FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Fraviz: Fraktální audio vizualizér
Student:	Radka Hošková
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2021/22

Pokyny pro vypracování

Cílem práce je navrhnout způsob získání dat z audio nahrávky hudby rozšiřující hudební zážitek a k získaným datům generovat vizuální odezvu s využitím fraktálů.

1) Proveďte rešerši
 a) v oblasti MIR, způsoby získání dat z hudby přínosných pro člověka;
 b) použití fraktálů pro vizualizování získaných dat z hudební audio nahrávky.
2) Analyzujte současná softwarová řešení pro vizualizaci hudby využívající netriviální způsob získávání dat, nebo vizualizace.
3) Navrhněte způsob, jakým získaná data z hudební audio nahrávky pomocí fraktálů vizualizovat.
4) Implementujte prototyp realizující toto řešení esteticky a netriviálně.
5) Otestujte na vhodných datech a debatujte výhody a rozdíly jednotlivých vizualizací.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 18. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Fraktální audio vizualizér

Radka Hošková

Katedra softwarového inženýrství
Vedoucí práce: Ing. Radek Richtr, Ph.D.

30. července 2020

Poděkování

Mé díky patří Ing. Radku Richtrovi, PhD. za vedení a pomoc při psaní bakalářské práce. Díky patří i všem, kteří se mnou práci diskutovali, nebo už jen byli inspirací.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. § 46 odst. 6 tohoto zákona 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 30. července 2020 ……

Odkaz na tuto práci

Hošková, Radka. *Fraktální audio vizualizér*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tématem práce je vizualizace hudby s pomocí fraktální geometrie. Získávání dat z hudby se věnuje rozvíjející se a mezioborová vědecká oblast *Music Information Retrieval*. Základy zpracování signálu, vlastnosti hudebních signálů a Fourierova transformace jsou úvodem pro rešerši tohoto oboru v rámci této práce. Práce představuje stávající nástroje pro analýzu hudby a uvádí příklady hudebních vizualizací, kde hudební produkce a vizualizace jsou úzce spjaty. Následně analyzuje tyto nástroje a také vybrané způsoby generování fraktálů. Poté navrhuje způsoby, jakými fraktály animovat a tak použít pro hudební vizualizaci. Z několika vytvořených verzí je pak klíčovou finální verze používající aplikační rozhraní společnosti *Spotify* pro získávání dat a různé způsoby generování fraktálů včetně L-systémů.

Klíčová slova vizualizace hudby, získávání dat z hudby (Music information retrieval), fraktály, IFS, L-systémy, Fourierova transformace, FFT, Spotify API, Threejs

Abstract

Music visualization by fractal geometry is the topic of this work. The first phase of the music visualization process is obtaining data for visualization. Evolving and interdisciplinary scientific field Music Information Retrieval (MIR) is specializing on this type of tasks. The basics of signal processing, properties of music signals and Fourier transform (presented in this work) were the introduction to make a brief research of this field for this work. Existing tools for music analysis and examples of music visualizations were introduced. Approaches to animate fractals in order to use them for music visualizations were proposed while presenting selected approaches to generate fractal objects. Several versions of music visualizations were implemented and discussed, including the final version which is using the Spotify application user interface and various methods of generating fractals, including L-systems.

Keywords music vizualization, Music information retrieval (MIR), fractals, IFS, L-Systems, Fourier transform, FFT, Spotify API, Threejs

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Hudba má nespočetné množství podob a použití. Kromě zvuku samotného můžeme hodnotit ale i subjektivní zážitek při poslechu. Jedním ze způsobů, jak vyjádřit zalíbení v námi vnímané hudbě, může být odevzdání se tanci a reagování na změny v hudbě každým pohybem těla prostorem. Ať už pohupováním do rytmu, nebo snahou zachycení melodie naší představivostí se můžeme na okamžik cítit součástí hudební produkce.

Tanec, představivost a ani samotný sluch ovšem není v moci každého člověka. Schopnost vnímat, porozumět a užít si hudbu je dána limity našeho těla a naší mysli. Kromě různých sluchových a pohybových obtíží můžeme být omezeni i vizuální představivostí (tzv. aphantasia).

Existuje množství způsobů, jak doprovodit poslech o další vjemy a tím tak učinit hudbu záživnější a přístupnější. S nárůstem elektronické hudby, kde mnohdy roli nástrojů zastane mixážní pult, lidé stále více usilují o doplnění zvuku digitálními vizuálními efekty. Právě analyzováním hudebních nahrávek a volbou algoritmů pro generování vizuálů se věnuje tato práce.

Cíle práce

Práce se věnuje způsobům získávání hudebních dat a možnostem využití fraktální geometrie k vizualizacím. Kromě výsledného prototypu si práce klade za cíl provedení analýzy problematiky, současných řešení a technologiích možných pro realizaci. Analýza zvuku má spojit hudební teorii a analýzu hudby jako zvukového signálu a závěrem určit jaká data je validní pro uživatele aplikace vizualizovat. Další analýza je průzkumem možností generování fraktálů a možnosti animování takových útvarů. Rešerše by se měli spojit v prototyp implementující závěry analýz.

Struktura

Bakalářská práce je rozdělena na teoretickou a realizační část. Předmětem teoretické části je získávání dat z hudby, možnosti využití fraktálů ve vizualizacích a analýza současných řešení zobrazování hudby.

Část realizační navazuje na teoretickou část rozborem možností implementace a popisuje postupný vývoj implementace zvoleného postupu. Následuje testování a ukázka vizualizací výsledného prototypu.

Část I

Teoretická část

KAPITOLA 1

Hudba

Hudba je organizovaný systém zvuků a lze na ní pohlížet z hlediska technického i uměleckého. Fyzikálně je zvuk mechanické vlnění v látkovém prostředí, které je schopno vyvolat sluchový vjem. Organizovaný sled zvuků, který tvoří hudbu je ale zkoumána spíše subjektivně a umělecky. Tato kapitola se věnuje možnostem získání pro člověka relevantních informací z digitálního záznamu.

1.1 Digitalizace zvuku

Analogový signál má svůj půvab. Přidává do nahrávky zkreslení daného nahrávacího zařízení, média i reprodukčního zařízení. Přenášena a uchovávána tak není jenom samotná zvuková nahrávka, ale i informace o použité technice a tedy potažmo i době, ve které byla nahrána. Pro lepší ukládání, přehrávání a případnou pozdější manipulaci se zvukem, se však analogový signál převádí na signál digitální. Digitalizace není bezztrátová a při nevhodně zvolených parametrech mohou vzniknout i nechtěné aliasy.

Vzorkování

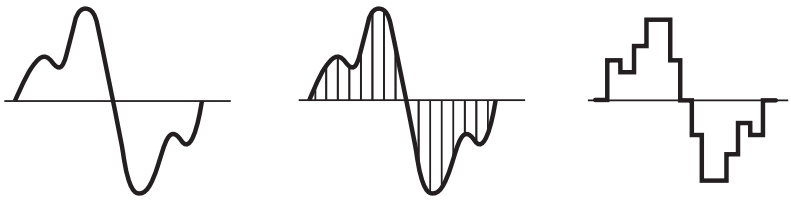
Prvním krokem v digitalizaci zvuku je vzorkování, nebo-li diskretizace signálu v čase. Nižší vzorkovací frekvence způsobí, že nebude slyšet celé slyšitelné spektrum původní nahrávky, nebo budou vyšší frekvence již zkreslené. Jaká by měla být správně vzorkovací frekvence udává **Nyquistův–Shannonův vzorkovací teorém** [1]:

$$f_v > 2f_{\max}\left[s^{-1}\right] \ . \tag{1.1}$$

V doslovném znění: „Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.“ V případě použití nižší vzorkovací frekvence může dojít k tzv. aliasingu, kdy rekonstruovaný signál je výrazně odlišný od původního vzorkovaného signálu.

Kvantování

Kvantování je diskretizace signálu v oboru hodnot, čili v amplitudě. Pokud kvantování není dostatečně jemné, vznikají falešné kvantizační hrany a tedy kvantizační šum.



Obrázek 1.1: Schématické znázornění digitalizace zvuku s časem na ose x. Zleva původní analogový signál s elektrickým napětím na ose y, uprostřed znázornění vzorkování signálu a vpravo kvantování, které bude následně reprezentováno bitově.

1.2 Způsob zaznamenání zvuku

Nynější digitální forma záznamu na audio soubor musela ujít dlouhou cestu. Namátkou například přes kamenné desky, notové zápisy, děrné pásky pianoly, gramofonové desky, magnetické pásky atd. Některé způsoby jako třeba notové zápisy stále mají svůj nenahraditelný význam. Záleží tudíž na výhodách, jaké konkrétní interpretace přinese a jak jich využít[1].

[1] Zvuk může být obsažen i v kvalitně pořízeném videu bez zvuku. Podle [2] lze zvuk z videa zpětnou rekonstrukcí získat, ovšem výsledná kvalita zvuku se samozřejmě nedá uvažovat jako optimální k účelům vizualizace hudby.

Audio soubor reprezentace pro digitální použití nejtypičtější. Dle [3] má soubor daný formát a skládá se většinou ze tří částí: hlavičky, metadat a samotných zvukových dat. Hlavička identifikuje formát souboru a parametry zvukových dat (vzorkovací frekvence, bitová hloubka, počet kanálů). *Metadata* obsahují informace o autorovi, v případě hudební stopy ještě např. název skladby, alba apod. Zvuková data reprezentují samotný zvukový záznam. Způsob ukládání zvukových dat je závislý na počtu kanálů zvuku. Vícekanálové audio, ať už stereo (dva kanály), nebo prostorové (5 kanálů), má většinou v souboru pro každý kanál vyhrazené příslušné místo.

Musical Instrument Digital Interface (MIDI) je rozhraní (standard) používané pro komunikaci elektronických hudebních nástrojů. MIDI je digitálním popisem zvukového záznamu, tzn. souborem informací o výšce jednotlivých tónů, jejich intenzitě, délce, nejrůznějších efektech a jiných informací charakterizujících výsledný sluchový efekt zvukového záznamu. Tyto informace pak zpracovává výstupní zvukové zařízení, které je schopné vytvářet nejrůznější zvuky (zvuková karta či syntezátor).

Tento digitální popis zvukového záznamu je vhodný pro vizualizace. Otázkou jen zůstává, jak se takový záznam získá. Některá zařízení, jako jsou například MIDI klávesy, poskytují snadný způsob získávání MIDI not přímo z hrané hudby. V případě hudby získané bez speciálních zařízení (tradičních hudebních nástrojů, zpěvu, ...) může být pro MIDI reprezentace provedena konverze pomocí specializovaných programů[2]. Například software *Dubler Studio Kit [4]* převádí v reálném čase zpěv na MIDI reprezentaci.

Stem je audio formát od německé firmy *Native Instruments*. Jde o vícekanálový zvukový formát, který usnadňuje práci se zvukem. Obsahuje kromě standardní stereo stopy ještě další čtyři vrstvy audiosignálu. Jednotlivé vrstvy v praxi zastupují čtveřici zvukových „stop“, které dohromady tvoří kompletní skladbu – typicky například bicí-basa-synth-vokál a pomocí kompatibilních ovladačů je lze samostatně ovládat.

[2]Základní rozpoznávání tónů poskytuje i ladička na kytaru.

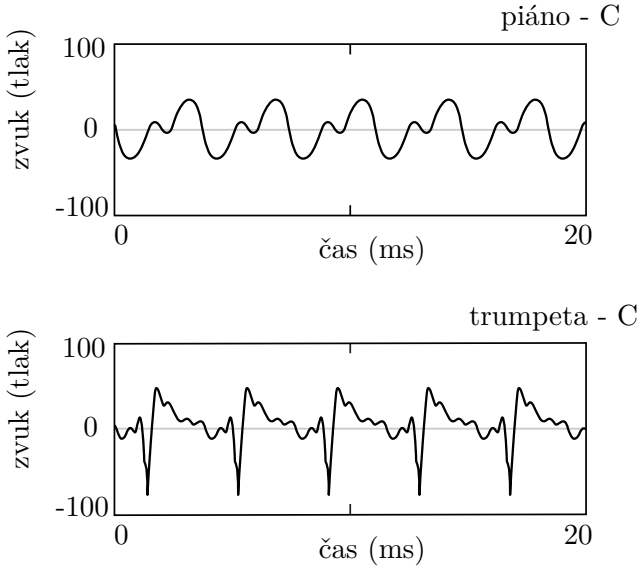
1.3 Vlastnosti hudebních signálů

Existují subjektivní atributy, které jsou při charakterizaci hudebních signálů obzvláště užitečné. Jsou jimi: výška, síla (hlasitost) a témbr (barva)[3].

Výška je charakteristika, která umožňuje seřazení zvuků na frekvenční stupnici zespodu nahoru. Přesněji je výška definovaná jako frekvence sinusové křivky, která je přiřazena zdrojovému zvuku lidským posluchačem.

Hlasitost se v počítačovém zpracování hudby se často udává v decibelech, logaritmické fyzikální jednotce, jelikož lidské tělo vnímá podněty logaritmicky jejich intenzitě.

Barva je vlastnost, díky které můžeme od sebe rozeznat např. zvuky houslí a flétny, které mohou být identické co do výšky, hlasitosti a délky. Koncept této vlastnosti se nedá jednoduše vyjádřit nějakou fyzikální vlastností zvuku, ale spíše souvisí s rozložením spektrální energie a její distribucí v čase jak je ukázáno na obrázku 1.2.



Obrázek 1.2: Náčrt průběhu signálu noty C hrané na různé nástroje.

Zatímco výška, hlasitost a délka zvuku mohou být jednoduše zaznamenány skalární hodnotou, barva je vícerozměrný koncept a je typicky reprezentována vektorem charakteristik. S barvou zvuku velice úzce souvisí tzv. vyšší harmonické frekvence (alikvotní tóny), které nástroje generují.

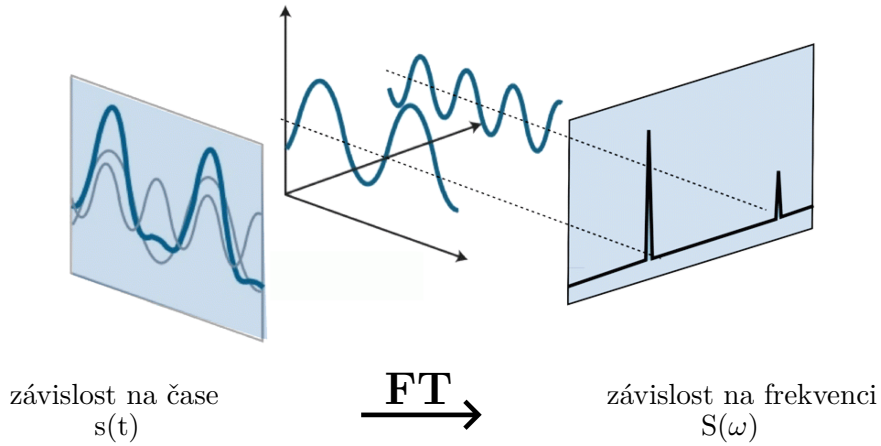
[3]Diplomová práce [5] popisuje vlastnosti hudebních signálů podrobněji.

1.4 Fourierova transformace

Fourierova transformace je ve vizualizaci hudby hojně využívaná. Za jejím vznikem stojí původně kontroverzní myšlenka Jeana Baptisty Josefa Fouriera (1768-1830), že spojitý periodický signál může být reprezentován součtem správně zvolených sinusových vln. Dnes víme, že tomu tak opravdu je (zvolených sinusových vln může být nekonečně mnoho).

Běžné signály jsou tvořeny směsí signálů o různých frekvencích. Postupy, které slouží k výpočtu frekvenčního spektra, jsou označovány jako *Fourierova analýza*. Toto souhrnné označení se rozděluje do kategorií podle druhu signálu. Singál může být diskrétní, či spojitý a periodický, či neperiodický.

Pro analýzu hudební nahrávky je vhodné použít *diskrétní Fourierovu transformaci*[4] (DFT) pro periodicky opakující se signály definované na diskrétních bodech v čase. V praxi se používá *rychlá Fourierova transformace*[5] (FFT), jakožto efektivní algoritmus pro výpočet diskrétní Fourierovi transformace.



Obrázek 1.3: Znázornění Fourierovy transformace.

Pomocí Fourierovi transformace lze převést signál z reprezentace závislé na čase na reprezentaci závislé na frekvenci (což je schématicky znázorněno na obrázku 1.3[6]). Na zpětnou transformaci lze použít inverzní Fourierovu transformaci. Fourierova transformace $S(\omega)$ funkce $s(t)$ je definována integrálním vztahem 1.2, a to jako:

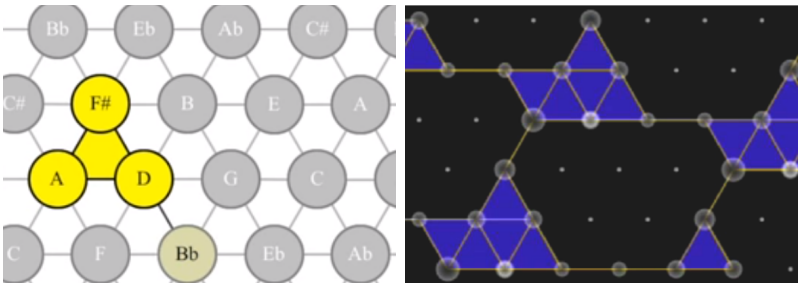
$$S(\omega) = \int_{-\infty}^{\infty} s(t)e^{-i\omega t}dt \ . \tag{1.2}$$

[4]Discrete Fourier Transform
[5]Fast Fourier Transform
[6]Zdroj: `https://tech.liuchao.me/2018/05/polynomial-multiplication/`

1.5 Music Information Retrieval

Music information retrieval (MIR) je zatím malý, ale rostoucí výzkumný obor s mnoho využitími. Je možné zde využít znalosti z muzikologie, psychoakustiky, psychologie, akademické hudební teorie, zpracování signálu, informatiky, strojového učení, optického rozpoznávání hudby, computational intelligence a kombinace těchto oborů. Mezi typické problémy řešené v oboru MIR patří:

- systémy doporučování hudby,
- oddělování stop ze skladby a rozpoznávání nástrojů,
- automatický přepis hudby,
- automatická kategorizace,
- generování hudby.



