujícího obdelníku svůj pravý tvar. Využít by se dalo některé z placených API.

Větší výběr zvukových fontů nebo nahrávání vlastních soundfontů do aplikace, by zvýšilo šanci, že nahrávka bude znít věrohodněji, zejména v případě kvalitnějších soundfontů. Dále by se mohly hudební nástroje vybírat z různých soundfontů, což by při stejném ladění zvukových nahrávek rozšířilo vertiáklní hudební kompozici.

Video zobrazující aktuálně přehrávané regiony by mohlo ukázat uživateli, jaký je vztah hudební skladby k obrázku v časové ose. Dostal by tak přímý vhled do relací mezi obrazem a hudbou.

Kapitola 13
Závěr

Práce byla zahájena literární rešerší, díky které byl čtenář seznámen s odbornými pojmy akustiky, hudební teorie a oblasti barevných prostorů, aby následně porozuměl objektivnímu popisu projektů podobných tématu sonifikace obrazu.

Následovala subjektivní analýza projektů a jejich podrobné srovnání mezi sebou, zahrnující hodnocení tvorby relací, oblasti hudebních nástrojů, vstupu uživatele a výstupu aplikace. Z výsledků porovnání byly vyvozeny závěry pro návrh prototypu.

Jako dalším bodem praktické části bylo hledání vhodného mapovování modalit z vizuální do zvukové sféry, přihlížející k dosavadní analýze předchozích projektů. Jako vhodným řešením se nakonec ukázalo několik relací, vycházejích především z barevných vlastností obrázku, jeho saturace a světlosti.

Pro nalezené řešení bylo třeba najít vhodné technologické řešení, a tak byla provedena analýza technologických možností s důrazem na uživatelskou přívětivost a přenositelnost. Jako programovací jazyk byl zvolen Python a jako klíčová technologie knihovna SCAMP, OpenCV, PySimpleGUI a ImageAI. K tomuto rozhodnutí bylo přihlédnuto v softwarovém návrhu, který položil teoretický základ pro následnou implementaci. V rámci něj bylo vypracována analýza funkčních a nefunkčních požadavků, model případů užití a podoba architektury.

Při implementaci byly největší problémy s zabalením pythonového balíčku, kde bylo potřeba vyzkoušet různé metody. Dále bylo třeba přidat aplikační rozhraní pro zjednodušení exportu souborů, protože se během implementace objevila chyba v knihovne `abjad`.

Výsledným prototypem je program s grafickým uživatelským rozhraním, který ze zadaného obrázku vygeneruje hudební artefakt se dvěma nástrojovými linkami - melodickou a doprovodnou. Program pojmenovaný *Rhythm of light*, dále generuje hudební notaci s možností uložení do `Lilypond`, `PDF` nebo do `MusicXML`, dále `MIDI` soubor a audio nahrávku ve formátu `WAV`. Co se týče rozhraní, uživatel má na výběr z několika parametrů generování, které může ovládat pomocí grafických tlačítek. Stěžejním vstupem je zdrojový obrázek a výstupní složka. K dispozici je také nastavení segmentace obrázku, tvorby not a možnost zobrazení detekovaných objektů.

Poslední fází bylo testování prototypu, které použilo navržené persony nehudebníka, hudebnice a výtvarnice a příslušné testovací scénáře. Probíhalo za účasti tří uživatelů, kteří přispěli k nalezení některých nedostatků, kterých bude moci být využito při dalším vývoji aplikace.

Na tuto práci lze navázat mnoha způsoby. Jedním z nich je rozšíření počtu nástrojových linek, které mohou hrát naráz. Dále přidáním linky bicích nástrojů, zapojením více kompozičních stylů nebo grafickým zvýrazněním částí obrázku, které jsou přehrávány.

