VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Jana Jelínková

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VIZUALIZACE ZVUKU
AUDIO VISUALIZER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jana Jelínková**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Kamil Říha, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Audio inženýrství**
Ústav telekomunikací

Studentka: Jana Jelínková ***ID:*** 174319
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Vizualizace zvuku

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte možnosti vizualizace zvuku na základě jeho parametrického popisu. Navrhněte a implementujte vytvoření syntetického obrazu/objektu, jehož parametry budou měněny na základě časově proměnných parametrů zvuku (hlasitost, spektrální složení, tempo…).
Doporučený nástroj pro implementaci: knihovny OpenCV, OpenGL a MS Visual C++.

DOPORUČENÁ LITERATURA:

[1] GONZALEZ, R. C.; WOODS R. E.: Digital Image Processing, Prentice Hall, New Jersey, 2002.

[2] BRADSKI, G.; KAEHLER A.: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, Inc. USA 2008, ISBN: 978-0-596-51613-0.

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: doc. Ing. Kamil Říha, Ph.D.
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cílem této bakalářské práce je vizualizace zvuku, tedy vytvoření objektu, jehož parametry budou měněny na základě časově proměnných parametrů zvuku. V první části se práce zabývá různými druhy vizualizací jak z historie, tak ze současnosti. Dále pak uměleckou teorií vztahující se k vizualizaci zvuku.
Ve druhé části práce je popsán postup řešení v prostředí Pure Data, princip získávání vybraných parametrů z nahrávky a vytvoření vlastního rozšíření pro Pure Data.

KLÍČOVÁ SLOVA

vizualizace zvuku; parametry zvuku; zpracování zvuku; synestézie; herní engine; Pure Data; Kandinskij

ABSTRACT

The aim of this thesis is to create an audio visualizer. That means an object, whose parameters will be changed in real time based on chosen parameters of an audio. The first part of this thesis deals with different kinds of audio visualizers through history till today and also deals with some artistic theories about visualization.
The second part deals with the main solution of the visualizer in Pure Data and principles used for audio processing and also describes development of an external for Pure Data.

KEYWORDS

sound visualisation; parameters of sound; audio processing; synesthesia; game engine; Pure Data; Kandinskij

JELÍNKOVÁ, J. *Vizulizace zvuku*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 47 s. Vedoucí práce byl doc. Ing. Kamil Říha, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Vizulizace zvuku“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu doc.Ing.Kamilu Říhovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Tato práce se věnuje softwarovému zpracování parametrů zvuku a jejich grafickému zobrazení v reálném čase. V první časti se zabývá obecnou teorií zvuku. Dále rozebírá vývoj a typy vizualizace zvuku a nastiňuje momentální situaci na poli multimediální produkce. Jelikož se jedná o vizualizaci parametrů zvuku, která nevzniká za účelem měření těchto parametrů, je nutné, aby se práce řídila i jistými uměleckými doporučeními. Teorií spojování barev a hudby, doporučeními ohledně tvarů a využití prostoru se zabývám v podkapitole Propojení zvuku, barev a tvarů. Dále následuje popis technického řešení zahrnující použitá prostředí, postup tvorby a principy, kterým dané parametry z nahrávky získáváme. Tato práce se zabývá celkem třemi parametry, a to skutečnou špičkovou hodnotou - amplitudou, barvou zvuku a tempem nahrávky. Všechny tyto parametry budou vizualizovány do jednoho zobrazovacího okna na model koule - planety.

1 TEORETICKÝ ÚVOD

Zvuk je pro člověka zdrojem informací. Díky zvukům přicházejícím z našeho okolí si utváříme představu o poloze, vzdálenosti, rychlosti či velikosti jednotlivých prvků v prostoru. V praxi existují dva pohledy na zvuk.

1.1 Základní informace

1.1.1 Umělecký popis

Z hudebního hlediska jsou zvuky rozlišeny na tóny a hluky. Hluk je zvuk, u kterého nemůžeme přesně určit výšku. Tón je označován jako hudební zvuk, který je zapisován primárně pomocí notového zápisu. Nota o tónu podává dvě informace: výšku a délku. Síla zvuku je v hudební terminologii nejčastěji popisována slovně či zkratkami. Barvu zvuku určuje z velké části použitý hudební nástroj a jeho rejstříky. Na barvě se však podílí i razance a výraz, se kterým hráč danou notu interpretuje.