(a) Tonnetz diagram (b) Isochords

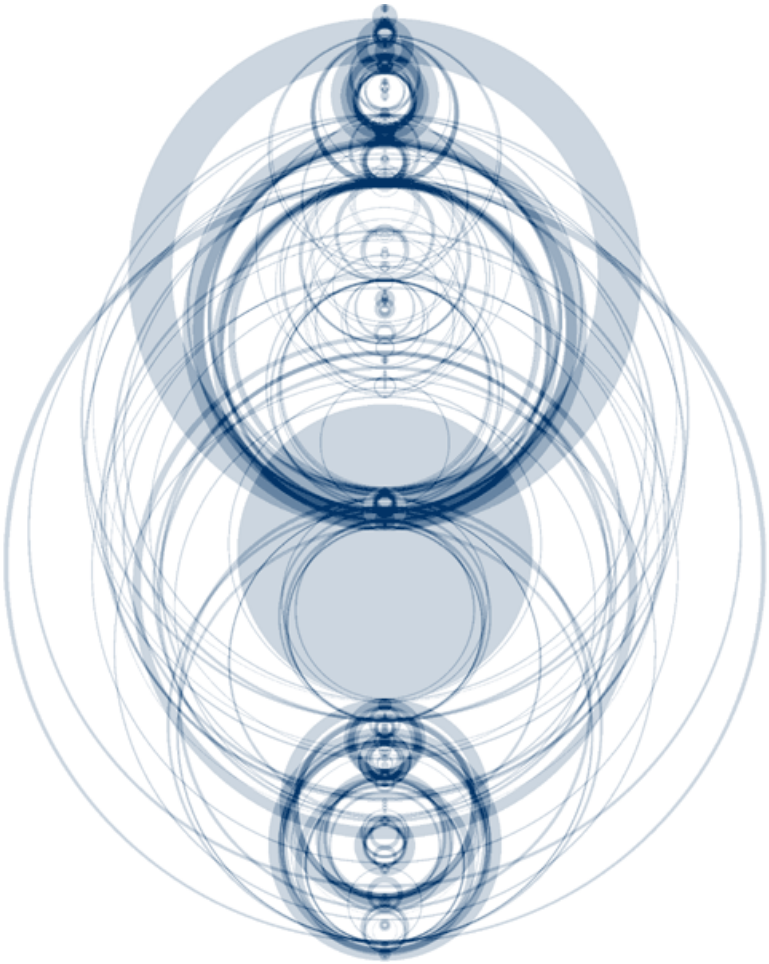
Obrázek 1.4: Isochords a Tonnetz diagram.

Isochords - vizualizace struktury v hudbě

Je mnoho prací, týkajících se tématiky MIR. Jednou z nich je práce [6] z roku 2007 představující *Isochords*. Jde o vizualizaci, která pomáhá při klasifikaci hudebních struktur. Předává informace o kvalitě intervalu, kvalitě akordu a průběhu akordu při přehrávání digitální hudby. Nabízí posluchači způsob uchopení základní struktury hudby bez nutnosti rozsáhlého tréninku. Pro vizualizaci používá Tonnetz[7] - konceptuální mřížkový diagram. Na obrázku 1.4a[8] je ukázka Tonnetz diagramu a na 1.4b Isochords představené v práci [6].

[7] Tonnetz (německy zvuková síť) reprezentuje tonální prostor, který poprvé popsal Leonhard Euler roku 1739.

[8] Zdroj: `youtu.be/NQ7LkWCzKxI`



Obrázek 1.5: *Arc diagram* vizualizující Vivaldiho Podzim ze Čtvera ročních období

Vizualizace sémantických struktur v klasické hudbě

Wingova práce [7] z roku 2009 je jedno z mnoha prací navrhující zajímavý způsob, jak vizualizovat klasickou hudbu. Obsahuje objemnou rešerši, ve které kromě *Isochords* a mnohých dalších projektů a možností vizualizace hudby autor zmiňuje tzv. *arc diagram*[9].

Arc diagram lze použít na zobrazení opakujících se struktur spojením stejných podřetězců polokruhem. Na obrázku 1.5[10] je znázorněna klasická skladba Podzim od Antonia Vivaldiho (s použitím kruhů místo polokruhů).

[9] Obloukový diagram

[10] Zdroj: `http://www.bewitched.com/song.html`

Současný postup získávání informací z hudebních dat

Lopez-Rinconova práce [8] z roku 2019 rovněž diskutuje existující možnosti vizualizace hudby (jakými jsou například *ImproViz* a *MIDIVis*) a navrhuje vlastní způsob, na jehož počátku je MIDI soubor.

Postupnými kroky je obdržen 12-rozměrný vektor, jehož dimenze postupně reprezentují noty chromatické stupnice[11]: *[C, C♯, D, D♯, E, F, F♯, G, G♯, A, A♯, B]*. Hodnoty vektoru jsou binární a v případě, že je nota v daný čas přítomna, je na její pozici hodnota 1. Například pokud je sekvence not *C, E, G* vypadají hodnoty vektoru následovně: *[1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]*. Výsledné hodnoty jsou v práci mapovány pomocí sférické projekce.

Poptávka po algoritmech tohoto typu je především ze stran velkých společností pro streamovanou[12] hudbu jako jsou *Spotify*, *Deezer*, *Apple Music*, *Google Play Music* a další. Některé společnosti jako je *Spotify* umožňují vývojářům používat své nástroje pro analýzu, nebo alespoň výstupy z analýzy zkrze API, zpřístupněného pro veřejnost.

Organizace a události

Vyhodnocování výsledků v oblasti MIR a dalšímu zkoumání tohoto oboru se věnuje hned několik organizací, pracovišť a jednotlivců, například organizace ISMIR[13] a událost MIREX[14]:

ISMIR je nezisková organizace, mimo jiného dohlížející na organizaci *ISMIR* konference. Tato každoroční konference je předním světovým fórem ve zpracování, hledání, organizování a přístupu k datům souvisejícím s hudbou.

MIREX je každoroční vyhodnocování MIR algoritmů, spojené s *ISMIR* konferencí. Je to soubor komunitně definovaných formálních hodnocení, prostřednictvím kterých je vyhodnocována široká škála nejmodernějších systémů, algoritmů a technik za kontrolovaných podmínek.

[11] Stupnice, rozdělující oktávu, interval mezi dvěma tóny, jejichž poměr frekvencí je 2:1, na dvanáct stupňů, mezi nimiž je vzdálenost jeden půltón.

[12] Streamování je technologie kontinuálního přenosu audiovizuálního materiálu mezi zdrojem a koncovým uživatelem.

[13] International Society for Music Information Retrieval

[14] Music Information Retrieval Evaluation eXchange

KAPITOLA 2

Nástroje pro hudební analýzu

V této kapitole lze nalézt několik vybraných nekomerčních programů, knihoven a API pro analýzu hudby. Často v nějaké podobě využívají Fourierovu transformaci stručně popsanou v sekci 1.4 předchozí kapitoly.

2.1 Sonic visualiser

Sonic Visualiser [9] je aplikace pro zobrazování a analyzování hudebních souborů. Je dostupný zadarmo pro Linux, OS X i Windows a distribuovaný pod GNU licencí. V základu nabízí různé možnosti vizualizace skladby:

waveform - průběh signálu; vzhled grafu zachycujího závislost okamžité hodnoty signálu na čase,

spectrogram - vizuální reprezentace spektra frekvencí,

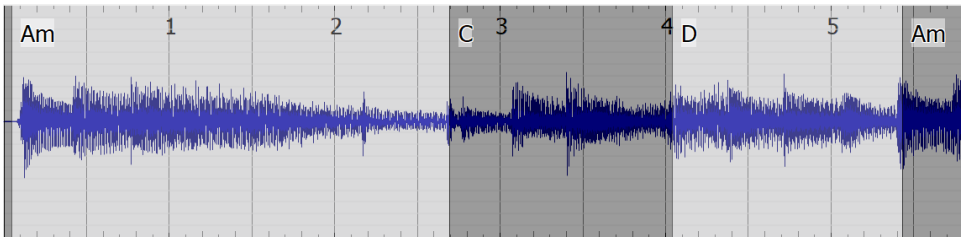
melodic range spectrogram - spektrogram melodického rozsahu,

peak frequency spectrogram - spektrogram nejvyšších frekvencí,

spectrum - amplitudy frekvencí v daném čase.

Výhodou je sbírka specializovaných *pluginů*[15] třetích stran, vhodných pro výzkumnou analýzu i pro hudebníky. Na obrázku 2.1 je vizualizováno prvních 5 vteřin skladby Hurt od Johnyho Cashe a použití pluginu *Chordino* k odhadu akordů.

[15]Plugin je doplňkový modul aplikace rozšiřující její funkčnost.



Obrázek 2.1: Program *Sonic Visualiser* a plugin *Chordino* pro odhad akordů

2.2 Spleeter

Běžná audio nahrávka má všechny nástroje a vokály neoddělitelně spojeny a je proto těžké manipulovat například pouze se zvukem klavíru. Francouzská online streamovací služba *Deezer* představila svoji open-source knihovnu, která dokáže audio rozložit na několik částí. State-of-the-art[16] knihovna Spleeter[10] je napsána v programovacím jazyce Python a používá platformu pro strojové učení TensorFlow. Poskytuje natrénovaný model pro separaci audia na následující části:

- vokály / doprovod (2 stems)
- vokály / bicí / basy / ostatní (4 stems)
- vokály / bicí / basy / klavír / ostatní (5 stems)

Rozdělení audia pomocí Spleeteru lze bez nutnosti instalování knihoven na různých webových stránkách, či lokálně na svém počítači. V dokumentaci programu Spleeter [11] je například uvedeno, že při spuštění na GPU lze oddělit zvukové soubory na 4 stems 100 krát rychleji, než v reálném čase.

2.3 WaoN

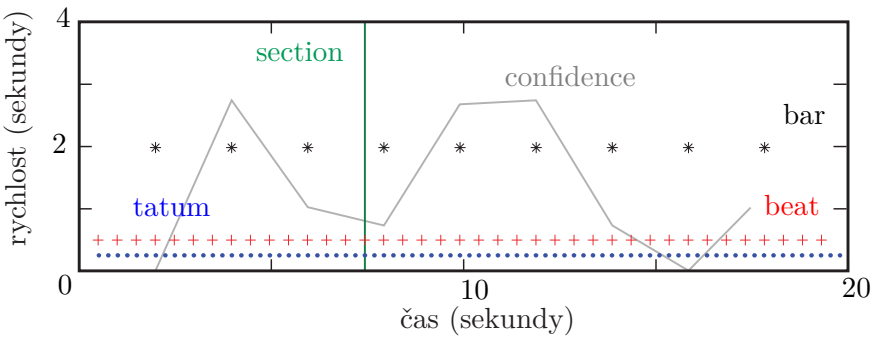
WaoN (neboli Wave-to-Notes) je program, který napsal Kengo Ichiki. Slouží jako Unixový nástroj pro převod zvuku do not. Kód je dostupný pod licencí *GNU General Public License v2.0*. Je složen ze tří programů:

waon – konvertor z WAV do MIDI formátu

pv – fázový vokodér[17] pro *time-streching*, neboli proces pro změnu rychlosti či trvání audio signálu za zachování stejné výšky (*pitch*) a *pitch-shifting*, neboli transpozice

gwaon – Graphical User Interface (GUI) pro programy *WaoN* a *pv*

[16] Obecně nejmodernější, nejvyšší stupeň vývoje v oboru.
[17] Z anglického *voice encoder*; slouží k syntéze zvuku a řeči



Obrázek 2.2: Dvacet sekund skladby "Around the World"od Daft Punk a vizualizace odhadovaných událostí v hudbě.

2.4 Spotify API

Spotify je jedna z mnoha služeb, nabízející streamovanou hudbu a podcasty. Uživatelé mohou kromě přehrávání zvukových záznamů například vytvářet veřejné, soukromé a sdílené playlisty.

The Echo Nest, platforma, kterou od roku 2014 vlastní *Spotify*, se specializuje na vývoj v oblasti hudby, umožňuje vytvářet algoritmická hudební doporučení, a tím tak předpovídat, jaká skladba se uživateli bude líbit.

Analýza skladby (Get Audio Analysis)

Některé ze *Spotify* API *endpointů* jsou vhodné k vizualizaci skladeb. Analýza skladby mezi ně patří, jelikož poskytuje odhadované události v konkrétní skladbě. Skladbu dělí na *bars* (takty), *beats* (doby), *sections* (sekce), *segments* (segmenty) a *tatums* (tatumy). Význam rozdělení je popsán v tabulce 2.1 a graficky zakreslen v obrázku 2.2.

Dělení skladby	Popis objektů rozdělující skladbu
Sekce	Nejdelší rozlišení skladby. Sekcí bývá označena například sloka, verš, kytarové sólo a tak dále.
Takty	Označuje začátek taktu, skládá se z několika dob.
Doby	Beat, neboli česky doba, je základní časová jednotka hudby. Například tik metronomu.
Segmenty	Segmenty, do kterých je skladba rozdělena, obsahují zhruba konzistentní zvuk po dobu svého trvání.
Tatumy	Tatum reprezentuje nejnižší intuitivně slyšitelný pravidelný puls.

Tabulka 2.1: Význam objektů získaných z analýzy skladby přes *Spotify* API.

Skladba je rozdělena na překrývající se úseky (viz. Tabulka 2.1) od nejdéle trvajících úseků po ty nejkratší. **Sekce**, **takty**, **doby**, **segmenty** a **tatumy** mají všechny atributy *start* (začátek konkrétního objektu), *duration* (dobu trvání) a *confidence* (jistotu správného označení).

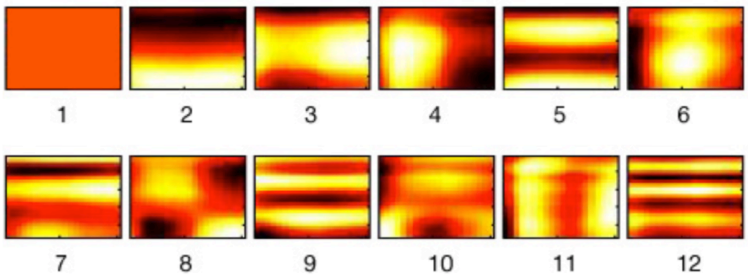
Sekce sekce v sobě obsahuje navíc atribut celkové hlasitosti v decibelech vhodný pro porovnávání relativní hlasitosti sekcí. Další atributy jako tonalita (mollová či durová), tempo, tónina a taktové předznamenání obsahují každý po jednom dalším atributu *confidence* udávající jistotu odhadu.

Segment obsahuje navíc atributy podrobněji definující hlasitost segmentu (počáteční, nejvyšší a koncovou hlasitost segmentu v decibelech a čas ve kterém hlasitost segmentu dosáhne maxima) a přidává chromatický vektor určující *pitch* podobný vektoru zmíněném v kapitole 1.5 o MIR.

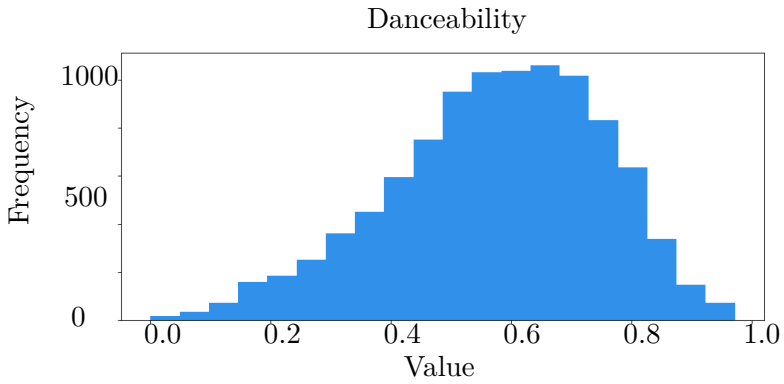
Barva (témbr) segmentu je určena ze spektrogramů a má tedy vysokou úroveň abstrakce. Je získána jako 12-dimenzionální vektor a pro správnou interpretaci je vyžedována znalost či intuice ve čtení spektrogramů. Dokumentace popisuje první dimenzi jako reprezentující průměrnou hlasitost segmentu, druhá dimenze vyjadřuje ”jas”, třetí nejvíce koreluje s ”plochostí” zvuku, čtvrtá zvuky se silnějším ”attackem”. Na obrázku 2.3 je vektor a jeho dimenze. Vodorovná osa udává čas, svislá osa frekvenci a vynesena je pak amplituda v daném čase a frekvenci.

Zkoumaná doména	Popis upřesňující zkoumanou doménu
Acousticness	Pravděpodobnostní odhad akustičnosti skladby.
Danceability	Odhad jak moc je skladba vhodná k tanci v závislosti na kombinaci ukazatelů jako je tempo, stabilita rytmu, síla, a celková pravidelnost.
Energy	Mezi vjemové rysy přispívající k tomuto atributu patří dynamický rozsah, vnímaná hlasitost, zabarvení a rychlost nástupu. Energetické skladby působí rychle, hlasitě a hlučně.
Instrumentalness	Atribut udává pravděpodobností předpověď, jestli skladba je bez vokálů. Když je hodnota hodně blízko 1.0, patrně jde o skladbu beze slov.
Liveness	Pravděpodobnost přítomnosti publika v nahrávce.
Loudness	Hladina intenzity zvuku v decibelech (dB). Užitečné pro porovnání relativní hlasitosti stop.
Speechiness	Odhad přítomnosti mluveného slova (např. recitativ).
Valence	Vysoká valence znamená pozitivní (šťastné, radostné, euforické) skladby, nízká negativní (smutné, depresivní, rozhněvané).
Tempo	Odhadované celkové tempo v BPM (beats per minute).

Tabulka 2.2: Tabulka odhadovaných vlastností skladby.



Obrázek 2.3: Dvanáct bazických funkcí vektoru reprezentující barvu segmentu.



Obrázek 2.4: Rozdělení hodnot funkce pro atribut *danceability*.

Vlastnosti skladby (Get Audio Features)

Ke každé skladbě je možné získat dobu trvání, tóninu, tonalitu a taktové předznamenání[18]. Další atributy[19] popisuje Tabulka 2.2. Ukázka průběhu atributu odhadujícího vhodnost skladby k tanci demonstruje Obrázek 2.4. Grafy[20] k dalším uvedeným atributům se nacházejí v Příloze C. Atributy jsou v rozsahu od 0 do 1, kromě atributu udávající tempo (BPM) a hlasitost (dB).