Příloha A

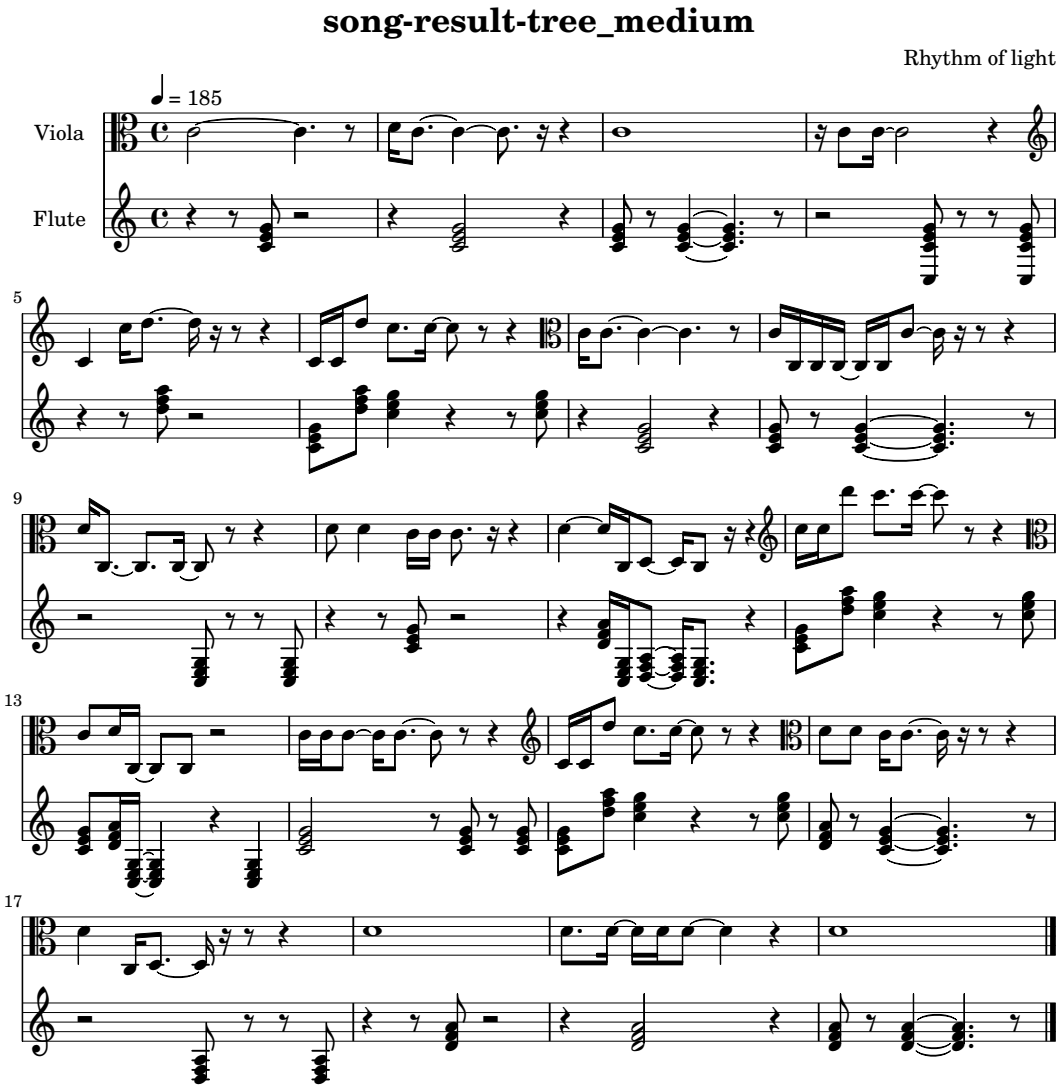
Ukázky výsledků práce

V této kapitole budou ukázány výsledky práce. První bude *originální obrázek*, následně obrázek *segmentovaný* a jako poslední bude první strana z vygenerovaného `PDF` souboru s hudební notací.

Všechny fotografie uvedené zde jsou dílem autorky, stejně tak jako přidané jako ukázkové obrázky v aplikaci. Uvedené výsledky lze nalézt také ve složce `demos` v adresářové struktuře přiloženého media.

Obrázek A.1 Originální obrázek – Strom (střední velikost)

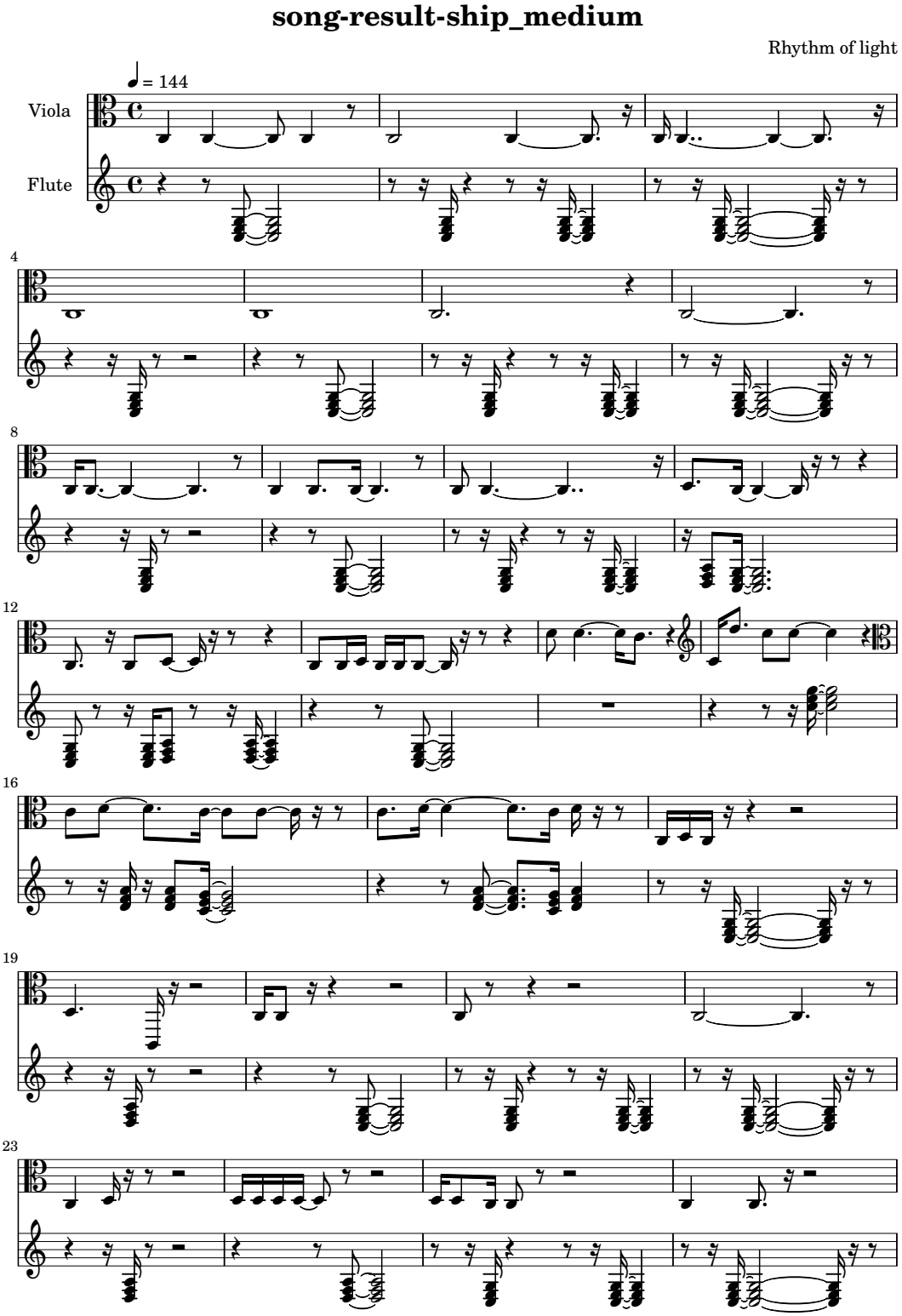
Obrázek A.2 Segmentovaný obrázek – Strom (střední velikost)



Obrázek A.3 Výsledný notový zápis – Strom (střední velikost)

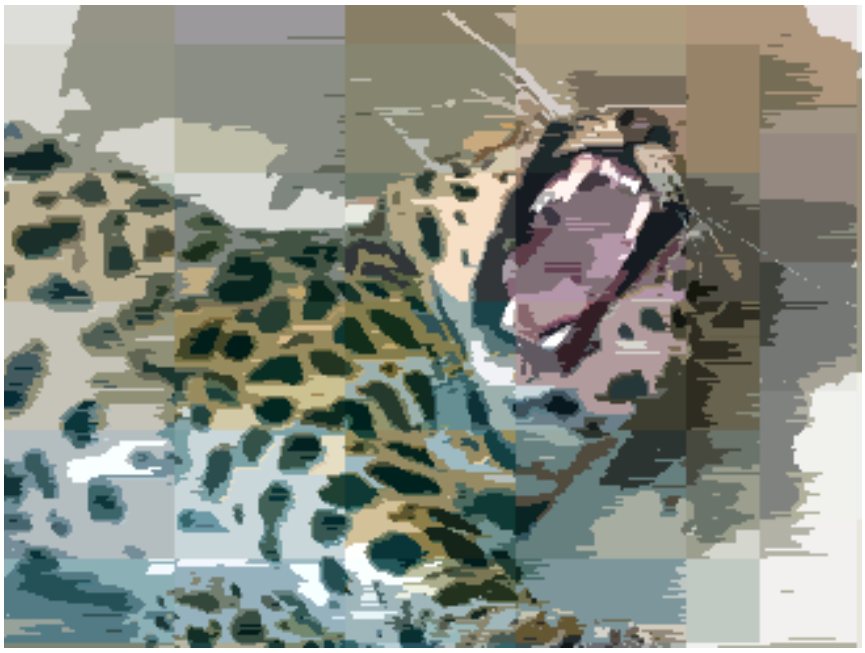
Obrázek A.4 Originální obrázek – Lodě (střední velikost)