1.1.2 Technický popis

Z technického hlediska je zvuk energie nebo informace s určitými parametry, přenášející se nějakým prostředím. Putuje-li tato informace akustickým prostředím, jedná se o mechanické kmitání částic. Kmitavý pohyb je dělen podle jeho periodicity. Neperiodickým kmitavým pohybům odpovídá hudební termín hluky. Speciálním případem periodického kmitání je harmonický kmitavý pohyb. Jak z názvu vyplývá, je popsán harmonickou funkcí (sinus, kosinus). Matematicky se jedná o řešení pohybové rovnice pro harmonické kmity, tedy lineární diferenciální rovnice 2. řádu s konstantními koeficienty a nulovou pravou stranou. Řešeními takové rovnice jsou potom harmonické funkce času.

$$x(t) \quad = \quad x_m sin(\omega t - \varphi_0) \tag{1.1}$$

Kde parametr x_m je amplituda funkce, hudebně síla zvuku. Úhlový kmitočet ω udává frekvenci, tedy výšku zvuku a φ_0 je počáteční fáze.

Pokud přenos informace probíhá přes elektrickou soustavu, jedná se o elektrický signál přenášený pomocí střídavých elektrických veličin (napětí, proud).

Prochází-li informace číslicovým systémem, jedná se o signály s diskrétním časem. Modely signálů rozlišujeme na deterministické a stochastické. Stochastické signály se vyznačují nepravidelným chováním, nelze určit jejich okamžitou hodnotu. Proto se často

označují jako tzv. náhodné signály nebo šumy. Druhy šumů se liší spektrální hustotou. Zatímco bílý šum má spektrální hustotu konstantní v celém pásmu kmitočtů, u barevných šumů se spektrální hustota mění v závislosti na kmitočtu. Deterministické signály jsou jednoznačně definované a popsané rovnicemi s konstantními parametry. Patří sem signály harmonické, periodické a kvaziperiodické. Harmonické signály jsou signály jejichž okamžitá hodnota s(t) se mění v závislosti na čase podle sinusové nebo kosinusové funkce:

$$s(t) \quad = \quad C_1 cos(\omega_1 t + \varphi_1) \tag{1.2}$$

Kde C_1 je maximální dosažená hodnota signálu, úhlový kmitočet je ω_1 = 2π/T a φ_1 udává posunutí signálu v čase. Periodické signály jsou složeny z harmonických složek, jejichž úhlové kmitočty jsou celistvými násobky úhlového kmitočtu základní harmonické složky. Úhlový kmitočet k-té harmonické složky:

$$\omega_k \quad = \quad k\omega_1 \tag{1.3}$$

Kde k $>$ 1. Amplituda dílčích harmonických složek je dána spektrem amplitud (modulů). Počáteční fáze dílčích harmonických složek je dána spektrem fází. Zobrazení signálu v kmitočtové oblasti je označováno jako spektrum signálu. Rozklad periodického signálu na jeho harmonické složky provádíme pomocí Fourierovy transformace.[12]

1.2 Parametry hudební nahrávky

Vizualizace zvuku vznikají za různým účelem, v některých případech je k tomuto tématu nutné přistupovat jak z hlediska uměleckého, tak z hlediska technického. Cílem mojí práce je vytvořit syntetický objekt reagující na parametry zvuku. Tudíž je to případ, kdy se nejedná o detekci parametrů za účelem měření a zjištění přesné hodnoty. Jsou tedy nutné oba výše zmíněné přístupy. V případě, že hodnotíme parmetry nahrávky samostatně rovněž narazíme na dva možné přístupy. Nahrávky disponují několika technickými a několika uměleckými parametry. Některé z nich jsou vzájemně ekvivalentní.

Dynamika

Jedná se o průběh hlasitosti zvuku v čase. Je zvykem dynamiku nahrávky zapisovat pomocí symbolů přímo do notového zápisu. Skladatelé často používají italský termín *piano* pro úsek, který se bude interpretovat slaběji a *forte* pro úsek, kde bude hráč hrát silněji. Termíny pro celou škálu popisující dynamickou gradaci od nejslaběji hraných zvuků: *pianissimo*, *piano*, *mezzopiano*, *mezzoforte*, *forte*, *fortissimo*. Postupný vývoj dynamiky popisují termíny *crescendo* (postupné zesilování zvuku) a *decresendo* (postupné zeslabení).