Acousticness*, *speechiness*, a *instrumentalness jsou atributy vhodné pro detekování absence zkoumaného jevu a původně byly zamýšleny jako binární klasifikátory. Lze díky tomu vyřadit skladby, které nejsou čistě akus-

[18] Připomenutí: vše kromě doby trvání je pouze odhad. Přesné algoritmy, které Spotify používá nejsou známy, ale lze se domnívat, že jde o podobné postupy, jako jsou použity v práci [8]. Na platformě *Youtube* je přednáška[12] o *Spotify API*.

[19] Dané termíny v tabulce 2.2 se nepřekládají, aby se předešlo vzniku podivných neologismů. Místo toho je uveden jejich význam.

[20] Zdrojem grafů je *Spotify* dokumentace[13].

tické, skladby které určitě neobsahují převážně mluvené slovo[21] a nebo skladby které s velkou pravděpodobností obsahují nějaké vokály.

Poslouchatelná mapa hudebních stylů

Every Noise At Once je projektem zaměstnancem firmy *The Echo Nest*. Vytvořil mapu, vizualizující spektrum více než 4300 hudebních žánrů a subžánrů, vygenerovanou ze *Spotify* dat. Každý žánr má navíc poslouchatelnou mapu všech interpretrů. Obrázek 2.5[22] je velice malým výsekem mapy žánrů.

Stránka umožňuje i různé druhy řazení. Lze dohledat oblíbené styly uživatelů Spotify v různých věkových rozmezích, nebo si zobrazit styly podle země. Například lze zobrazit poslouchatelné mapy jen českých dětských písniček, či českého black metalu.

Obrázek 2.5: Minimalistická ukázka z obsáhlé poslouchatelné mapy hudebních žánrů.

[21] Což je velmi užitečné při automatizovaném přehrávání hudby pro vyřazení projevů, bonusových skladeb, atd.

[22] Zdroj: `http://everynoise.com`

KAPITOLA 3

Generování fraktálů

Obrázek 3.1: Příklad fraktálu z přírody - zelenina romanesco.

Fraktál je geometrický objekt, který po rozdělení na menší části vykazuje tvarovou podobnost s těmito částmi. Fraktálními objekty se zabývá samostatná vědní disciplína nazývaná fraktální geometrie [14].

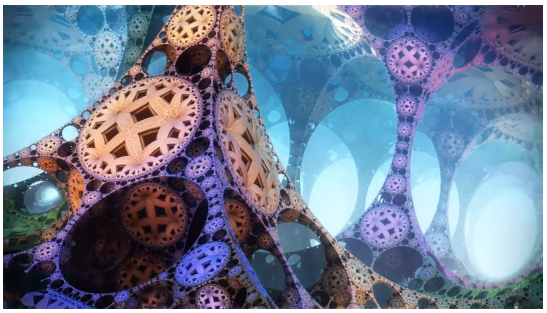
Fraktály jsou používané například pro modelování přírody a přírodních jevů (příklad na obrázku 3.1[23]), inspirování se fraktálním uspořádání (fraktální antény, komprese dat atd.), či jen z estetických důvodů.

V této kapitole lze nalézt postupy generování fraktálů za účelem nalezení vhodných způsobů pro vizualizaci. Některé fraktály lze získat více postupy generování.

[23]Zdroj: https://pixabay.com/cs/photos/zelenina-makro-květák-romanesco-659404/

3.1 Programy pro generování fraktálů

Pro počáteční inspiraci, jak se fraktály při generování chovají je možné vyzkoušet již existující software pro jejich vykreslování. Příkladem jsou specializované programy pro generování frakálů *Chaotica*, *Apophysis* a *Mandelbulb 3D* pro trojdimenzionální fraktály (obrázek 3.2a[24]). Generovat fraktály ale lze i grafickým programem *GIMP* nebo dokonce v *MS PowerPoint*[25], pro zajímavost k vidění na obrázku 3.2b[26].



(a) Fraktály v *Mandelbulb 3D*.

(b) Fraktály v *MS PowerPoint*.

Obrázek 3.2: Generování fraktálů pomocí programů.

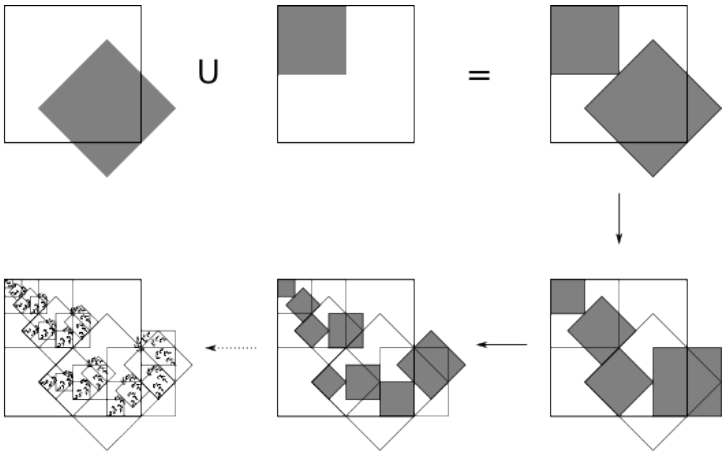
Poznámka: V následujících poznámkách budou navrženy příklady způsobů, jak fraktály animovat a tím je využít pro hudební vizualizace. Triviálním způsobem, který může být aplikovaný pro řadu fraktálů je například prolínání výsledků iteračních kroků při konstrukci (což si lze představit jako časosběr při vykreslování).

3.2 IFS

Iterated Function System, neboli systém iterovaných funkcí je jedna z metod konstrukce fraktálů. Fraktály vznikají sjednocením několika kopií sama sebe, z nichž každá je transformovaná jinou funkcí ze systému. Tyto funkce jsou kontrahující, tj. obraz při této funkci je menší než jeho vzor. Celý fraktál je tedy složen z menších kopií sebe sama, které jsou také složeny z menších kopií sebe sama, atd. Je tedy soběpodobný.

[24] Zdroj: `youtu.be/TTpbP5BVtiA`
[25] Například tak, že slide odkazuje sám na sebe.
[26] Zdroj: `youtu.be/O8l_awjgoMI`

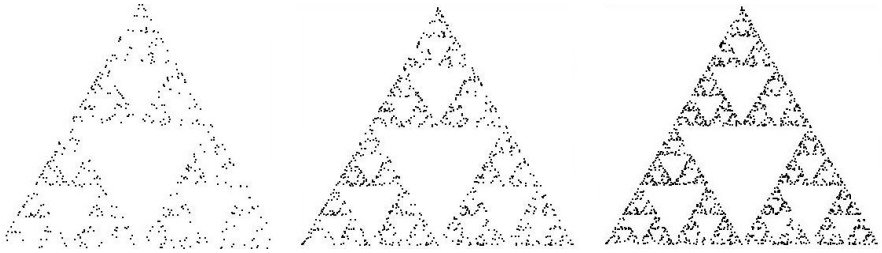
Za transformační funkce se nejčastěji[27] používají afinní transformace, které provádějí s daným objektem následující operace: rotaci, zmenšování ve všech směrech a posun. Konstrukce IFS fraktálu znázorňuje Obrázek 3.3[28]. Efektivnějším algoritmem pro generování IFS fraktálů je stochastická metoda.



Obrázek 3.3: Konstrukce fraktálu pomocí IFS.

Stochastická metoda

Fraktál vzniká z počátečního bodu, na který se aplikuje sada transformačních pravidel. Jednotlivým pravidlům se přiřazuje určitá pravděpodobnost, pravidlo tudíž vybírá náhodnou (stochastickou) cestou. Podle [15] se algoritmus někdy nazývá chaotická hra.



Obrázek 3.4: Sierpińského trojúhelník pomocí chaotické hry. Výsledky po 500, 1000 a 2000 krocích.

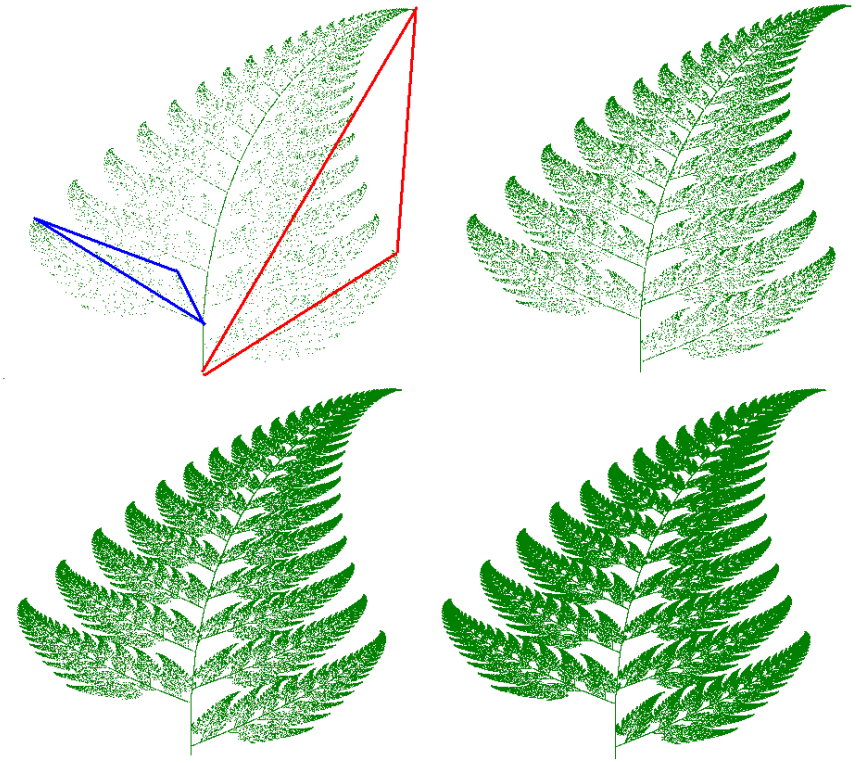
[27] Transformace mohou být i nelineární. Příkladem takových fraktálů je skupina *Fractal flame*, které se dají vytvářet v již zmíněném programu *Apophysis*.

[28] Zdroj: `https://en.wikipedia.org/wiki/Iterated_function_system`

Jednotlivá transformační pravidla lze označit jako funkce $w_1, w_2, w_3, \ldots, w_n$, kde každé funkci se přiřadí pravděpodobnost $p_1, p_2, \ldots, p_n$. Jejich součet dává 1 (100%). Funkce w_i se dá matematicky vyjádřit rovnicí 3.1, kde x, y jsou souřadnice bodu, parametry a, b, c, d určují rotaci a parametry e, f translaci[29]

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} . \tag{3.1}$$

Nový bod se stává opět počátkem pro další transformaci. Výsledek je obdržen po několika tisících iterací. Průběžné výsledky konstrukce Sierpińského trojúhelníku pomocí této metody je vidět na obrázku Obrázek 3.4[30].



Obrázek 3.5: Generování fraktálů pomocí stochastického IFS.

[29] Přesnější popis parametrů lze nalézt například zde: [16], [17].
[30] Zdroj: `https://thatsmaths.com/2014/05/22/the-chaos-game/`

	w_1	w_2	w_3	w_4
a	0.0	0.2	-0.15	0.85
b	0.0	-0.26	0.28	0.04
c	0.0	0.23	0.26	-0.04
d	0.16	0.22	0.24	0.85
e	0.0	0.0	0.0	0.0
f	0.0	1.6	0.44	1.6
p	0.01	0.07	0.07	0.85
generovaná část	stonek	levý list	pravý list	menší lístky

Tabulka 3.1: Příklady parametrů pro generování Barnsleyho kapradí.

Příklad parametrů ukazuje Tabulka 3.1. Tyto parametry generují známé Barnsleyho kapradí pojmenované po Michaelu Barnsleym, který jako první popsal tento fraktál ve své knize [18]. Zmíněnou pravděpodobnost[31] udává v tabulce parametr p.

Obrázek 3.5[32] znázorňuje postupné generování Barnsleyho kapradí podle parametrů z tabulky. Obrázek 3.6 pro zajímavost ilustruje obměnu parametrů, konkrétně nenulový parametr a vykreslující stonek, zapříčiňující vykreslování malého listu místo stonku.

Obrázek 3.6: Bernsleyho kapradí s nenulovým parametrem a (obrázek je otočen).

Poznámka: Pro animace IFS fraktálů je možné upravovat hodnoty pravidel přidáním závislosti nějakého z parametrů na čase (například těch, upravujících rotaci).

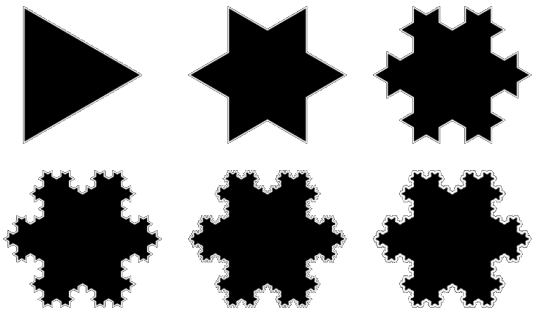
[31] Lze se domnívat, že je to z důvodu rozložení překreslovacích pravidel (které nemusí být rovnoměrné). Kdyby všech n iterací připadlo jednomu z pravidel, nebylo by dosaženo požadovaného výsledku, neb každé z pravidel by se dalo vykreslovat donekonečna.

[32] Zdroj: `http://paulbourke.net/fractals/ifs/`

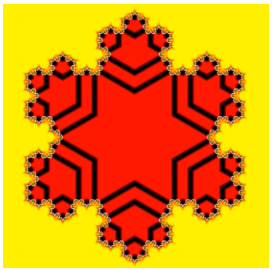
3.3 Kaleidoskopické IFS

V různých zdrojích (například [19, 20]) lze nalézt konstrukci fraktálu pomocí iterativního přehýbání prostoru, nazvané Kaleidoscopic (kaleidoskopické) IFS.

Na obrázku 3.7[33] je znázorněna klasická iterativní konstrukce Kochovy vločky. Kochova vločka vygenerovaná přehýbáním a zrcadlením prostoru je na obrázku 3.8a. Výhody tohoto komplikovanějšího způsobu se projeví při aplikaci libovolné textury (například 3.8b), která po překládání prostoru vytvoří zajímavé obrazce (například 3.8c[34]), objasňující výraz kaleidoskopické.



Obrázek 3.7: Iterativní konstrukce Kochovy vločky.



(a) Kochova vločka.

(b) Foto z Londýna

(c) Aplikace textury.

Obrázek 3.8: Aplikace textury na kaleidoskopický IFS fraktál.

Poznámka: Fraktály tohoto typu mohou být animované například posunováním textury (či obecně obsahem zpřehýbaného prostoru), nebo změnou definic přehybání prostoru (přesně tak, jak funguje kaleidoskop).

[33] Zdroj: `http://datagenetics.com/blog/january12016/index.html`
[34] Zdroj: `youtu.be/il_Qg9AqQkE`

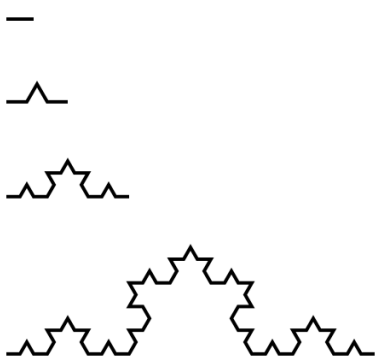
3.4 L-Systémy

L-systém nebo také Lindenmayerův systém je varianta formální gramatiky, vyvinutá pro modelování růstu rostlin. L-Systém je formálně definován trojicí $G = (\Sigma, S, P)$. Kde Σ je abeceda (neprázdná množina symbolů), S je axiom (konečné slovo z abecedy definující počáteční stav systému) a P je množina přepisovacích pravidel. Za bibli L-systémů se považuje kniha [21], kterou publikoval Lindenmayer spolu s Prusinkiewiczem.

gramatika	
abeceda:	F + -
axiom:	F
přepis. pravidla:	F → F+F–F+F
interpretace	
úhel otočení:	60°

Tabulka 3.2: L-systém - příklad pravidel pro tvorbu Kochovy křivky. Znaménka + a − reprezentují otočení doleva, či doprava. Axiom F je instrukce pro nakreslení úsečky.

V tabulce 3.2 je příklad L-systému pro tvorbu Kochovy křivky (části Kochovy vločky, jejíž konstrukce pomocí jiných postupů byly popsány výše). Na obrázku 3.9[35] je shora znázorněna 0., 1., 2. a 3. generace vygenerovaná pomocí těchto pravidel.



Obrázek 3.9: Kochova křivka vygenerovaná pomocí L-systému.

[35] Zdroj: `https://cs.wikipedia.org/wiki/L-systém`

Poznámka: Fraktály získané pomocí L-systémů lze animovat například upravováním úhlu v přepisovacích pravidlech.

3.5 Využití chaosu a dynamiky

Dynamický systém je matematický model, jehož stav je závislý na nějaké nezávislé veličině (nejčastěji to bývá čas). Je popsán pomocí dynamických podmínek, které popisují změnu tohoto systému v čase. Stav systému v libovolném časovém okamžiku je reprezentován stavovým vektorem.

Atraktor dynamického systému je množina stavů, do kterých systém směřuje. Jedná se o množinu hodnot, kterých může nabývat stavový vektor dynamického systému po dostatečně dlouhém časovém úseku od počátečního impulsu.