Obrázek A.5 Segmentovaný obrázek – Lodě (střední velikost)

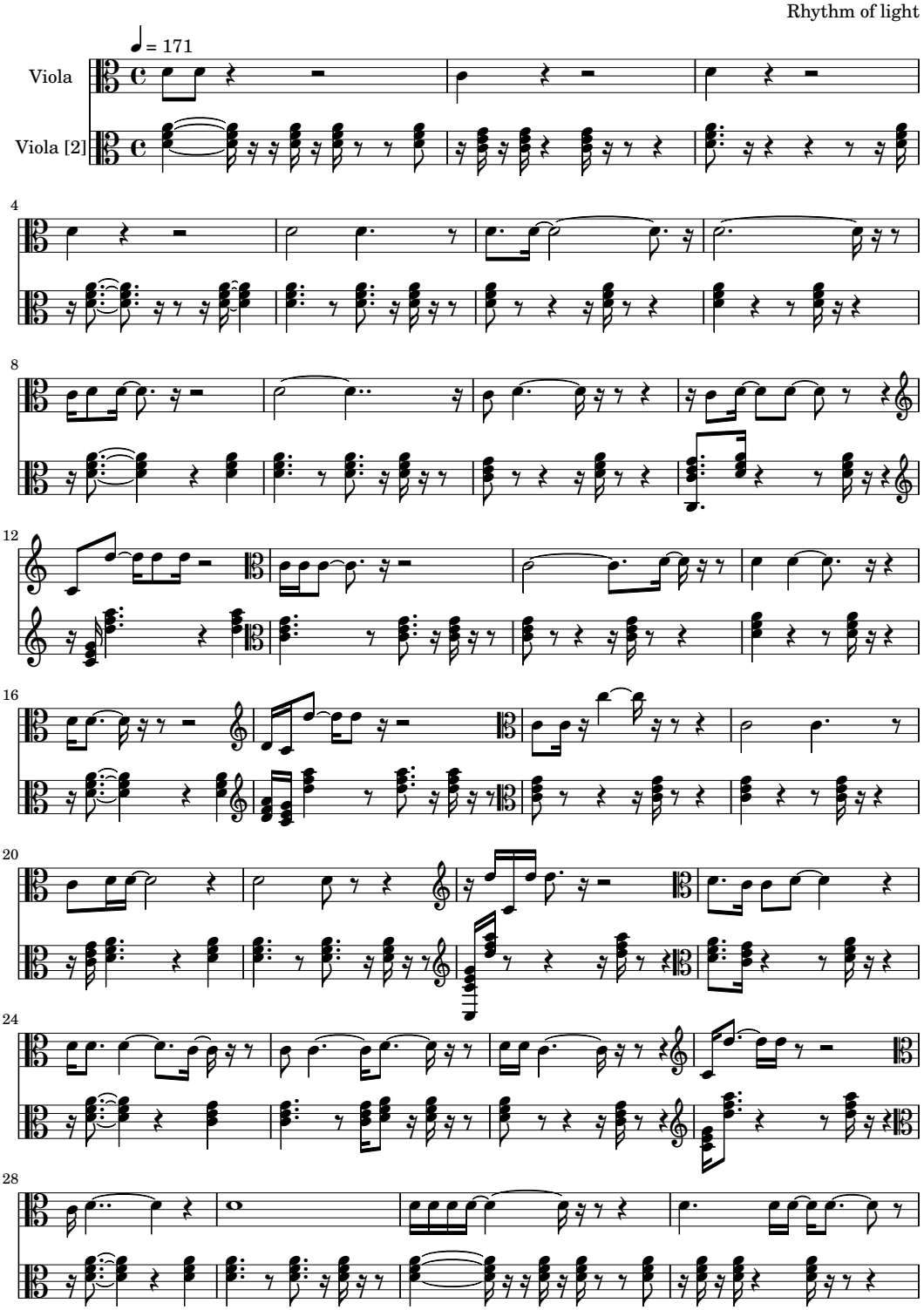


Obrázek A.6 Výsledný notový zápis – Lodě (střední velikost)

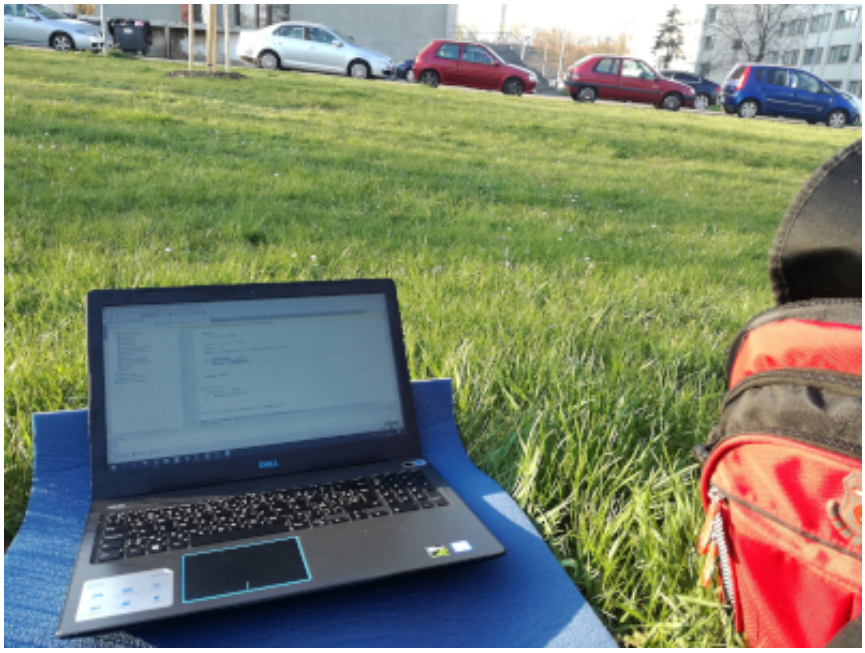
Obrázek A.7 Originální obrázek – Divoké zvíře (střední velikost)