Teorie chaosu[36] vysvětluje proč některé fenomény jsou nepředvídatelné, i přesto, že jsou popsané matematickými rovnicemi. Fraktály jsou nekonečně členité geometrické objekty a tím tedy nekonečně komplexní. Podivné atraktory jsou spojením mezi chaosem a fraktály. Skutečnost, že dynamické systémy mají atraktory (anglicky *attractor*, z původně latinského *attrahere* - přitahovat) by se řečnicky dala vyjádřit tak, že atraktory přitahují řešení (nakonec tedy řešení bude tak komplikované jako samotný atraktor). Podivné atraktory mají tuto fraktální strukturu a jsou proto nekonečně komplikované.[37]

> „Dynamické systémy s fraktální strukturou existují i v komplexní rovině. Z těchto systémů jsou asi nejvíce známé *Juliovy množiny* a *Mandelbrotova množina*. Existuje i rozšíření Juliových množin a Mandelbrotovy množiny do hyperkomplexního prostoru. Dalším notoricky známým příkladem dynamického systému je *Lorenzovo vodní kolo*. Existují i jiné fraktály generované pomocí dynamických systémů, například *King's dream*, který vymyslel Clifford Pickover čistě z estetického hlediska."
>
> (Robert Wiesner — Užití a zneužití fraktálů [24])

Juliovy množiny

Juliovy[38] množiny mohou být vytvářeny pomocí iterace funkce komplexní paraboly 3.2. Vygenerovat Juliovu množinu lze zvolením komplexního čísla c, které se pro jeden obrazec nemění. Následně pro každý bod komplexní roviny

[36] Chaos se v tomto kontextu myslí stochastické chování v deterministickém systému, neboli nepravidelné chování podle přesných pravidel. Zdroj: [22].

[37] Text volně přeložen z [23].

[38] Byly poprvé popsány Gastonem Juliou a Pierrem Fatouem.

z nutno zjistit, zda neustálým mocněním bodu z a přičítáním konstanty c diverguje. Pokud nediverguje, patří bod do množiny.

$$z_{n+1} = z_n^2 + c \ . \tag{3.2}$$

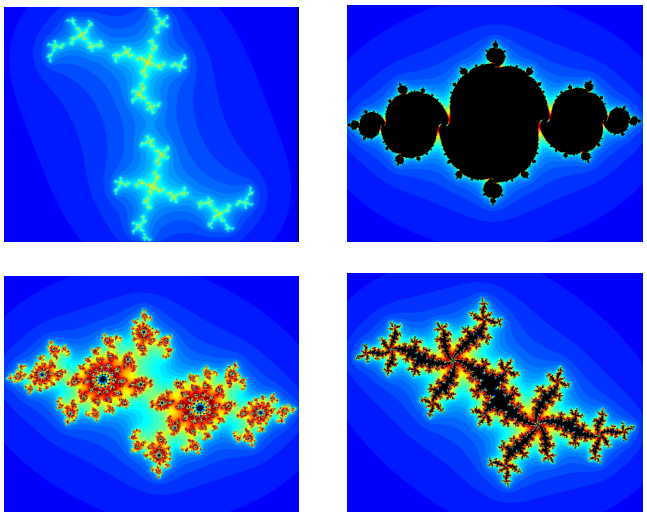
Mandelbrotova množina

Mandelbrotova množina je narozdíl od Juliových množin[39] jen jedna. K jejímu určení se používá zobrazení, které každému komplexnímu číslu c přiřazuje určitou posloupnost komplexních čísel z_n. Tato posloupnost je určena následujícím rekurzivním předpisem:

$$z_0 = 0, \ \ z_{n+1} = z_n^2 + c \ . \tag{3.3}$$

Mandelbrotova množina je pak definována jako množina komplexních čísel c, pro která je posloupnost $z_0, z_1, z_2, \ldots$ omezená, tj. splňuje následující podmínku: existuje reálné číslo m takové, že pro všechna n je $|z_n| \leq m$.

Mandelbrotova množina lze ovšem také definovat jako množina všech komplexních čísel c, kdy je Juliova množina J_c souvislá. Tento vztah ukazuje Obrázek 3.11[40], kde je na levé straně Mandelbrotova množina a na pravé straně Juliova množina.

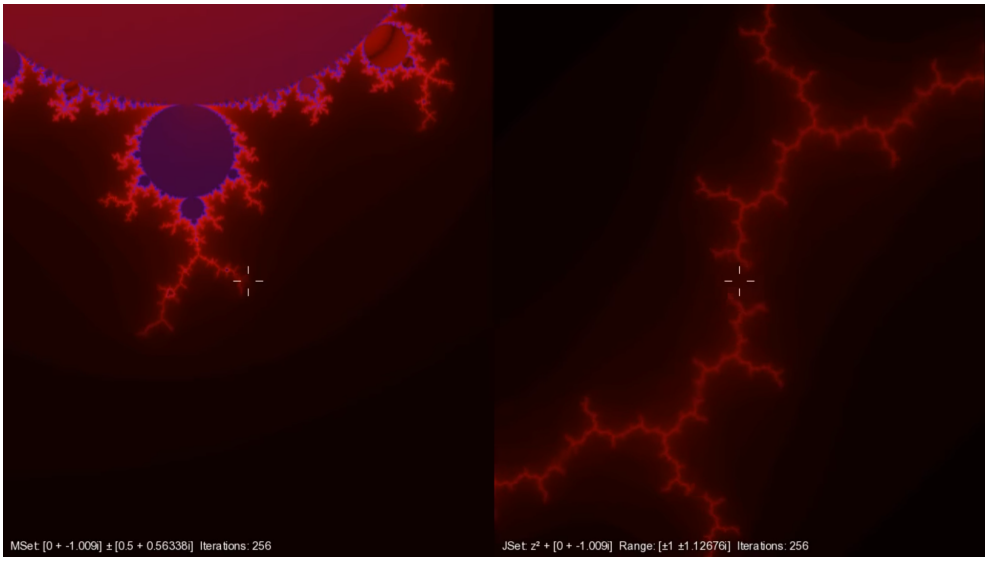


Obrázek 3.10: Animace Juliových množin pomocí vztahu $fc(z) = z^2 + C$.

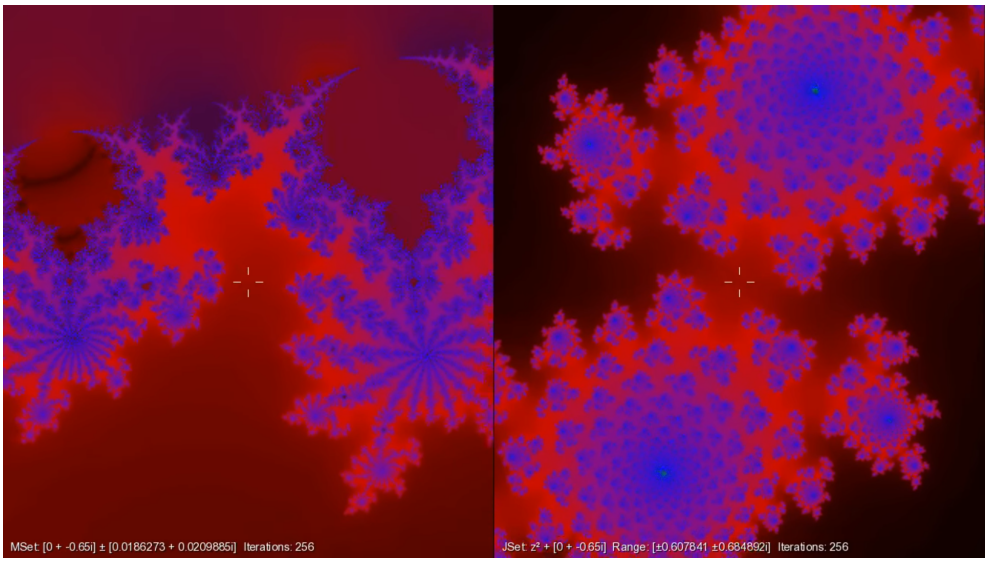
Poznámka: Fraktály obdržené pomocí dynamických systémů vybízí pro animace použít veličinu, na které systém závisí - tedy často čas.

[39] Volbou jiného komplexního čísla c vznikne jiný obraz Juliovy množiny. Juliovy množiny mohou být generovány jiným předpisem.

[40] Zdroj: `youtu.be/vfteiiTfE0c`



(a) MSet: $[0 + -1.009i] \pm [0.5 + 0.56338i]$ Iterace: 256.
JSet: $z^2 + [0 + -1.009i]$ Range: $[\pm 1 \pm 1.12676i]$ Iterace: 256.



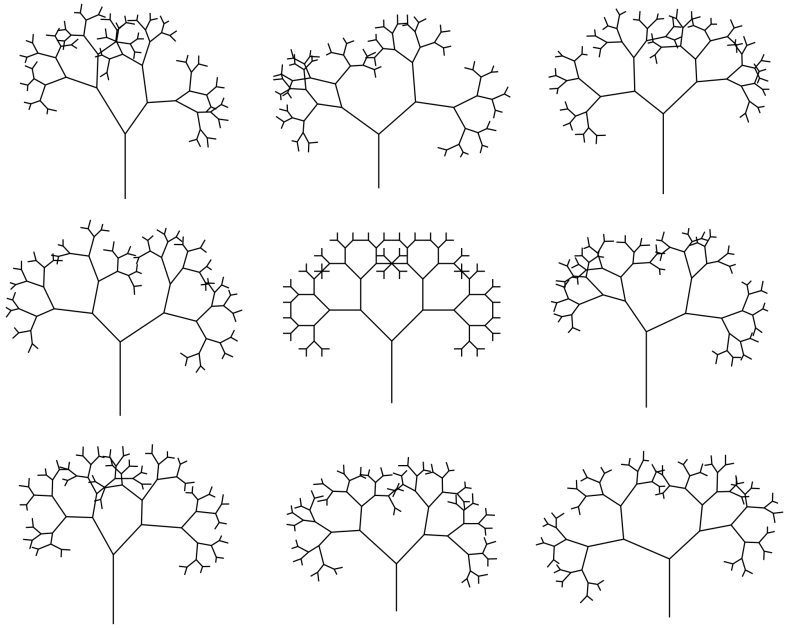
(b) MSet: $[0 + -0.65i] \pm [0.0186273 + 0.0209885i]$ Iterace: 256.
JSet: $z^2 + [0 + 0.65i]$ Range: $[\pm 0.607841 \pm 0.684892i]$ Iterace: 256.

Obrázek 3.11: Souvislost Mandelbrotovy množiny a Juliovy množiny.

Poznámka: Juliovy množiny lze animovat volbou komplexního čísla c, například deklarováním $c = 0.7885\, e^{ia}$, kde a nabývá hodnot od 0 do 2π. Průběh animace ukazuje Obrázek 3.10.

3.6 Náhodné fraktály

Náhodné fraktály se vyznačují tím, že jsou generovány algoritmy, do nichž je vnesena náhoda. Tím ztrácejí na symetričnosti a lépe tak odrážejí přírodní jevy. Výsledná podoba je ovlivněna způsobem, jak byla náhoda do generování zakomponována. Výhodou přidání náhody je i získání alternativ stejného fraktálu. Obrázek 3.12[41] je příkladem několika alternativ stromů pomocí randomizovaného úhlu otočení.



Obrázek 3.12: Stromy vygenerované pomocí L-systémů za použití randomizace.

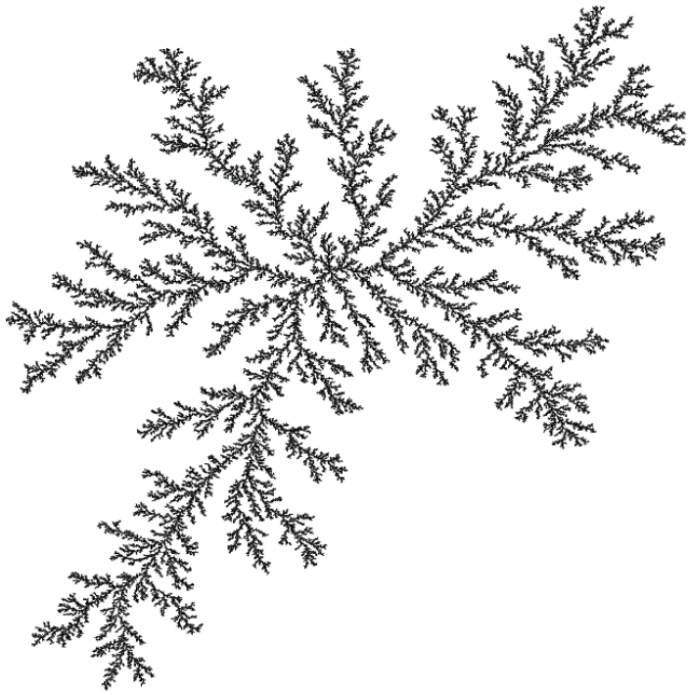
Přírodní náhodné fraktály

Fraktálem je i pobřeží (existuje i pojem "coastline paradox", neboli paradox pobřeží), neb má vlastnosti fraktální křivky, jejíž délka závisí na měřítku. Při přiblížení je možné naměřit délku navýšenou o struktury odhalené novým měřítkem. Pobřeží a obecně terén lze generovat pomocí počítačové grafiky, za využití například různých šumů, generátorů náhodných čísel a interpolace. Generováním fraktálního terénu se například věnuje práce [25] z roku 2019. Mezi příklady náhodných fraktálů tedy patří:

[41] Zdroj: `https://cs.wikipedia.org/wiki/L-systém`

- Brownův pohyb,
- *Lévy flight* (Lévyho let - alternativa náhodné procházky),
- *percolation clusters* (perkolační klastry),
- *self avoiding walks* ("sobě se vyhýbající cesty"),
- fraktální krajiny,
- trajektorie Brownova pohybu a *Brownian tree* (Brownovský strom), či
- dendritické fraktály modelovány agregací s omezenou difúzí (DLA).

Příklad fraktálu, který vznikl procesem difúzní limitní agregace zobrazuje Obrázek 3.13[42]



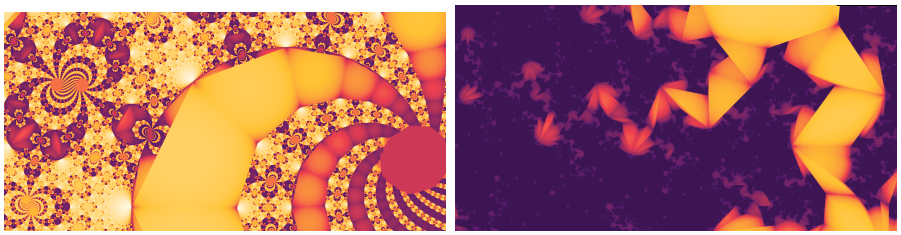
Obrázek 3.13: Přírodní fraktál (proces difúzní limitní agregace).

Poznámka: V počítačové grafice je běžné vygenerovat vegetaci (například pomocí L-systémů) a rozpohybovat ji simulováním působení větru, atd.

[42]Zdroj: https://www.researchgate.net/figure/Distribution-of-particles-determined-by-a-diffusion-limited-aggregation-DLA-process_fig3_224053708

3.7 Cohomology fractals

Cohomology fractals[43] popsali David Bachman, Saul Schleimer a Henry Segerman. Jde o fraktály, tvořené různými 3D hyperbolickými varietami. Obrázek 3.14[44] ilustruje pár možností nastavení parametrů.



(a) Fraktál v pohledovém módu vzdálenost.

(b) Fraktál pomocí jiné variety v pohledovém módu cohomology.

Obrázek 3.14: Cohomology fraktály.

> "We introduce cohomology fractals; these are certain images associated to a cohomology class on a hyperbolic three-manifold. They include images made entirely from circles, and also images with no geometrically simple features. They are closely related to limit sets of kleinian groups, but have some key differences. As a consequence, we can zoom in almost any direction to arbitrary depth in real time. We present an implementation in the setting of ideal triangulations using ray-casting."
>
> (David Bachman — Cohomology fractals [26])

Poznámka: Jednoduchá "animace" *cohomology* fraktálů by mohlo být přibližování a pohybování se v prostoru. Lze se domnívat, že dalších animací by šlo docílit jemnými přechody mezi definic variet.

[43] Jde o práci z roku 2020 uvedenou zde spíše pro zajímavost.
[44] Zdroj: `https://henryseg.github.io/cohomology_fractals/`

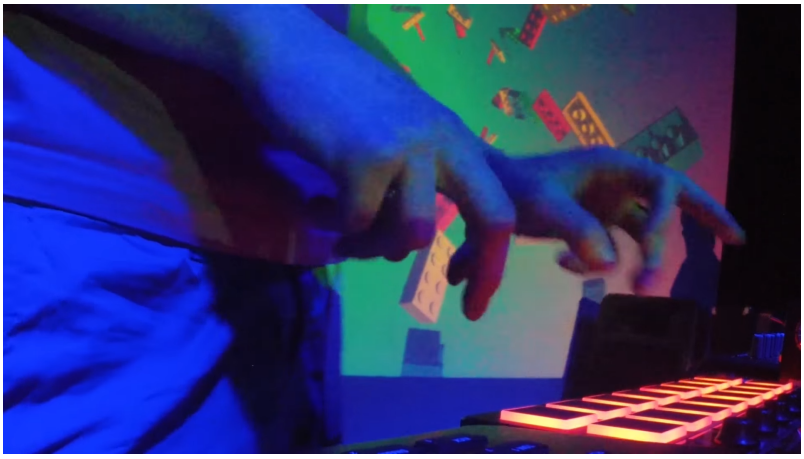
KAPITOLA 4

Příklady vizualizace hudby

Obrázek 4.1: Vizualizace skladby programu *Windows Media Player*

Nápad vizualizovat hudbu zdaleka není novinkou. Existuje mnoho způsobů, jak toho docílit a proto bude výběr v této kapitole specifikován na netriviální způsoby. Těmi se pro účely této práce bude rozumět triviální zobrazení frekvenčního spektra. Příklad ukazuje Obrázek 4.1[45].

[45]Zdroj: `https://www.windows-media-player.com/set-up-the-visualizations/`



Obrázek 4.2: Živé představení Vadima Petrova

4.1 Vizualizace živé hudby

Poptávka po vizualizacích k hudbě je žádaná při hudebních vystoupeních. Takové vizualizace by měly umět reagovat na právě hranou hudbu. To může mít na starost VJ (visual DJ) – osoba zodpovědná za zobrazování vizualizací hodících se k hudbě v reálný čas. Další možností je vytvoření programu, který přizpůsobuje vizualizace hudbě. Tyto možnosti zkoumal již Vadim Petrov[27] ve své práci, kde vytvořil i vlastní aplikaci použitou ve svém vystoupení.

4.2 Klasické vizualizéry hudby

Za *klasické vizualizéry hudby* jsou v této práci považovány programy přijímající jako vstup audio nahrávku, ke které v reálném čase generují doplňkové vizualizace.

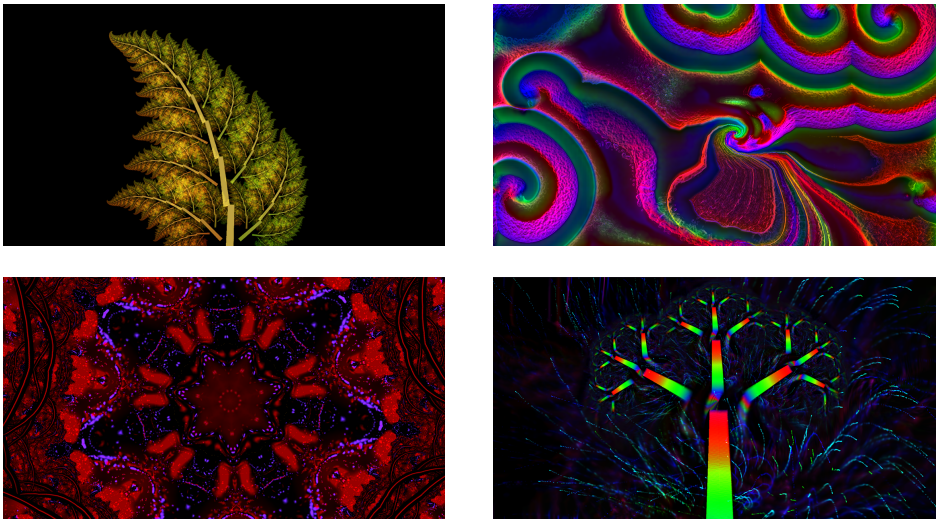
Milkdrop 2.0

Milkdrop je *plugin* programu na přehrávání hudby *Winamp* vytvořený původně v roce 2001 Ryanem Geissem. Krása *Milkdrop* spočívá v prolínavosti mezi neustále se měnícími vizuály, jak je ukázáno na obrázku 4.3.

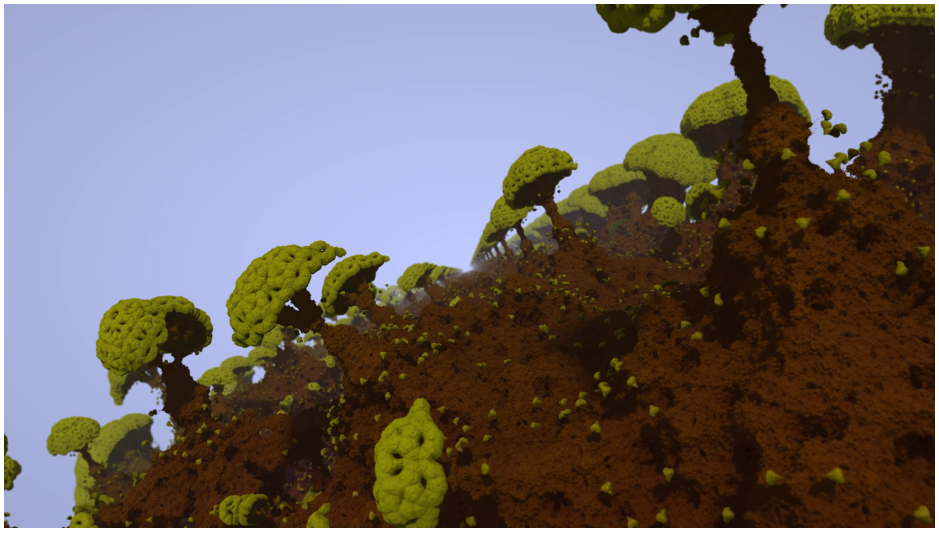
Podle dokumentace[28] používá detekci rytmu, vyvolává nesčetné psychedelické efekty a vytváří bohatou vizuální cestu zvukem. *MilkDrop* může pro vizualizace používat také živý vstup (mikrofon nebo line-in).

PySpace

PySpace od CodeParade[29] je hudební vizualizér. Je unikátní použitím renderu *Ray Marching* na vykreslování trojrozměrných fraktálů. Touto technikou



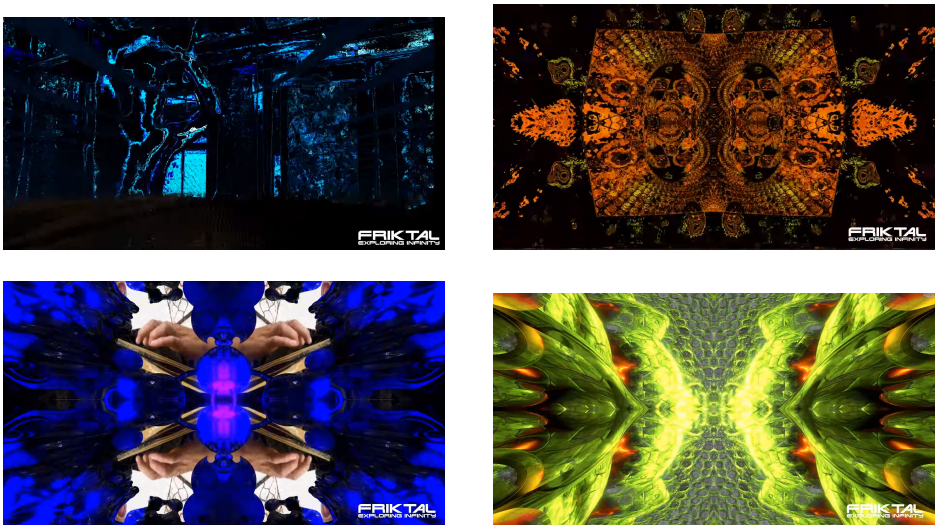
Obrázek 4.3: Plugin *Milkdrop 2.0* pro program *Winamp*



Obrázek 4.4: Hudební vizualizér *PySpace*

je možné generovat fraktály v reálném čase a relativně lehce získat měkké stíny, *ambient occlusion*, nebo dokonce kolize[46]. Ukázka PySpace je k vidění na obrázku 4.4[47].

[46] Příklad ve videu: `youtu.be/9UOXVdvQwAI`
[47] Zdroj: `youtu.be/EkZsPcsV7yE`



Obrázek 4.5: Ukázka hudební vizualizace programu *Friktal*.

Friktal

Projekt *Friktal*, mimo jiné i s aktivní komunitu na komunikační platformě *Discord*, opět využívá trojrozměrné fraktály k vizualizaci. Uživatel prochází surrealistickým světem, který reaguje na hudbu. Projekt je k červenci roku 2020 stále ve vývoji.

Aplikace je interaktivní. Do vizuální projekce je možné použít i vstup z webové kamery v reálném čase a přispívat tak do vizuální produkce. Uživatel tak může například vidět svou tvář v kaleidoskopických fraktálovitých útvarech.

Obrázek 4.5[48] je koláží ukázky z programu Friktal. V horních obrázcích byla použita interakce webkamery do vizualizací, obrázek vlevo dole znázorňuje navíc i živý vstup hraní na kytaru zachycený na mikrofon. Obrázek vpravo dole je klasický vstup z audio souboru bez webkamery.

4.3 Souběžné generování hudby a vizualizací

Tato sekce představí projekty, ve kterých je hudba s vizualizacemi úzce spjata. Jedná se o způsoby vytváření hudby zároveň poskytující obrazovou interpretaci procesu (*Reactable*, *SuperCollider*), či prakticky dokonalou vizualizaci hudby (*Animusic*, *Oscilloscope music*).

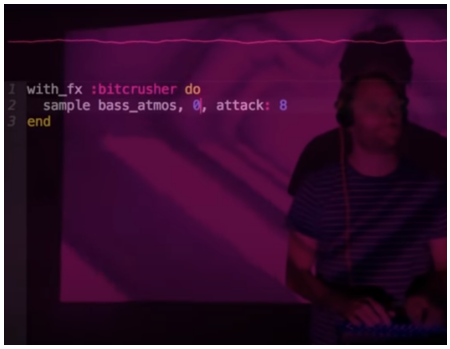
[48] Zdroj: `https://www.youtube.com/channel/UC66ZIkuYlxspf8NnmorFxqg`

Reactable

Reactable[30] je elektronický hudební nástroj s fyzickým uživatelským rozhraním. Byl vyvinut *Music Technology Group* na univerzitě v Barceloně. Projekt byl zahájen roku 2003 s cílem vyvinout nejlepší počítačový hudební nástroj nevázaný na konkrétní technologii.

Projekt ReacTable je jedním z možných použití *tangible user interface* (TUI), neboli hmatatelným uživatelským rozhraním, jako nástroj kterým uživatel interaguje s digitální informací skrz fyzické prostředí. Jedním z průkopníků *TUI* je Hiroshi Ishii, profesor Media Laboratory na MIT[49].

Obrázek 4.6: Eletronický hudební nástroj *Reactable*.

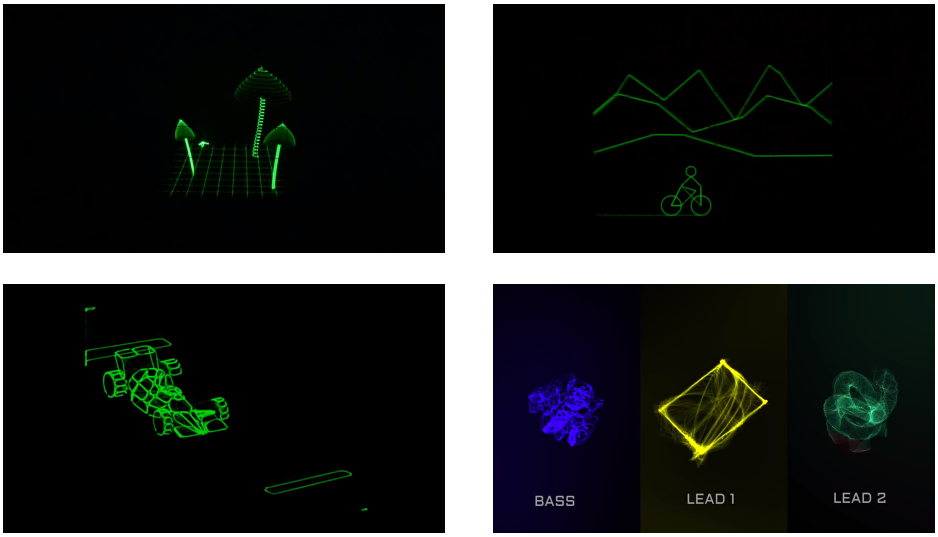


Obrázek 4.7: Živé vystoupení pomocí nástroje *Sonic Pi*.

SuperCollider

SuperCollider je platforma pro syntézu zvuku a algoritmickou kompozici. Je používán hudebníky, umělci a ve výzkumu zvuku. Může být použit pro živé programování hudby. Obrázek 4.7[50] ukazuje použití programu podobného typu jako nástroje při živém vystoupení.

[49] Massachusetts Institute of Technology
[50] Zdroj: `youtu.be/G1m0aX9Lpts`



Obrázek 4.8: Oscilloscope music

4.4 Oscilloscope Music

Myšlenka za *oscilloscope music*[51] je vytvořit audiovizuální zážitek s úzkou korelací mezi obrazem a zvukem. Vizuály jsou proto vytvářeny stejnými zvukovými vlnami, jako hudba.

Pomocí osciloskopu lze měřit a vykreslovat časový průběh měřeného napěťového signálu. V případě *oscilloscope music* je použit vektroskop, který umožňuje zobrazovat dva signály proti sobě na vertikální a horizontální ose. Vykreslován je zde levý a pravý audiokanál.

Tak by se dalo popsat nastavení, ve kterém lidé, věnující se *oscilloscope music* zobrazují jejich hudbu. Jakákoliv hudba může být na osciloskop vykreslena, klíčem umělců jako Jerobeam Fenderson je většinou vymodelovat pomocí hudby konkrétní animované scény.

4.5 Animusic

Animusic je americká společnost založena roku 1995 specializujíc se na 3D vizualizace hudby. Společnost je známá pro počítačem generované animace, které vycházejí z MIDI událostí, které mají vliv jak na hudbu, tak na vizuální efekty. Nástroje připomínají roboty hrající sami na sebe a animace se vyznačují dramaticky osvětlenými pokoji, nebo krajinou. V hudbě se nevyskytují vokály a většina zvuků je generovaná syntetizéry. Obrázek 4.9[52] je příkladem takových animací.

[51] Hudba používající osciloskop.
[52] Zdroj: `https://www.animusic.com`

Obrázek 4.9: Animusic

Část II

Realizace

KAPITOLA 5

Rozbor možností implementace

Při rozhodování který konkrétní přístup použít k implementaci, byla provedena následující úvaha výhod a nevýhod z užšího výběru možností.[53]

5.1 Možnosti analýzy audio souboru

V teoretické části práci byly představeny některé možnosti přístupu k analýze hudební nahrávce. Jedním z nich je oddělit stopy vokálů a nástrojů (případně nástroje rozdělit na více typů), například pomocí formátu *stem*. Dalším je pak použití reprezentace v MIDI formátu. Další možností je vhodně tyto přístupy kombinovat.

Využití rozdělení kanálu

Využitím rozdělení audia do více stop by se různé skupiny nástrojů izolovaly a získala by se tak nad nimi větší kontrola.

- Výhody:
  - Možnost vizualizovat odděleně vokály a doprovod.
  - Možnost rozlišit doprovod na bicí, basy, klavír a ostatní.
  - Analýza a vizualizace takto oddělených stop by byla přesnější.
- Nevýhody

[53] Již ze zadání se uvažují pouze možnosti audio nahrávky, nikoliv živého vstupu.

  - Delší implementační čas. Výsledná analýza by byla pravděpodobně podobná jako při sjednocených kanálech.
  - Problematika získávání vstupu. Musí být provedena analýza dostupnosti skladeb ve formátu *stem*.
  - Získat oddělení stop pomocí softwaru (například *Spleeter*) není úplně přesné.

Využití MIDI

Reprezentace pomocí MIDI by rozhodně měla přijít v úvahu, jelikož jde o dokonalé vyjádření hudebních událostí.

- Výhody
  - Možnost vizualizovat přesné frekvence a konkrétní nástroje, které zvuky hrají.
  - Možnost vizualizovat výstup z MIDI zařízení v reálném čase na živé vystupování.
  - K některým skladbám existuje jejich dostupná MIDI reprezentace.
- Nevýhody
  - Problematika získávání MIDI - ne vše lze získat, reprezentace není pro všechny.
  - Pro automatickou konverzi získání MIDI reprezentace je vhodnější mít oddělené stopy všech nástrojů a všech hlasů

5.2 Spotify API

Možnosti *Spotify API* byly diskutovány v sekci 2.4. Jiné existující API v této práci není uvedeno, jelikož byla posuzována pouze API služeb streamujících hudbu a to od Spotify[54] se zdá být nejrozsáhlejší.

- Výhody
  - Jedná se o již existující řešení.
  - Při integraci *Spotify API* se získá i objemná databáze skladeb.
  - Jednoduchá možnost oddělení vizualizací a přehrávání tzn. zařízení pro vizualizaci a pro přehrávání nemusí být stejné a není třeba implementovat vlastní přehrávač. *Spotify* umožňuje vybrat zařízení pro přehrávání a díky tomu je i jednoduché použití velké promítací stěny, která nemusí mít zapojeny reproduktory.

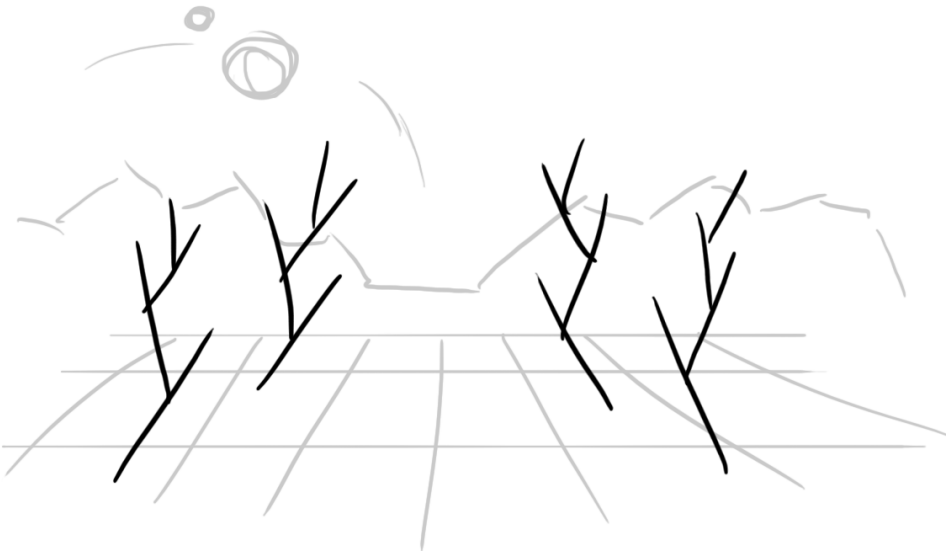
[54] Vlastnící firmu *The Echo Nest*, která se specializuje na vývoj v oblasti hudby.

- Nevýhody
  - Pro přehrávání celých skladeb nutnost účtu *Spotify*.
  - Riziko nedostupnosti a změn *Spotify* služeb.

5.3 Závěr

Bylo rozhodnuto pro využití *Spotify API*, která umožní soustředit se na grafickou stránku, poskytne databázi skladeb a jednoduchou možnost prezentování. Je zde rovněž možnost do budoucna změnit způsob získávání dat z hudby a benefitovat z existující implementace vizuálů a zkušeností s tím, na co se v implementaci vlastní analýzy zaměřit.

Obrázek 5.1 znázorňuje původní grafický návrh aplikace jako "surrealistické krajiny"s fraktály reagujícími na vizualizace. Byla zde i představa rozšíření do "galaxie planet", každá s jinými vizualizacemi hudby. Nad obzorem mělo vycházet a zapadat slunce (případně měsíc) podle časového postupu v hudbě. Vizí byla i trajektorie průletu po manifoldu a možnost při přetáčení skladby se vrátit na stejné místo ve vizualizacích.



Obrázek 5.1: Počáteční návrh aplikace *Fraviz*.