Obrázek A.8 Segmentovaný obrázek – Divoké zvíře (střední velikost)

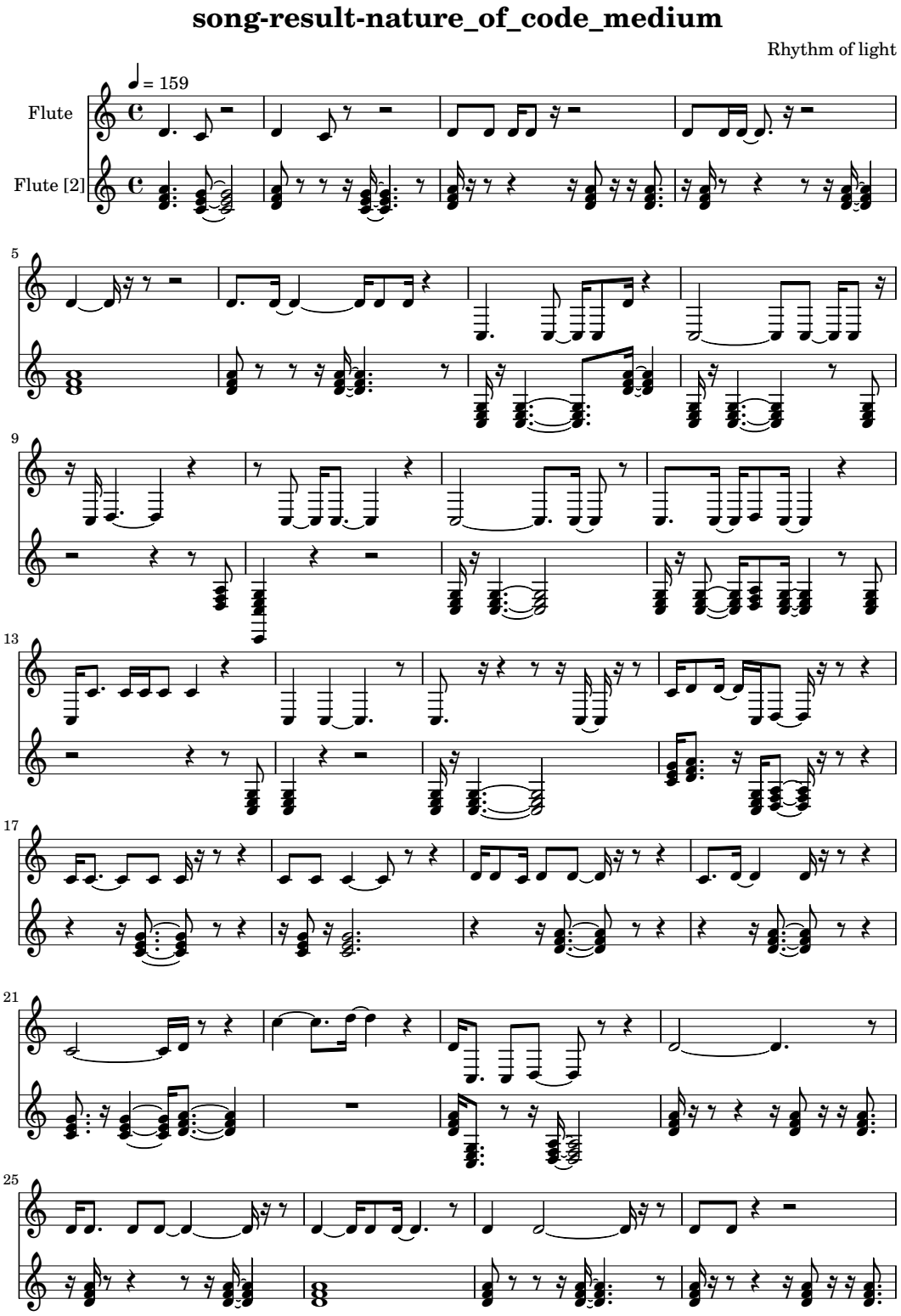


Obrázek A.9 Výsledný notový zápis – Divoké zvíře (střední velikost)



Obrázek A.10 Originální obrázek – Nature of code (střední velikost)

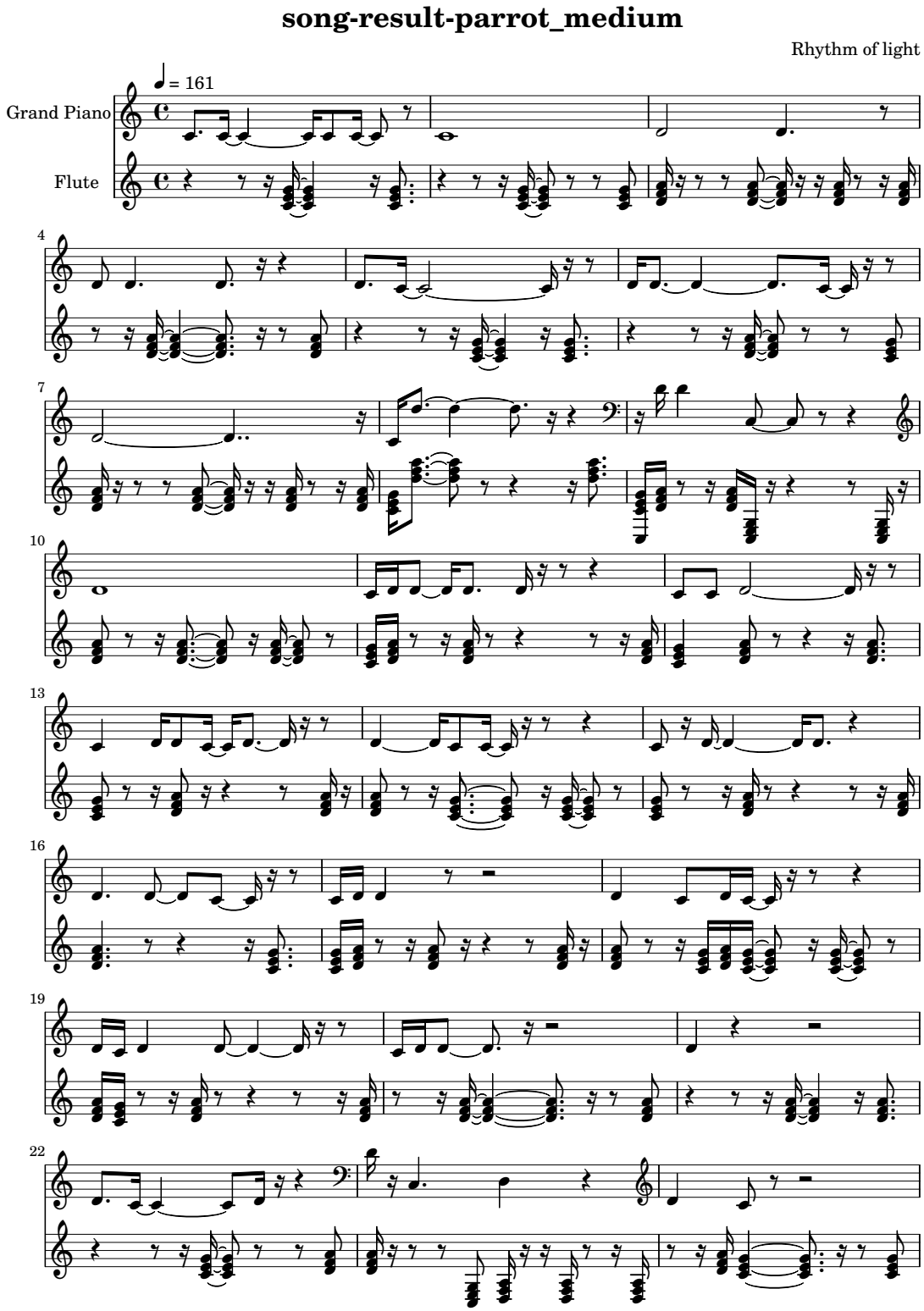
Obrázek A.11 Segmentovaný obrázek – Nature of code (střední velikost)