KAPITOLA 6

Postupný vývoj prototypu

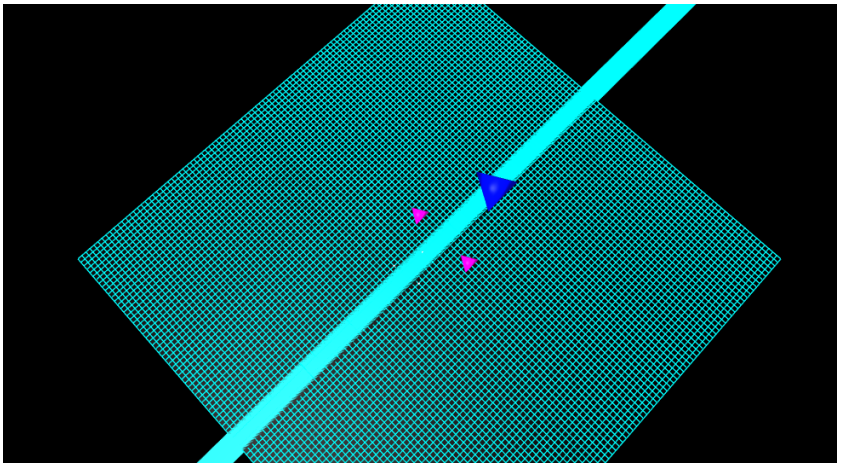
V rámci této práce vznikalo množství verzí prototypů postupně se přibližující původní vizi. Některé z nich v této kapitole budou popsány, jakožto podklady pro nadcházející verze prototypů.

6.1 Předchozí verze

Za předchůdce nápadu vizualizace hudby pomocí fraktálů by se daly považovat velmi naivní *pluginy* autorky této práce triviálně vizualizující frekvenční spektrum dané skladby na SAGE[55] a jednoduchý generátor fraktálovitých objektů podle parametrů.

Obrázek 6.1: Plugin pro Blender

[55]Framework (aplikační rámec) umožňující účastníkům přístup, zobrazení a sdílení informací v řadě rozlišení a formátů z více zdrojů na telestěnách, zpravidla z více monitorů.



Obrázek 6.2: Ukázka změny pohledu kamery

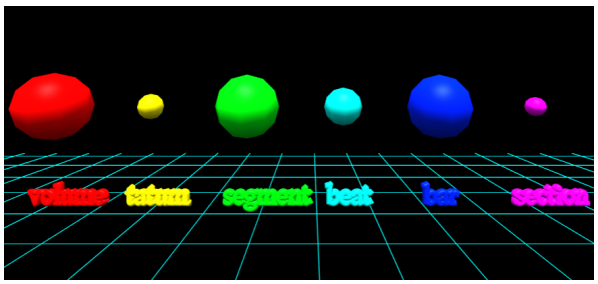
Verze využívající audio soubor

Všechny prototypy byly navrženy pro webové rozhraní z důvodu jednoduchého nasazování softwaru pro přístup veřejnosti. Pro vizualizace byla použita[56] *javascriptová* knihovna *Threejs*. Ve všech případech animace mají evokovat průlet krajinou. K tomu slouží pohybující se mřížky směrem k pozorovateli.

Spotify API

Problémem předchozích přístupů bylo získání skladeb pro přehrávání. Jedním z řešení se nabízelo dovolit nahrát uživateli soubor z počítače, či využít API existujících služeb pro streamování hudby zdarma, jako je například služba *Soundcloud*. Byla zvolena služba *Spotify* umožňující navíc využít API pro analýzu skladeb popsanou v sekci 2.4 a zároveň dovolujíc se tak více zaměřit na samotné vizualizace.

[56] Kromě prvního pokusu jednoduché mřížky bez knihoven.



Obrázek 6.3: Přiblížení verze 2.0

Fraktálovité stromy

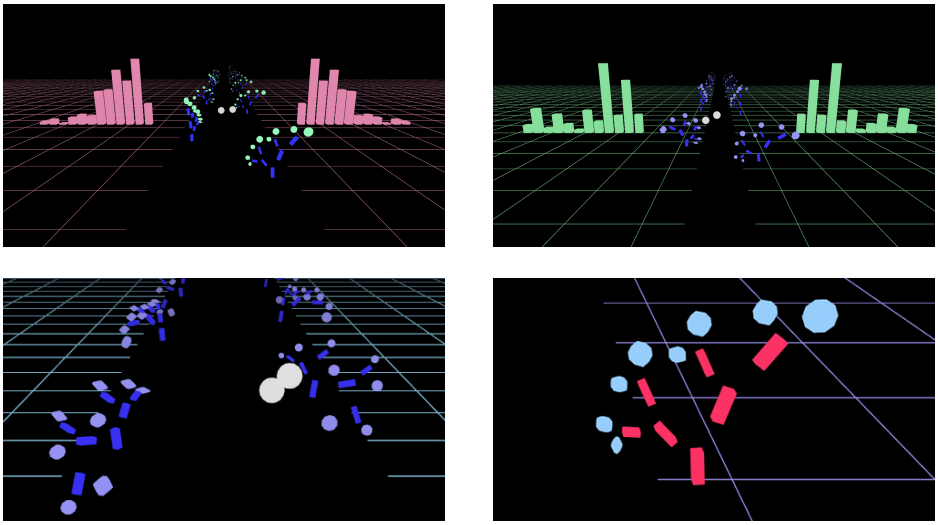
První serióznější verze používá pro vizualizace objekty, připomínající fraktálovité stromy. Tyto stromy mění svou barvu při změně *baru* a kývají se ze strany na stranu. Směr animace kývání je určován podle parity indexu *baru*[57]. Jedna strana stromů se pohybovala na pulsy *beatů*, druhá na pulsy *barů*.

Verze opět využívá chromatický vektor frekvencí pro každý nový segment. V této verzi je ovšem statický a mění se jen pozice, či škála sloupců. Jejich umístění má za účel evokovat hory. Tyto hory mění svou barvu a způsob vykreslování s každou sekcí. Sloupce jsou konstantních rozměrů a mění se jen jejich pozice na ose y, nebo se škálují na ose y. V tom případě se ještě vizualizace mohou lišit způsobem zarovnání (na střed sloupce, nebo spodní stranu sloupce).

Ve vizualizaci jsou další drobné experimentální prvky. Jedním jsou dvě malé sférické objekty obíhající konstantní trajektorii a pulsující do rytmu (původně mělo jít o světla). Druhým jsou pulsující stromy na horizontu. Pulsuje (nastaví průhlednost na maximum a postupně ji opět ztrácejí na minimum) pokud je jistota (*confidence*) určeného *beatu* větší, než hodnota 0.5. Závěrem různě testovaných hodnot bylo zjištění, že míra sepnutí této animace na základě přesnosti nelze dobře zobecnit pro všechny skladby.

Byla využita třída pro zobrazování statistik výkonu aplikace z důvodu ladění a pro informování uživatele o stavu aplikace. Lze vidět počet snímků za sekundu (FPS), doba renderování jednoho obrazu (ms) a využití paměti (MB).

[57] Což vede k nerovnoměrnému pohybu ze strany na stranu, který ke konci skladby někdy vedl k zamotání celého stromu. Strom se totiž pohyboval trochu jako rameno robota - měl složené rotace větví.

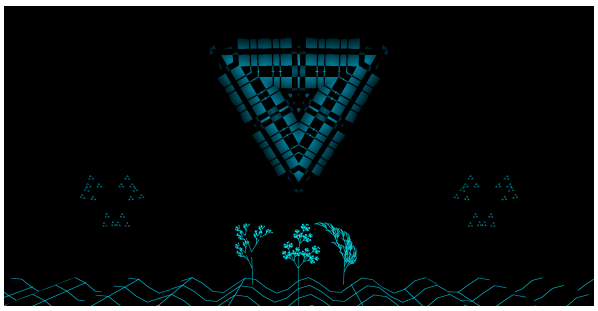


Obrázek 6.4: Vizualizace pomocí fraktálovitých stromů.

Využití shaderů

Využití klasických trojdimenzionálních objektů vyplněných souvislou barvou se neukázalo jako optimální řešení pro tvorbu fraktálů a účely této aplikace[58]. Z toho důvodu dosavadní materiály vystřídaly shadery[59], umožňující větší kontrolu nad vizualizacemi.

Kochova vločka, nebo alespoň křivka se objevuje na všech následujících vizualizacích. Je generována pomocí kaleidoskopických IFS, způsobu generování fraktálů přehýbáním prostoru popsaným v sekci 3.3. Pozdější verze aplikují měnící se texturu a tím tak utváří dojem kaleidoskopu.



Obrázek 6.5: Využití shaderů v aplikaci.

[58] Připadalo v úvahu generovat trojdimenzionální fraktály pomocí L-systémů.

[59] Počítačové programy upravující jednotlivé programovatelné části zobrazovacího řetězce grafické karty.

6.2 Popis uživatelského rozhraní

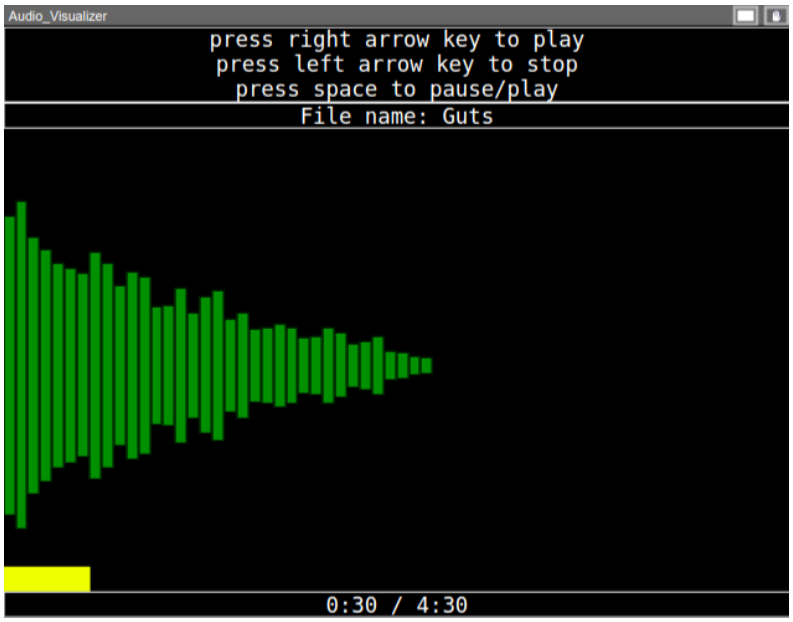
Pro výběr a ovládání skladeb se používá přehrávač aplikace Spotify. Aplikace Spotify může být spuštěna ve webovém prohlížeči, nebo ji lze stáhnout do jiného zařízení (tablet, mobil, počítač, atd.).

Dalším prvkem pro ovládání je skrytí, či zobrazení minimalistického uživatelského rozhraní, které se provádí mezerníkem. V uživatelském rozhraní je možné vidět již zmíněné statistiky výkonu aplikace společně s interpretem a názvem skladby.

Aplikace má administrátorské rozhraní, které je pro uživatele skryto. Jedná se o pomocné rozhraní pro umísťování objektů do scény a nastavování hranic pro vizualizace (například vlnová délka a amplituda mřížky). Toto rozhraní nebylo testováno.

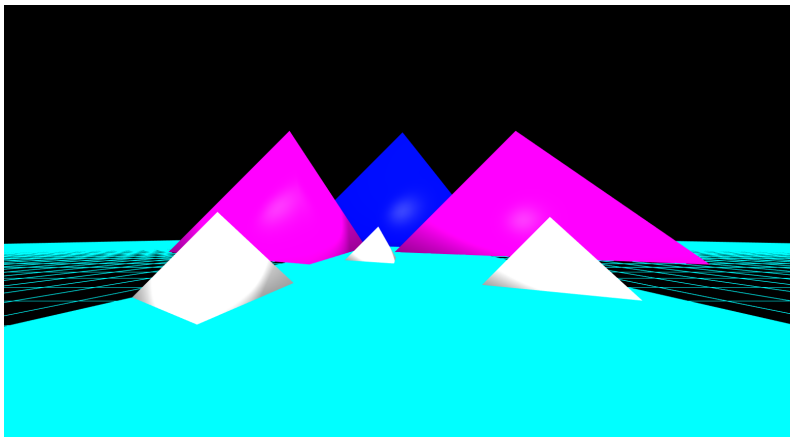
Obrázek 6.6: Statistiky výkonu aplikace.

Přehrávač pro *SAGE*



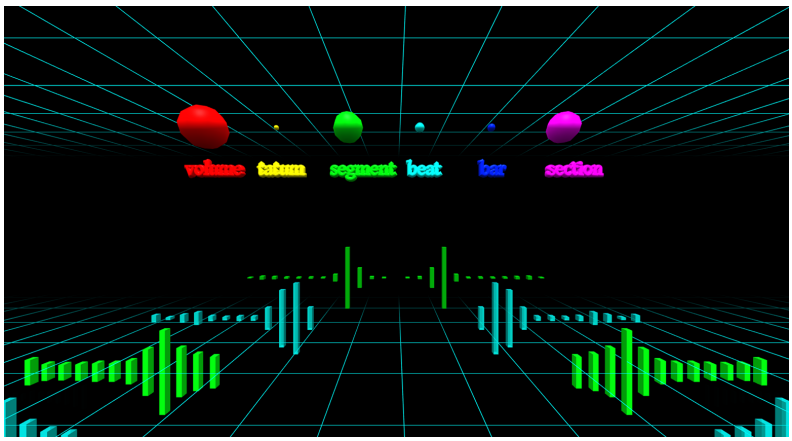
verze:	0.1
technologie:	HTML + Web Audio API
parametry:	nahrávka ve formátu mp3
krátký popis:	Prvotní verze pouze zobrazovala frekvenční spektrum a časový postup skladbou.

První Threejs verze reagující na zvukový input



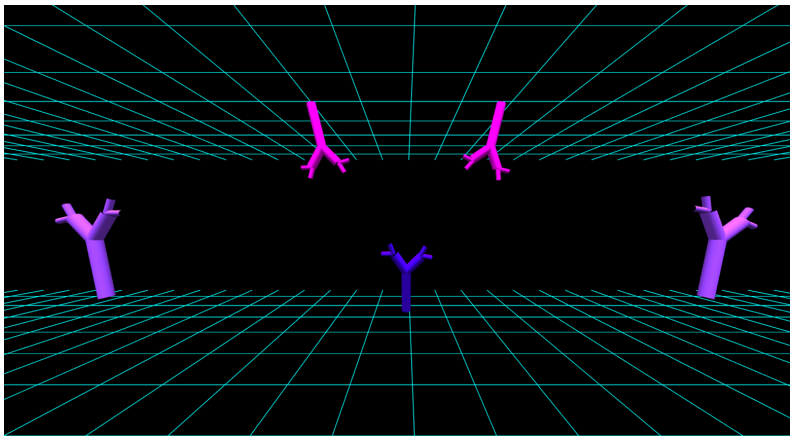
verze:	1.2
technologie:	Threejs + Web Audio API
parametry:	změna kamery, výběr z pár přednastavených skladeb
krátký popis:	Pro vizualizace byly prvně použity jednoduché mnohostěny rozdělené na tři skupiny podle barvy, které reagovaly na *beaty* v hudbě. Největší a nejvzdálenější tmavě modrý mnohostěn reprezentoval *beaty* basů, prostřední růžové objekty středy a přední bílé mnohostěny *beaty* výšek. Souvislý tyrkysový pruh měnil svou tloušťku podle hlasitosti. Rovněž rychlost průletu byla závislá na hlasitosti skladby.

Implementace spotify API



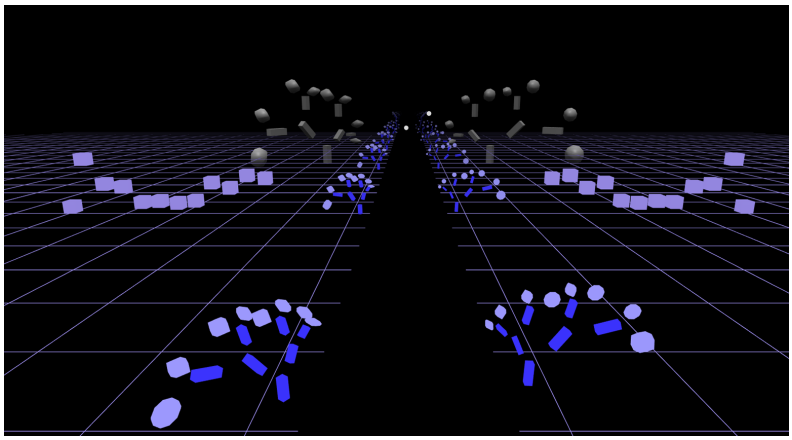
verze:	2.0
technologie:	v 1.2 + Threejs, Spotify API, spotify-viz
parametry:	hlasitost, bary, beaty, segmenty, tatumy a sekce
krátký popis:	Všechny intervaly jsou zobrazeny pomocí škálování koule poměrem celkové délky intervalu ku uplynulé délce. Díky tomu je možné ze začátku intervalu sledovat puls, který postupem intervalu mizí. Jedná se předem o verzi pro testování dat získaných ze *Spotify API*. Vektor frekvencí aktuálního segmentu popisovaný v 2.4 je zobrazen pomocí dvanácti sloupců. Tyto sloupce jsou zrcadlově otočeny proti sobě a mění se s příchodem nového segmentu.

Abstraktní stromy animované do hudby



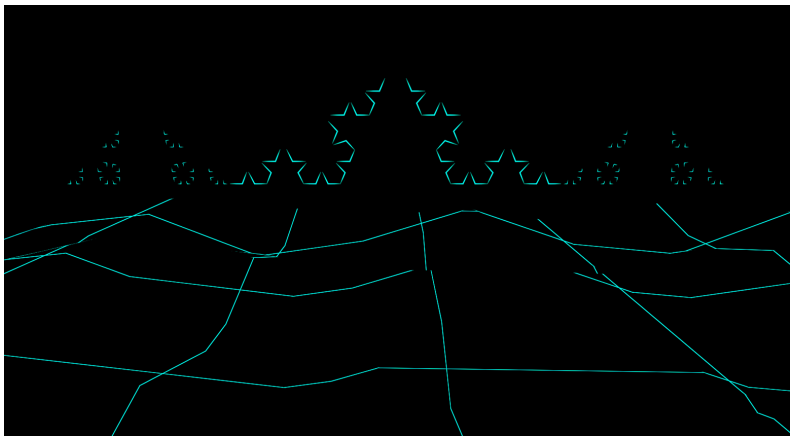
verze:	2.2
technologie:	v 2.0 + d3-interpolate
parametry:	v 2.0 + index baru a beatu
krátký popis:	Tato verze využívá interpolaci barev podle *beatu* a tím tak mění barvu abstraktních obrazců. Ty se otáčejí a škálují podle segmentů, beatů a barů. Verze byla poměrně chaotická a brzy se přešlo na jiný druh vizuálů.