Obrázek A.12 Výsledný notový zápis – Nature of code (střední velikost)

Obrázek A.13 Originální obrázek – Papoušek (střední velikost)

Obrázek A.14 Segmentovaný obrázek – Papoušek (střední velikost)



Obrázek A.15 Výsledný notový zápis – Papoušek (střední velikost)

Příloha B

Návod ke spuštění a instalaci

B.1 Windows

Instalace pomocných programů

1. Zjistěte, zda je na vašem počítači nainstalovaný program *Lilypond*.
2. Pokud tento program máte již nainstalovaný, pokračujte krokem 8
3. Pokud *Lilypond* na svém počítači nemáte, postupujte dle následujících kroků:
4. Stáhněte si instalační soubor programu *Lilypond* z oficiální webové stránky *Lilipondu*:
   `https://lilypond.org/windows.cs.html`
5. Spusťte instalaci programu *Lilypond* a postupujte dle průvodce instalací.
6. Při instalaci *Lilypondu* nechte průvodce instalací nainstalovat všechny komponenty programu.
7. Po úspěšné instalaci restartujte počítač.
8. Nyní jste připraven na spuštění aplikace *Rhythm of light*.

Spuštění aplikace

1. Ve složce `Rhythm_of_light` se nachází spustitelný soubor `Rhythm_of_light.exe`. Otevřete ho.
2. Po spuštění aplikace se na obrazovce objeví terminálové okno.
3. Vyčkejte, než se objeví grafické uživatelské rozhraní (Otevírání aplikace *Rhythm of light* může chvíli trvat).
4. Aplikace je spuštěná a vy ji můžete začít používat.

B.2 Linux

Instalace pomocných programů

1. Zjistěte, zda je na vašem počítači nainstalovaný program *Lilypond*.
2. Pokud tento program máte již nainstalovaný, pokračujte k bodu 6.
3. Pokud *Lilypond* na svém počítači nemáte, postupujte dle následujících kroků:
4. Stáhněte si instalační soubor programu *Lilypond* z oficiální webové stránky *Lilypondu*:

   `https://lilypond.org/unix.cs.html`

   Nebo si *Lilypond* nainstalujte pomocí svého správce balíčků.
5. Při instalaci *Lilypondu* nechte průvodce instalací nainstalovat všechny komponenty programu.
6. Zjistěte, zda je na vašem počítači nainstalovaný program *Fluidsynth*.
7. Pokud tento program máte již nainstalovaný, pokračujte k bodu 11.
8. Pokud *Fluidsynth* na svém počítači nemáte, postupujte dle následujících kroků:
9. Stáhněte si instalační soubor programu *Fluidsynth* z oficiální webové stránky programu *Fluidsynth*:

   `https://www.fluidsynth.org/download/`

   Nebo si *Fluidsynth* nainstalujte pomocí svého správce balíčků.
10. Při instalaci *Fluidsynthu* nechte průvodce instalací nainstalovat všechny komponenty programu.
11. Pokud jste instaloval nějaké balíčky, restartujte počítač.
12. Nyní jste připraven na spuštění aplikace *Rhythm of light*.

Spuštění aplikace

1. Ve složce `Rhythm_of_light` se nachází spustitelný soubor `Rhythm_of_light`. Otevřete ho.
2. Vyčkejte, než se objeví grafické uživatelské rozhraní (Otevírání aplikace *Rhythm of light* může chvíli trvat).
3. Aplikace je spuštěná a vy ji můžete začít používat.

B.3 Spuštění ze zdrojového kódu

Instalace pomocných programů

1. Zjistěte, zda je na vašem počítači nainstalovaný program *Lilypond*.
2. Pokud tento program máte již nainstalovaný, pokračujte k bodu 6.
3. Pokud *Lilypond* na svém počítači nemáte, postupujte dle následujících kroků:

4. Stáhněte si instalační soubor programu *Lilypond* z oficiální webové stránky *Lilypondu*:

   `https://lilypond.org/download.cs.html`

   Nebo si *Lilypond* nainstalujte pomocí svého správce balíčků.

5. Při instalaci *Lilypondu* nechte průvodce instalací nainstalovat všechny komponenty programu.

6. Zjistěte, zda je na vašem počítači nainstalovaný program *Fluidsynth*.

7. Pokud tento program máte již nainstalovaný, pokračujte k bodu 11.

8. Pokud *Fluidsynth* na svém počítači nemáte, postupujte dle následujících kroků:

9. Stáhněte si instalační soubor programu *Fluidsynth* z oficiální webové stránky programu *Fluidsynth*:

   `https://www.fluidsynth.org/download/`

   Nebo si *Fluidsynth* nainstalujte pomocí svého správce balíčků.

10. Při instalaci *Fluidsynthu* nechte průvodce instalací nainstalovat všechny komponenty programu.

11. Pokud jste instaloval nějaké balíčky, restartujte počítač.

Instalace aplikace

1. Stáhněte si `Python 3.7.6.` z oficiálních stránek *Pythonu*:

   https://www.python.org/downloads/release/python-376/

   a dle pokynů *Python* nainstalujte.

2. Je doporučeno instalovat v aplikaci ve virtuálním prostředí, i když to není nutné.

3. Pokud používáte virtuální prostředí, zaktivujte ho.

4. Ujistěte se, že máte nainstalovaný balíček `pip` a `setuptools` nejnovější verze. Nainstalovat je můžete pomocí příkazu:

   ```
   python3 -m pip install --upgrade pip setuptools
   ```

5. Ve složce `source_code` naleznete zdrojové soubory aplikace.

6. Spusťte příkaz:

   ```
   python3 -m pip install -r requirements-gpu.txt
   ```

   Případně příkaz:

   ```
   python3 -m pip install -r requirements.txt
   ```

   Výše uvedené příkazy se liší tím, jestli bude aplikace používat verzi balíčku `tensorflow` s využitím *grafické karty* nebo ne.

7. Po úspěšné instalaci můžete přejít ke spuštění aplikace.

Spuštění aplikace

1. Pokud používáte virtuální prostředí, zaktivujte ho.
2. Přepněte se do složky `source_code` a následně do `Rhythm_of_light`.
3. Následujícím příkazem spustíte aplikaci:
   ```
   python3 -m cz
   ```

Bibliografie

1. ZENKL, Luděk. *ABC HUDEBNÍ NAUKY*. 3. vyd. Praha: Supraphon, 1982. ISBN 02-002-82.
2. KOFROŇ, Jaroslav. *UČEBNICE HARMONIE*. Third. Praha: Státní hudební vydavatelství, 1963. ISBN 02-003-63.
3. KYTNEROVÁ, Šárka. *Harmonizace lidových písní u lidových souborů* [online]. 2010 [cit. 2022-01-06]. Dostupné z: `https://is.muni.cz/th/h4gzh/`. Diplomová práce. Masarykova univerzita, Pedagogická fakulta, Brno. Vedoucí práce Petr HALA.
4. VIATOUR, Luc. *Fire breathing 2 Luc Viatour.jpg* [online]. 2008-08 [cit. 2022-05-08]. Dostupné z: `https://en.wikipedia.org/wiki/File:Fire_breathing_2_Luc_Viatour.jpg`.
5. VIATOUR, Luc; JACOBOLUS. *Fire-breather CIELAB L*.jpg* [online]. 2010-02 [cit. 2022-05-08]. Dostupné z: `https://en.wikipedia.org/wiki/File:Fire-breather_CIELAB_L*.jpg`.
6. VIATOUR Luc, jacobolus. *Fire-breather HSV V.jpg* [online]. 2010-02 [cit. 2022-05-08]. Dostupné z: `https://en.wikipedia.org/wiki/File:Fire-breather_HSV_V.jpg`.
7. ZMEŠKAL, Oldřich; ČEPPAN, Michal; DZIK, Petr. Barevné prostory a správa barev. *Image Science Fundamentals* [online]. 2002 [cit. 2022-05-08]. Dostupné z: `http://imagesci.fch.vut.cz/download/stud06_rozn02.pdf`.
8. BUNKS, Carey. *Grokking the Gimp* [online]. New Riders Publishing, 2000 [cit. 2022-05-08]. ISBN 0-7357-0924-6. Dostupné z: `http://tinf2.vub.ac.be/~dvermeir/manual/gimp/Grokking-the-GIMP-v1.0/node52.html`.
9. DANNHOFEROVÁ, Jana. *Velká kniha barev: kompletní průvodce pro grafiky, fotografy a designéry*. 1. vyd. Brno: Computer Press, 2012. ISBN 978-80-251-3785-7.
10. LUO, M. R.; CUI, G.; RIGG, B. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application* [online]. 2001, roč. 26, č. 5, s. 340–350 [cit. 2021-11-14]. Dostupné z DOI: `https://doi.org/10.1002/col.1049`.
11. TEIXEIRA, Joana; PINTO, H. Sofia. Cross-Domain Analogy: From Image to Music. In: *Proceedings of the 5th International Workshop on Musical Metacreation* [online]. MUME, 2017, s. 8 [cit. 2021-11-13]. ISBN 978-1-77287-019-0. Dostupné z DOI: `10.5281/zenodo.4285254`.
12. TEIXEIRA, Joana Borges. *Cross-Domain Analogy: From Image to Music* [online]. 2017 [cit. 2022-01-07]. Dostupné z: `https://fenix.tecnico.ulisboa.pt/cursos/meic-a/dissertacao/1691203502342721`. Dipl. pr. Técnico Lisboa, Information Systems a Computer Engineering, Lisboa. Vedoucí práce Helena Sofia Andrade Nunes Pereira PINTO.

13. COMPUTATIONAL CREATIVITY, Int. Conference on. *Musical Metacreation Workshop, ICCC 2017 (June 19) Part Three* [online]. 2017-06-19 [cit. 2022-01-07]. Dostupné z: https://www.youtube.com/watch?v=iNvdqRJVEqE&t=4276s.

14. LUI, Simon. Generate expressive music from picture with a handmade multi-touch music table. In: BERDAHL, Edgar; ALLISON, Jesse (ed.). *Proceedings of the International Conference on New Interfaces for Musical Expression* [online]. Baton Rouge, Louisiana, USA: Louisiana State University, 2015, s. 374–377 [cit. 2021-01-03]. ISSN 2220-4806. Dostupné z DOI: 10.13140/RG.2.1.3330.0323.

15. CYTOWIC, Richard E. *Synesthesia: A Union of the Senses*. A Bradford book, 2002. Bradford Books. ISBN 9780262032964.

16. LOUI, Psyche; ZAMM, Anna; SCHLAUG, Gottfried. *Absolute Pitch and Synesthesia: Two Sides of the Same Coin? Shared and Distinct Neural Substrates of Music Listening* [online]. 2012-01-01 [cit. 2022-05-08]. Dostupné z: https://europepmc.org/articles/PMC3596158.

17. CAIVANO, José Luis. *Color and Sound: Physical and Psychophysical Relations*. 1994. Č. 2. Dostupné z DOI: 10.1111/j.1520-6378.1994.tb00072.x.

18. PALMER, Stephen E.; SCHLOSS, Karen B.; XU, Zoe; PRADO-LEÓN, Lilia R. *Music-color associations are mediated by emotion* [online]. 2013-05-13 [cit. 2022-01-07]. Dostupné z: https://doi.org/10.1073/pnas.1212562110.