Fraktálovité stromy

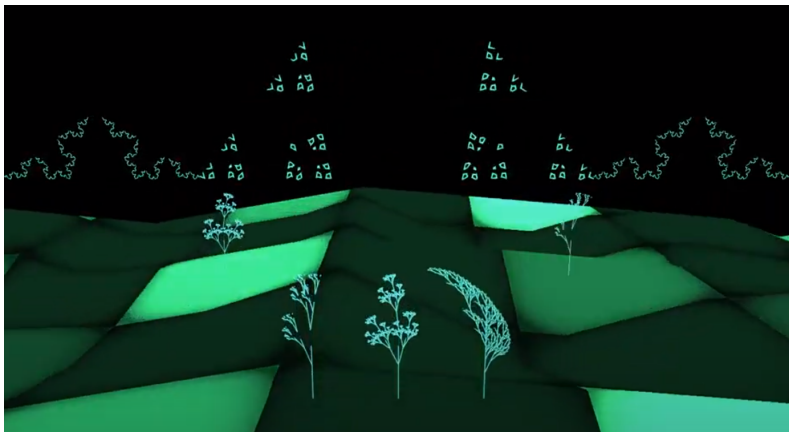


verze:	3.3
technologie:	v 2.2 + stats.module
parametry:	v 2.2 + index sekce, jistota odhadu beatu
krátký popis:	Tyto stromy mění svou barvu při změně *baru* a kývají se ze strany na stranu. Podrobnější textový popis lze nalézt v podsekci ”Fraktálovité stromy”.

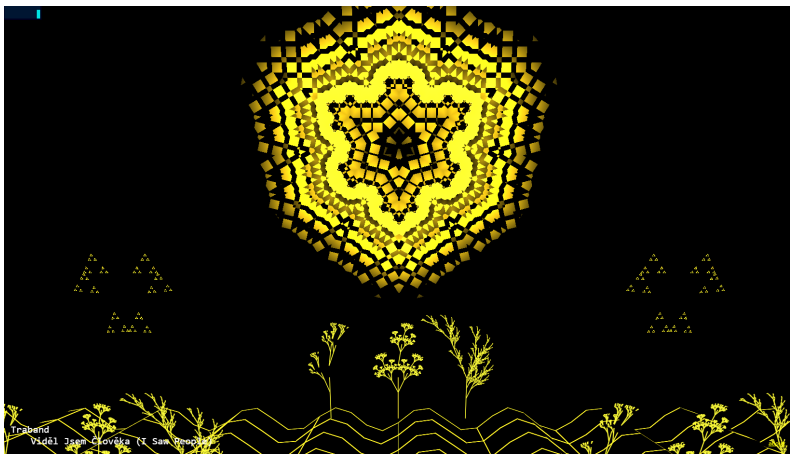
Kochovy tóniny



verze:	4.0
technologie:	v 3.3 + GLSL shadery
parametry:	v 3.3 + tonalita
krátký popis:	Společně s vizuálním posunem, aplikace nyní umožňuje přizpůsobit barevné schéma tonalitě skladby. Teplé barvy jsou pro durovou tonalitu a studené pro mollovou tonalitu. Barvy se opět střídají se změnou sekce.

Taneční parket	
	
verze:	4.1
technologie:	v 4.0
parametry:	v 4.0 + danceability
krátký popis:	Nově přibyly vizualizace fraktálů generovaných pomocí L-Systémů, které se otáčejí s postupem času a od začátku taktu se rozpadají. Stromy po stranách navíc mizí v dáli. Zároveň byl zakomponován atribut danceability, který když je větší, než daná hodnota, objeví se povrch, který má simulovat taneční plochu. Skladba splňující tuto hodnotu je například *The Prodigy - Breathe*.

Živé publikum



verze:	4.2
technologie:	v 4.0
parametry:	v 4.1 + liveness
krátký popis:	Kaleidoskopická kochova vločka byla popsána v druhém odstavci u ”Využití shaderů”. Dále u skladeb s parametrem liveness větší, než je daná hodnota je více fraktálů v popředí. Tím mají simulovat živé publikum. Jev lze vidět na skladbě *Traband - Viděl Jsem Člověka*.

KAPITOLA 7

Použité technologie

Pro lehký přístup přes internet byl prototyp napsán jako webová aplikace. V následující kapitole bude popsáno, jaké webové technologie byly použity. Finální prototyp používá *Spotify API* a pro usnadnění práce s touto službou je tento prototyp rozšířením existujícího repozitáře na stránce *github.com*. Aplikace je napsaná v programovacím jazyce JavaScript.

7.1 Programovací a značkovací jazyky

Ve stručnosti zde budou vyjmenovány programovací a značkovací jazyky pomocí kterých je napsán finální prototyp.

HTML5 je verze značkovacího jazyka *Hypertext Markup Language (HTML)* sloužícího pro tvorbu webových stránek. Proti předchozí verzi *HTML4* z roku 1997 přináší podstatné změny, přičemž mezi nejdůležitější patří přímá podpora přehrávání multimédií v prohlížeči a podpora pro aplikace, které fungují i bez připojení k Internetu.

Sass je preprocesorový a skriptovací jazyk *syntactically awesome style sheets*, který je interpretován nebo kompilován do kaskádových stylů (Cascading Style Sheets (CSS)) starajíc se o způsob zobrazení stránek psaných v *HTML*. Oproti samotnému CSS umožňuje psát kratší a přehlednější kód.

JavaScript je multiplatformní, objektově orientovaný, *event-driven* skriptovací jazyk. Odpovídá specifikaci *ECMAScript*. Patří do skupiny vyšších programovacích jazyků a multiparadigmatických programovacích jazyků. Často využívá *just-in-time* kompilaci.

7.2 Vizualizace

K vizualizacím se již od prvních verzí používala knihovna *Three.js*, později se v *Three.js* začalo využívat i shaderů.

Three.js je JavaScriptová knihovna a aplikační rozhraní sloužící k vytváření a zobrazování 3D animací ve webovém prohlížeči. Využívá aplikační rozhraní *WebGL*. *Three.js* umožňuje do webové stránky začlenit 3D animace akcelerované přes grafický procesor (GPU). Knihovnu vytvořil Ricardo Cabello, aka Mr.Doob, a je na *GitHubu* od roku 2010. Nyní je počet přispívajících uživatelů přes 1200.

OpenGL Shading Language (zkráceně GLSL) je vyšší programovací jazyk pro psaní shaderů, který vychází ze syntaxe jazyka C pro všeobecné použití. Umožňuje ovládat programovatelnou část zobrazovacího řetězce.

Knihovna d3-interpolate byla použita pro interpolaci číselných hodnot a barev. Díky *d3-interpolate* je možné například zadat barevné spektrum dvěma barvami a získat tak postupné barevné přechody podle zadaných parametrů.

7.3 Nasazení aplikace

Byl použit repozitář zaštiťující *back endovou* (serverovou) stránku aplikace a služba *Heroku* pro nasazení.

Repozitář *spotify-viz* slouží jako základ aplikace. Je dostupný na stránce *github.com* od uživatele *zachwinter* zveřejněn s licencí MIT. Repozitář umožňuje autentizaci uživatele pomocí přesměrování na stránky služby *Spotify* a tím tak začne poslouchat přehrávaná data z účtu přihlášeného uživatele. Repozitář má připravenou šablonu pro vizualizace. Výsledná stránka po přihlášení je pouze *HTML* element *canvas*, připravený na vykreslování dat z přehrávané hudby, která jsou obdržena pomocí dotazů (*request*). Repozitář používá *Node.js* framework *Express.js* a balíčkovací správce (*package manager*) *npm*. Detailnější informace lze nalézt na stránce[31] projektu.

Heroku je platforma umožňující rychlé nasazování webových aplikací. Původně podporovala pouze *Ruby*, nyní podporuje jazyky *Java*, *Node.js*, *Scala*, *Clojure*, *Python*, *PHP* a *Go*. Je ve vývoji od roku 2007, jako jedna z prvních cloudových platforem. Název je spojením slov "heroic"a "haiku".

Aplikace je dostupná na webové adrese: `http://fraviz.herokuapp.com`.

KAPITOLA 8

Testování

Tato kapitola obsahuje závěry z průběžného uživatelského testování. Výsledky těchto poznatků byly podkladem pro soustavné zlepšování aplikace v rámci jednotlivých verzí. Primárně byly testovány verze 2.0 až 3.0.

8.1 Persony

Byly vytvořeny dvě abstraktní persony pro uživatelské testování aktuálního stavu aplikace tak, aby reprezentovaly velkou část z celkové skupiny možných uživatelů.

Persona A: Student informatiky

Persona studenta, jehož věk je zhruba 20 let. Na počítači tráví většinu svého času a rozumí informatice. Občas poslouchá hudbu a rád by také ji také produkoval.

Persona B: Nadšenkyně do hudby

Persona ženy, které je zhruba 45 let se zájmem o hudbu. Počítač používá pouze uživatelsky a informatice se nevěnuje. Za to ráda poslouchá hudbu, zpívá a hraje na kytaru.

Persona C: Umělec

Persona umělce, jemuž je zhruba 30 let a věnuje se výtvarnému umění. Věnuje se i počítačovým vizualizacím, tudíž si rozumí s počítačem.

8.2 Scénáře testování

Testování bylo rozděleno na dvě skupiny. První je ovladatelnost aplikace. Možnými úkony jsou přihlášení do aplikace a ovládání skladeb. Přihlašovací systém je formou přesměrování na stránku *Spotify* a ovládání skladeb je formou *Spotify* přehrávače. Testování této skupiny nebylo hodnoceno, jelikož využívá uživatelské rozhraní třetí strany (*Spotify*).

Druhou skupinou testování, bylo subjektivní vnímání vizualizací a celkový dojem z aplikace. Osoby vyzvané k testování měly pozorovat vizualizace a sdílet své myšlenky o fungování aplikace, ty pak byly průběžně použity v rámci vývoje jednotlivých verzí.

Poznatky

Testovací subjekt A upozornil na neúměrně vysoké (především paměťové) nároky aplikace (verze 2.2), které by se měly optimalizovat. Participantovi rovněž nebyla jasná trajektorie pulsujícího a pohybujícího se sférického objektu. Na základě těchto poznatků byla trajektorie zvýrazněna duplikováním objektu, který byl krátce zpožděn a tím tak vynikla trajektorie. Problém optimalizace aplikace byl vyřešen minimalizováním objektů a především používáním shaderů (verze 4.0 a výše). Subjekt A spolu se subjektem C okomentovali ”nízkou fraktálnost”vizualizace, jež vedla k aplikaci L-systémů (verze 4.2).

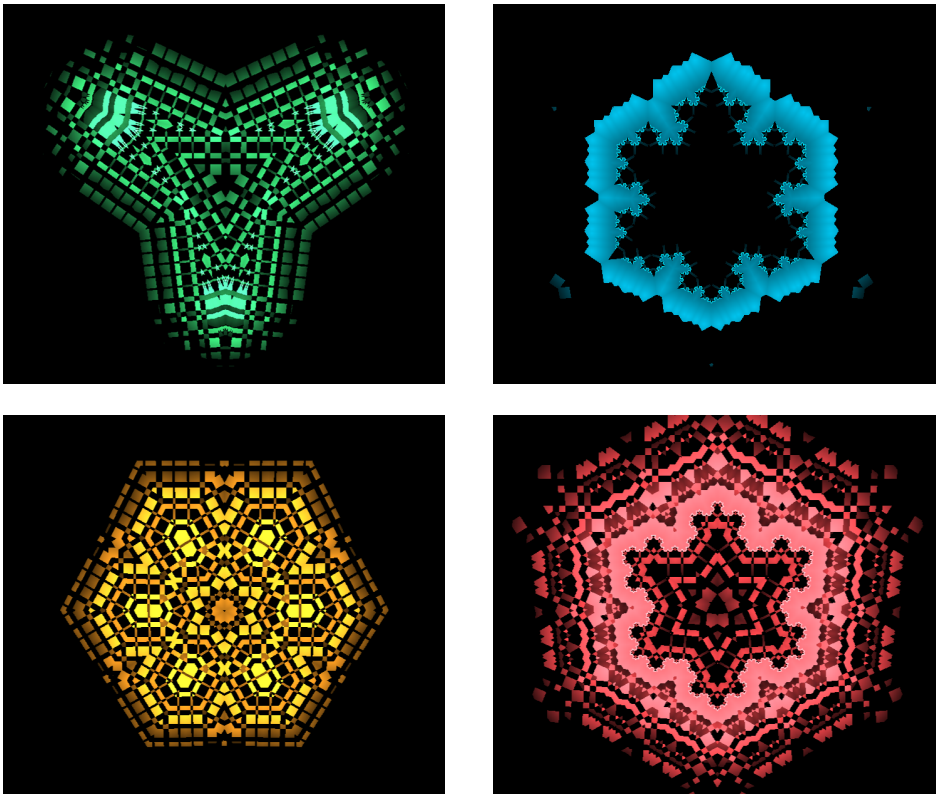
Z vizuálního hlediska (persona C) byla jako pozitivní hodnocena vizualizace hudby jakožto průletu nad údolím stromů, stejně jako responzivita mobilního rozhraní. Jako možný další podnět bylo navrženo zvýšení intenzity vykreslování, které by však mělo špatný vliv na čas vykreslení a paměťové nároky.

Z hlediska uživatelského rozhraní (osoba B) se nejproblematičtější částí ukázala (především v mobilním rozhraní) nutnost souběhu se *Spotify* aplikací a anglické varianty ovládacích popisků.

Při testování aplikace se subjekty byla hodnocena i správnost získaných atributů ze *Spotify*. Subjekty (především A a B) nesouhlasily s detekovanými atributy, především danceability.

Při vlastním testování skutečně některé z atributů od *Spotify* určeny nebyly, například mylné detekování přítomnosti publika (jednalo se například o skladbu Pink Floyd - Wish You Were Here).

Z výše zmíněného vyplynulo, že základní principy vizualizace jsou pro většinu participantů pochopitelné, ale vyskytují se typické nejasnosti ve vizualizaci, kterými trpí především kratší intervaly. Některé testovací subjekty (persona C) bez většího hudebního vzdělání/cítění nedokázaly dostatečně rozpoznat abstraktnější atributy (tónina, danceability apod.). Na základě tohoto se zdá jako ideální vytvořit databázi typických ukázek pro jednotlivé parametry vizualizace (tutoriál, guide) a tím neznalému uživateli umožnit porozumět všem modalitám vizualizace na hlubší úrovni.



Obrázek 8.1: Přiblížení ukázky kaleidoskopické Kochovy vločky.

Závěr

Před rešeršní práci v oblasti *MIR* byl sepsán úvod do digitalizace zvuku a načrtnuta základní analýza hudebních signálů. Následně byla přiblížena oblast *MIR* společně s pár vybranými vědeckými prácemi a byly zmíněny organizace a události věnující se tomuto odvětví.

V rešerši fraktálů byly diskutovány některé z možných způsobů získávání fraktálů pomocí počítačové grafiky. Záměrem bylo soustředit se spíše na různé způsoby, kterými docílit generování fraktálovitých útvarů a jak takové útvary animovat, než na definice jejich vlastností.

V analýze současných řešení byly představeny některé existující projekty využívající fraktály k vizualizaci hudby a zároveň poukázat na řešení, kde hudební produkce a vizualizace jsou úzce spjaty. Lze se domnívat, že taková řešení budou vysoce vypovídající o vztahu hudby a vizuálů.

Zpočátku nebylo jasné, jaký způsob získávání hudebních dat bude pro vizualizace využit. Bylo otestováno získávání hudebních dat z audio nahrávky a pomocí mikrofonu. Nakonec bylo použito *Spotify API* pro jeho možnosti analýzy vlastností hudební nahrávky a pro možnost získání atributů, které vznikly z (nejspíš největší existující) databáze skladeb mezi sebou. Tato volba zároveň umožnila soustředit se na vizuální stránku díky možnosti přehrát jakoukoliv (v databázi přítomnou) skladbu a odstranila tak i problém jak získávat hudební stopy pro vizualizace.

V rámci práce vzniklo mnoho prototypů, z nichž ty nejvýznamnější prezentuje Kapitola 6, kde je uveden postupný vývoj a přidávané funkce. Výsledný prototyp vizualizuje netriviálně data získaná z hudby, která vychází nejen z analýzy konkrétní skladby (doba, takt, hlasitost, sekce, segment, tonalita), ale i komplexnějších atributů jakými jsou například *danceability* (tancovatelnost), energie a tempa. Výsledek má stále prostor pro zlepšení a to zejména optimalizaci výkonnosti vizualizací.

Literatura

[1] Wikipedia contributors: Nyquistův–Shannonův vzorkovací teorém. 2019 [cit. 2020-07-10], [online]. Dostupné z: https://cs.wikipedia.org/wiki/Nyquistův-Shannonův_vzorkovací_teorém

[2] Davis, A.; Rubinstein, M.; Wadhwa, N.; aj.: The Visual Microphone: Passive Recovery of Sound from Video. *ACM Trans. Graph.*, ročník 33, č. 4, Červenec 2014, ISSN 0730-0301. Dostupné z: https://doi.org/10.1145/2601097.2601119

[3] Kabelka, R.: Teorie digitálního audia. 2020 [cit. 2020-07-10], [online]. Dostupné z: https://www.portal-pelion.cz/teorie-digitalniho-audia/

[4] Vochlea Music: Dubler Studio Kit. [software]. Dostupné z: https://www.vochlea.co.uk

[5] Holčík, L.: *Spektrální analýza hudební skladby*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2009. Dostupné z: https://is.muni.cz/th/blifg/dp.pdf

[6] Bergstrom, T.; Karahalios, K.; Hart, J. C.: Isochords: Visualizing Structure in Music. In *Proceedings of Graphics Interface 2007*, New York, NY, USA: Association for Computing Machinery, 2007. Dostupné z: https://doi.org/10.1145/1268517.1268565

[7] Chan, W.-Y.; Qu, H.; Mak, W.-H.: Visualizing the Semantic Structure in Classical Music Works. *IEEE transactions on visualization and computer graphics*, 2010.