19. KALTENBRUNNER, Martin; BOVERMANN, Till; BENCINA, Ross; COSTANZA, Enrico. *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), Vannes, France*. TUIO: A Protocol for Table-Top Tangible User Interfaces. 2005-05. Dostupné také z: https://www.researchgate.net/publication/225075863_TUIO_A_Protocol_for_Table-Top_Tangible_User_Interfaces.

20. MERCER-TAYLOR, Andrew; ALTOSAAR, Jaan. Sonification of Fish Movement Using Pitch Mesh Pairs. In: BERDAHL, Edgar; ALLISON, Jesse (ed.). *Proceedings of the International Conference on New Interfaces for Musical Expression*. Baton Rouge, Louisiana, USA: Louisiana State University, 2015, s. 28–29. ISSN 2220-4806. Dostupné z DOI: 10.5281/zenodo.1179138.

21. EULER, Leonhard. *Tentamen novae theoriae musicae ex certissimis harmoniae principiis dilucide expositae*. Ex typographia Academiae scientiarum, 1739.

22. ANDREWJMT. *fishmusic* [online]. GitHub, 2015-01-22 [cit. 2022-01-04]. Dostupné z: https://github.com/andrewjmt/fishmusic.

23. MERCER-TAYLOR, Andrew. *Sonification of Fish Movement Using Pitch Mesh Pairs (NIME 2015)* [online]. 2015-04-18 [cit. 2022-01-04]. Dostupné z: https://www.youtube.com/watch?v=HzsFGQyIpuc.

24. LTD, Gardos Software. *Melobytes.com – Music inspiration apps* [online]. 2022 [cit. 2022-01-05]. Dostupné z: https://melobytes.com/.

25. JACK, Olivia. *Pixelsynth* [online]. 2021 [cit. 2021-12-19]. Dostupné z: https://ojack.xyz/PIXELSYNTH/.

26. JACK, Olivia. *Pixelsynth - olivia jack* [online]. 2021 [cit. 2021-12-19]. Dostupné z: https://ojack.xyz/articles/pixelsynth/index.html.

27. JACK, Olivia. *PIXELSYNTH* [online]. GitHub, 2021-09-30 [cit. 2021-12-30]. Dostupné z: https://github.com/ojack/PIXELSYNTH.

28. PILARCZYK, Joanna; KUNIECKI, Michał; WOŁOSZYN, Kinga; STERNA, Radosław. Blue blood, red blood. How does the color of an emotional scene affect visual attention and pupil size? *Vision Research*. 2020, roč. 171, s. 36–45. ISSN 0042-6989. Dostupné z DOI: https://doi.org/10.1016/j.visres.2020.04.008.
29. *Emotional Response to a Picture by the Change of Color: A comparison study between adults and children* [online]. Zenodo, 2019 [cit. 2022-01-04]. Dostupné z DOI: 10.5281/zenodo.2596035.
30. JOHN, Radek. *To, jak jsou dlouhé, poznají ryby čichem* [online]. 2013-02-11 [cit. 2022-04-23]. Dostupné z: https://www.tyden.cz/rubriky/veda/priroda/to-jak-jsou-dlouhe-poznaji-ryby-cichem_261034.html.
31. SCHIMMEL, Jiří. Komunikační rozhraní MIDI. *Elektrorevue* [online]. 2002 [cit. 2022-02-17]. ISSN 1213-1539. Dostupné z: http://www.elektrorevue.cz/clanky/02069/index.html.
32. GEGENFURTNER, Karl R.; BLOJ, Marina; TOSCANI, Matteo. *The many colours of 'the dress'* [online]. Elsevier BV, 2015-05-14 [cit. 2022-01-12]. Č. 13. Dostupné z DOI: 10.1016/j.cub.2015.04.043.
33. THOMSON, G.; MACPHERSON, F. *Adelson's Checker-Shadow Illusion* [online]. 2017-07 [cit. 2022-05-09]. Dostupné z: https://www.illusionsindex.org/ir/checkershadow.
34. SOBOTKOVÁ, Jana. *Funkční harmonie v praktických příkladech* [online]. Brno, 2007 [cit. 2022-04-05]. Dostupné z: https://is.muni.cz/th/wpzem/. Diplomová práce. Masarykova univerzita, Pedagogická fakulta. Vedoucí práce Petr HALA.
35. MARKS, Lawrence E. On Associations of Light and Sound: The Mediation of Brightness, Pitch, and Loudness. *The American Journal of Psychology* [online]. 1974, roč. 87, č. 1/2, s. 173–188 [cit. 2022-04-04]. ISSN 0002-9556. Dostupné z DOI: 10.2307/1422011.
36. STEEVES, Janice Mason. *The Importance of Silence in Art* [online]. 2014-08 [cit. 2022-04-25]. Dostupné z: https://janicemasonsteevesartwork.blogspot.com/2014/08/the-importance-of-silence-in-art.html.
37. CAIVANO, José Luis. *Black, white, and grays: Are they colors, absence of color or the sum of all colors?* [Online]. 2022-02-10 [cit. 2022-04-07]. Č. 2. Dostupné z DOI: https://doi.org/10.1002/col.22727.
38. ČALOUD, Karel. *Harmonické funkce* [online]. 2022 [cit. 2022-02-06]. Dostupné z: https://webhouse.cz/kurz-harmonie/harmonicke_funkce.htm.
39. HURST, Ashley. *Movement – A Principle of Art* [online]. 2022 [cit. 2022-04-04]. Dostupné z: https://thevirtualinstructor.com/blog/movement-a-principle-of-art.
40. WELLS, Alan. Music and Visual Color: A Proposed Correlation. *Leonardo* [online]. 1980, roč. 13, č. 2, s. 101–107 [cit. 2022-05-09]. ISSN 0961-1215. Dostupné z: http://www.jstor.org/stable/1577978.
41. THEORY, Simplifying. *What is Harmonic Function* [online]. 2022 [cit. 2022-05-09]. Dostupné z: https://www.simplifyingtheory.com/harmonic-function/.
42. CHANDLER, Daniel; MUNDAY, Rod. *A Dictionary of Media and Communication* [online]. Oxford University Press, 2011 [cit. 2022-03-10]. Dostupné z DOI: 10.1093/acref/9780199568758.001.0001.
43. SHAKI, Samuel; FISCHER, Martin H.; GÖBEL, Silke M. *Direction counts: A comparative study of spatially directional counting biases in cultures with different reading directions* [online]. 2012-06 [cit. 2022-03-10]. Č. 2. ISSN 0022-0965. Dostupné z DOI: https://doi.org/10.1016/j.jecp.2011.12.005.

44. HOUSER, Pavel. *Proč píšeme zleva doprava?* [Online]. 2007-08-02 [cit. 2022-03-16]. Dostupné z: https://www.scienceworld.cz/clovek/proc-piseme-zleva-doprava-918/.
45. SCHLOSS, Karen B.; LAWLER, Patrick; PALMER, Stephen E. *The color of music* [online]. 2008-05 [cit. 2022-04-07]. Č. 6. ISSN 1534-7362. Dostupné z DOI: 10.1167/8.6.580.
46. TSANG, Tawny; SCHLOSS, Karen B. *Associations between color and music are mediated by emotion and influenced by tempo* [online]. 2010 [cit. 2022-04-07]. Dostupné z: https://cpb-us-w2.wpmucdn.com/campuspress.yale.edu/dist/a/1215/files/2015/11/2010_Tsang-and-Schloss-The-Color-of-Music-1w1j7sr.pdf.
47. BLANCHARD, Julian. The Brightness Sensibility of the Retina. *Phys. Rev.* [Online]. 1918, roč. 11, s. 81–99 [cit. 2022-05-09]. Dostupné z DOI: 10.1103/PhysRev.11.81.
48. ZENKL, Luděk. *ABC hudebních forem*. First. Praha: Supraphon, 1984. ISBN 02-012-84.
49. MORAN, Kate. *Heatmap Visualizations from Signifier Eyetracking Experiment* [online]. 2017-09-03 [cit. 2022-04-25]. Dostupné z: https://www.nngroup.com/articles/heatmap-visualizations-signifiers/.
50. KHAN, Shanaz. *Eye tracking Heatmap: Simplify visitor behavior analysis* [online]. 2022-02-04 [cit. 2022-04-25]. Dostupné z: https://vwo.com/blog/eye-tracking-heatmap/.
51. HARKER, Stephen. *The Golden Ratio/Fibonacci Sequence: What It Means to Photographers* [online]. 2020-02-26 [cit. 2022-05-09]. Dostupné z: https://phlearn.com/magazine/golden-ratio-fibonacci-sequence-photographers/.
52. CARABAJAL, Sergio. *Abstract Rainbow Pattern - Free photo on Pixabay* [online]. Pixabay, 2021-06-01 [cit. 2022-05-08]. Dostupné z: https://pixabay.com/photos/abstract-rainbow-pattern-texture-6297317/.
53. DISLEY, Alastair; HOWARD, David; HUNT, Andy. Timbral description of musical instruments. *Research Gate* [online]. 2006 [cit. 2022-05-09]. Dostupné z: https://www.researchgate.net/publication/242557004_Timbral_description_of_musical_instruments.
54. OLAFENWA, Moses; OLAFENWA, John. *ImageAI, an open source python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities* [online]. 2021-05-08 [cit. 2022-04-25]. Dostupné z: https://github.com/OlafenwaMoses/ImageAI.
55. PYSIMPLEGUI. *PySimpleGUI* [online]. 2022-05-03 [cit. 2022-05-03]. Dostupné z: https://github.com/PySimpleGUI/PySimpleGUI.
56. EVANSTEIN, Marc P. *SCAMP (Suite for Computer-Assisted Music in Python)* [online]. GitHub, 2021-12-09 [cit. 2022-05-03]. Dostupné z: https://github.com/MarcTheSpark/scamp.
57. BRADSKI, G. *The OpenCV Library* [online]. 2000 [cit. 2022-05-03].
58. OLAFENWA, Moses; OLAFENWA, John. *ImageAI, an open source python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities* [online]. 2021-01-04 [cit. 2022-04-25]. Dostupné z: https://github.com/OlafenwaMoses/ImageAI/releases/download/essentials-v5/resnet50_coco_best_v2.1.0.h5.
59. CONTRIBUTORS, Wikipedia. *Hue - Wikipedia, The Free Encyclopedia* [online]. 2022-05 [cit. 2022-05-09]. Dostupné z: https://en.wikipedia.org/wiki/Hue#24_hues_of_HSL/HSV.
60. CUTHBERT, Michael Scott; CUTHBERTLAB. *music21* [online]. GitHub, 2022 [cit. 2022-04-05]. Dostupné z: https://github.com/cuthbertLab/music21.

61. HANAPPE, Peter et. al. *FluidSynth* [online]. GitHub, 2022-04-27 [cit. 2022-05-04]. Dostupné z: `https://github.com/FluidSynth/fluidsynth`.

62. S. COLLINS, Christian et al. *MuseScore_General.sf3* [online]. 2022-03 [cit. 2022-05-09]. Dostupné z: `https://musescore.org/en/handbook/3/soundfonts-and-sfz-files`.

63. S. COLLINS, Christian et al. *MuseScore_General.sf2* [online]. 2022-03 [cit. 2022-05-09]. Dostupné z: `https://musescore.org/en/handbook/3/soundfonts-and-sfz-files`.

64. HARRIS, Charles R.; MILLMAN, K. Jarrod; WALT, Stéfan J. van der; GOMMERS, Ralf; VIRTANEN, Pauli; COURNAPEAU, David; WIESER, Eric; TAYLOR, Julian; BERG, Sebastian; SMITH, Nathaniel J.; KERN, Robert; PICUS, Matti; HOYER, Stephan; KERKWIJK, Marten H. van; BRETT, Matthew; HALDANE, Allan; RÍO, Jaime Fernández del; WIEBE, Mark; PETERSON, Pearu; GÉRARD-MARCHANT, Pierre; SHEPPARD, Kevin; REDDY, Tyler; WECKESSER, Warren; ABBASI, Hameer; GOHLKE, Christoph; OLIPHANT, Travis E. Array programming with NumPy. *Nature*. 2020, roč. 585, č. 7825, s. 357–362. Dostupné z DOI: `10.1038/s41586-020-2649-2`.

Obsah přiloženého média

Adresářová struktura média

- `source_code` ..... adresář se zdrovými kódy aplikace
- `bin` ..... adresář se spustitelnými soubory aplikace
  - `Windows_10` ..... adresář se spustitelným souborem aplikace pro Windows 10
  - `Manjaro_Linux` ..... adresář se spustitelným souborem aplikace pro Manjaro Linux
- `text` ..... složka s textem práce
  - `thesis_kolemrad_colors.pdf` ..... PDF práce barevně
  - `thesis_kolemrad_gray.pdf` ..... PDF práce černobíle
- `readme.txt` ..... stručný popis obsahu média
- `demos` ..... ukázky výsledků práce