[8] Lopez-Rincon, O.; Starostenko, O.: Music Visualization Based on Spherical Projection With Adjustable Metrics. *IEEE Access*, ročník 7, 2019.

[9] Centre for Digital Music at Queen Mary: Sonic Visualiser. [software].

[10] Hennequin, R.; Khlif, A.; Voituret, F.; aj.: Spleeter: A Fast And State-of-the Art Music Source Separation Tool With Pre-trained Models. 2019, [software].

[11] Hennequin, R.; Khlif, A.; Voituret, F.; aj.: Spleeter documentation. 2020 cit. [2020-07-10]. Dostupné z: https://github.com/deezer/spleeter

[12] Koh, M.: Audio Analysis with the Spotify Web API. 2018 cit. [2020-07-26], [online]. Dostupné z: https://youtu.be/goUzHd7cTuA

[13] Spotify: Get Audio Features for a Track. 2008 cit. [2020-21-03], [online]. Dostupné z: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

[14] Pauš, P.: *Počítačové generování fraktálních množin.* Rešeršní práce, České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská, Praha, 2003.

[15] Sixta, T.: Linearizované systémy iterovaných funkcí. 2003 cit. [2020-07-19], [online]. Dostupné z: http://chaos.fraktaly.sweb.cz

[16] Bourke, P.: Macintosh IFS manual. 1990 cit. [2020-07-20], [online]. Dostupné z: http://paulbourke.net/fractals/ifs/

[17] Čápka, D.: IFS fraktály. 2016 cit. [2020-07-19], [online]. Dostupné z: https://www.itnetwork.cz/navrh/algoritmy/algoritmy-graficke/algoritmus-ifs-fraktaly-teorie-vykresleni-a-tabulka-hodnot

[18] Barnsley, M.: *Fractals Everywhere.* Academic Press, 1993.

[19] Forums, F.: Kaleidoscopic (escape time) IFS. 2015 cit. [2020-07-20], [online]. Dostupné z: http://www.fractalforums.com/sierpinski-gasket/kaleidoscopic-(escape-time-ifs)/

[20] Wiggins, R.: 3D Kaleidoscopic Fractals: Folding the Koch Snowflake. 2017 cit. [2020-07-20], [online]. Dostupné z: http://roy.red/folding-the-koch-snowflake-.html

[21] Prusinkiewicz, P.; Lindenmayer, A.: *The Algorithmic Beauty of Plants.* Berlin, Heidelberg: Springer-Verlag, 1996, ISBN 0387946764.

[22] Stewart, I.: *Hraje Bůh kostky?* Praha: Argo, první vydání, 2009, ISBN 978-80-257-0024-2.

[23] Pyke, R.: Chaos, Fractals and Dynamics. 2000 cit. [2020-27-07], [online]. Dostupné z: https://www.sfu.ca/~rpyke/335/summary.html

[24] Wiesner, R.: *Užití a zneužití fraktálů.* Diplomová práce, Přírodovědecká fakulta Masarykovy univerzity, Brno, 2006. Dostupné z: https://is.muni.cz/th/fpil7/diplomka.pdf

[25] Kříž, J.: *Multi-Fractal Terrain Generation*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2019 [cit. 2020-07-27]. Dostupné z: https://theses.cz/id/4adcbi/

[26] Bachman, D.; Schleimer, S.; Segerman, H.: Cohomology fractals. 2020.

[27] Petrov, V.: *Nástroj pro vizualizaci datového toku formátu MIDI pro živá vystoupení*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

[28] Geiss, R.: MilkDrop. 2012 cit. [2020-07-19], [online]. Dostupné z: http://www.geisswerks.com/milkdrop/

[29] CodeParade: Audio Reactive Fractals (4K) - Let's Go for a Walk. 2019 cit. [2020-05-11], [online]. Dostupné z: https://www.youtube.com/watch?v=EkZsPcsV7yE

[30] University, P. F.: Reactable. 2007 cit. [2020-02-21], [online]. Dostupné z: https://www.upf.edu/web/mtg/reactable

[31] Winter, Z.: spotify-viz. [software]. Dostupné z: https://github.com/zachwinter/spotify-viz

PŘÍLOHA A

Seznam použitých zkratek

API Application Programming Interface.

CSS Cascading Style Sheets.

GUI Graphical User Interface.

HTML Hypertext Markup Language.

MIDI Musical Instrument Digital Interface.

MIR Music information retrieval.

PŘÍLOHA B

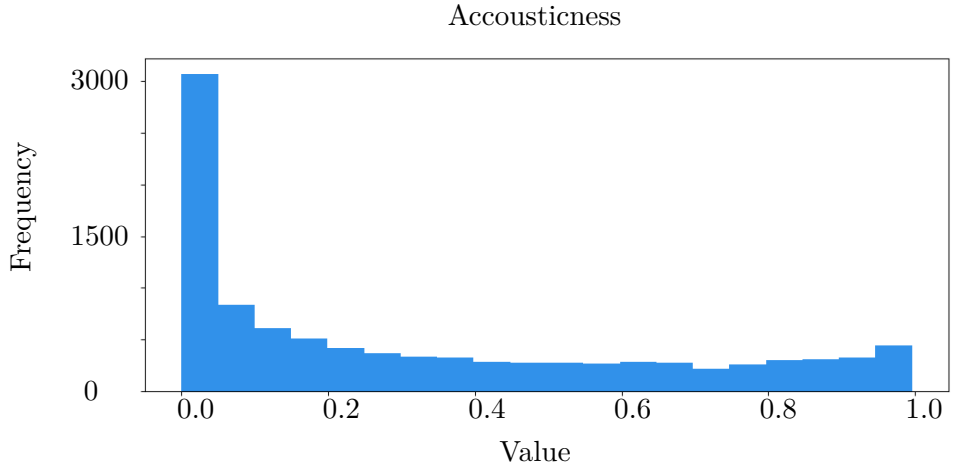
Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS

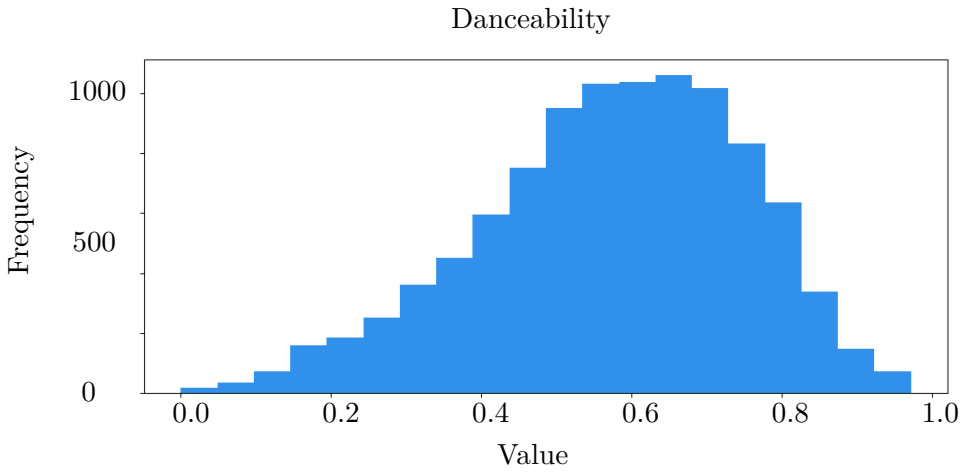
PŘÍLOHA C

Grafy detekovaných atributů skladby

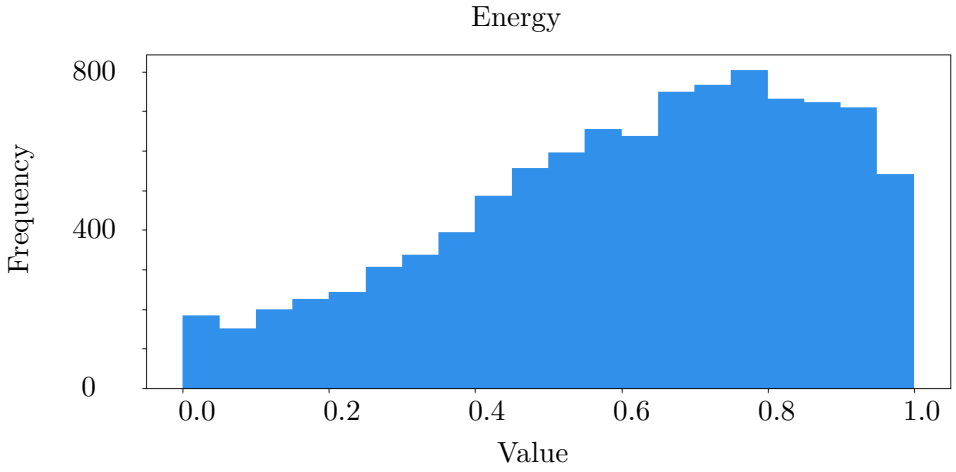
Následující grafy jsou grafy atributů debatovaných v rešeršní části práce v sekci o *Spotify API*. Opírá se o ně realizace prototypu.



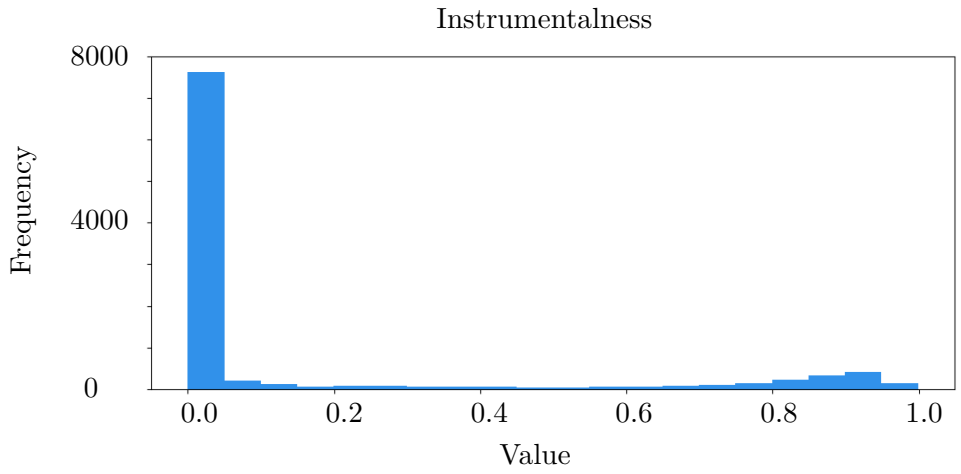
Obrázek C.1: Distribuce hodnot atributu *accousticness*.



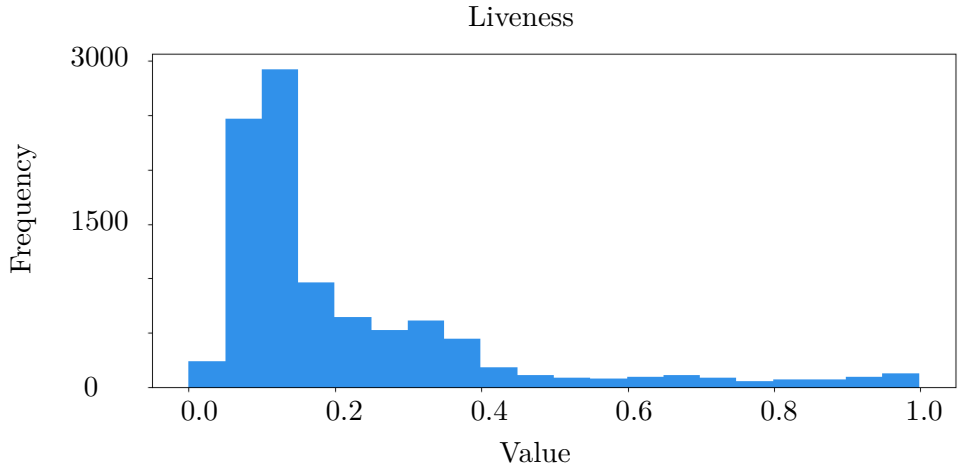
Obrázek C.2: Distribuce hodnot atributu *danceability*.



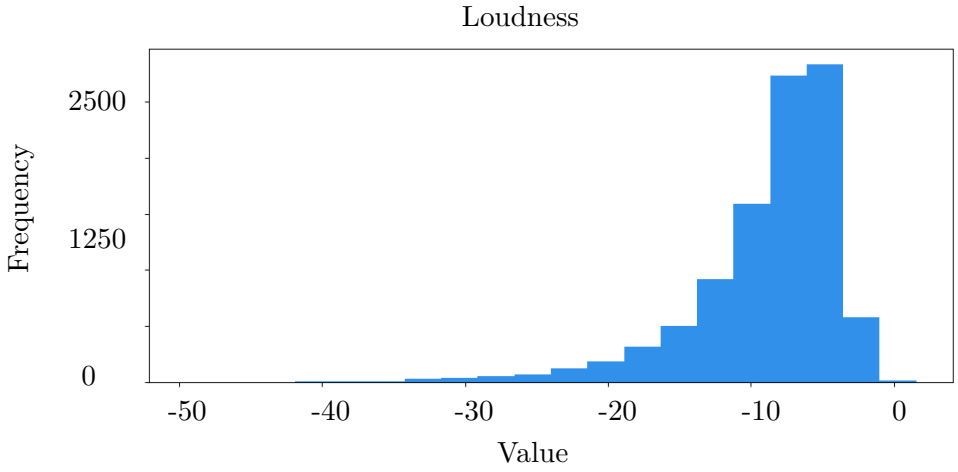
Obrázek C.3: Distribuce hodnot atributu *energy*.



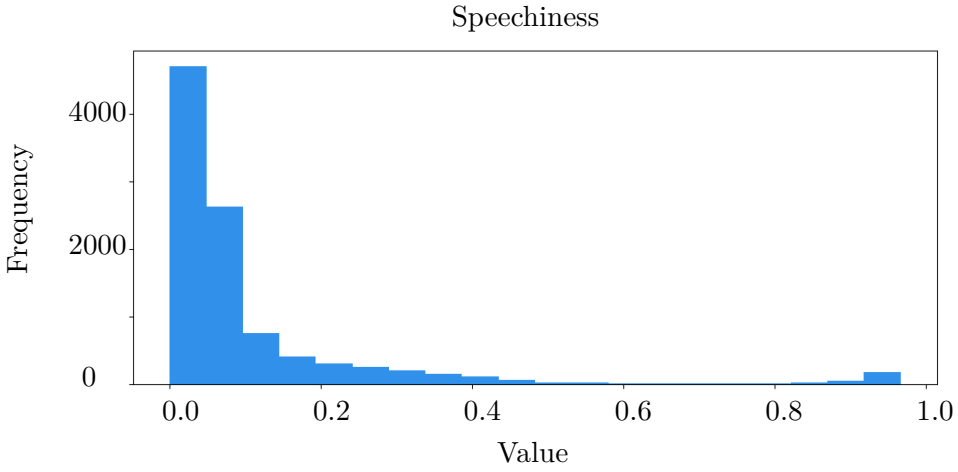
Obrázek C.4: Distribuce hodnot atributu *instrumentalness*.



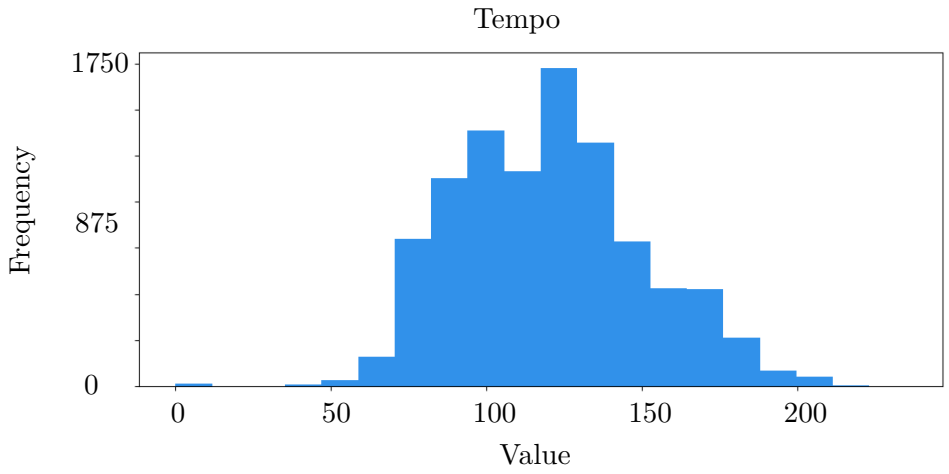
Obrázek C.5: Distribuce hodnot atributu *liveness*.



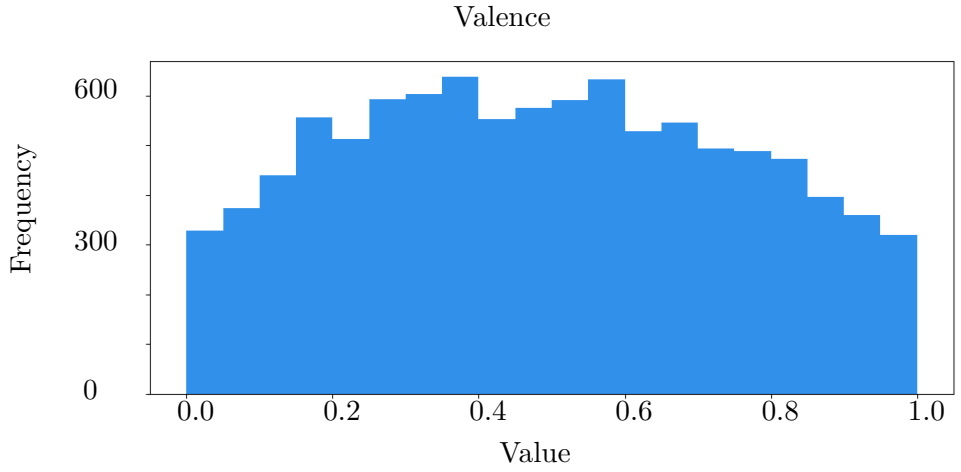
Obrázek C.6: Distribuce hodnot atributu *loudness*.



Obrázek C.7: Distribuce hodnot atributu *speechiness*.



Obrázek C.8: Distribuce hodnot atributu *tempo*.



Obrázek C.9: Distribuce hodnot atributu *valence*.