Tyto termíny slouží jako informace pro interprety, kteří na jejich základě subjektivně upravují dynamiku produkovaného zvuku. Práce z dynamikou jako taková se tedy liší v závislosti na interpretovi, jeho nástroji a na hraném žánru. Na žánru je také závislý dynamický rozsah, což je rozdíl hlasitostí mezi nejtišším a nejhlasitějším bodem nahrávky. Zatímco v klasické hudbě nebo v jazzu je dynamický rozsah velký (desítky dB), v nahrávkách elektronické taneční hudby nebo metalu je dynamický rozdíl většinou minimální. Dynamice odpovídá veličina U_{RMS}, tedy hodnota napětí zvukového signálu.

Tempo

Tempo v hudbě určuje rychlost, s jakou po sobě následují časové jednotky, taktové doby. Je to poměr hudebního času k času astronomickému. Počet taktových dob za jednu sekundu bývá běžně v rozsahu 40 - 208. V moderní hudbě se však horní hranice může pohybovat výše. Od doby baroka se tempo udává slovním výrazem, nejčastěji mezinárodně platným italským tempovým označením. Dnes je obvyklé dělit tempo do pěti skupin.

- 1. Velmi pomalá tempa (40: grave až 54: adagio)
- 2. Pozvolná tempa (60: larghetto až 80: comodo)
- 3. Mírná tempa (84: maestoso a 88: moderato)
- 4. Rychlá tempa (106: allegretto až 144: allegro assai)
- 5. Velmi rychlá tempa (152: allegro vivace až 208: prestissimo)

Kvůli upřesnění tempa se mohou slovní výrazy kombinovat nebo doplňovat. Pro mnoho žánrů jsou také typické dílčí tempové odchylky. Může se jednat o tzv. agogiku, kdy interpret dle svého citu a vkusu provede nepatrné tempové změny. Agogika nebývá zaznačena v notovém zápisu. Jiným případem jsou náhlé nebo postupné změny tempa nebo změna taktu, ty se do notového zápisu značí. Kromě slovního značení je možné použít metronomické označení, které udává počet taktových dob za jednu minutu (také značeno jako BPM = beats per minute).[7]

Metrum

Jedná se o systém, který člení časový průběh skladby a sled tónů na menší celky, ve kterých se střídají přízvučné a nepřízvučné stejně dlouhé doby. Podle střídání přízvučých a nepřízvučných dob rozlišujeme metrum na dvoudobé, třídobé, čtyřdobé atd. Dle pravidelnosti ve střídání přízvučných a nepřízvučných dob rozlišujeme metrum pravidelné a nepravidelné. V těsné souvislosti s metrem je rytmus. Jedná se o kontrast použitých tónových délek.[7]

Barva (témbr)

Je určena zejména spektrálním složením tónu. Závisí na počtu současně znějících harmonických složek, na velikosti jejich amplitud a na šumech a šelestech produkovaných při hře.

Výška tónu a harmonie

Výška tónu závisí na počtu kmitů, které udělá oscilátor nástroje za minutu, tj. na frekvenci. Z celého slyšitelného frekvenčního rozsahu, který obecně sahá od 16 Hz do 20 kHz využívá hudba pouze část, a to frekvence od 16,35 Hz (odpovídá C_2, C subkontra nebo-li C0, v případě že komorní a1 = 440Hz) do 8 372 Hz (odpovídá c6, šestičárkované c nebo-li C9, v případě že komorní a1 = 440Hz). V tomto rozsahu jsou tóny pro lidské ucho snadno rozlišitelné. Tónová oblast se dělí na tři základní skupiny:

- 1. tóny hluboké: oblast subbasová, basová, barytonová, zahrnuje oktávy subkontra, kontra, velkou (16,35Hz - 123,47Hz)
- 2. tóny středně vysoké: oblast tenorová, altová a mezzosopránová, zahrnuje oktávy malou, jednočárkovanou a část dvoučárkované (130,81 Hz - 784 Hz)
- 3. tóny vysoké: oblast sopránová a oblast nejvyšších tónů, zahrnuje část oktávy dvoučárkované a vyšší (od 830,61 Hz).

V hudební teorii jsou tyto tóny uspořádány do systému - tzv. tónové soustavy. Tónovou soustavu je možné rozdělit na půltóny, popř. čtvrtóny (vychází z temperovaného ladění) nebo na větší celky - oktávy. Pro oktávy platí, že frekvence oktávových tónů je vždy 2 : 1. Základem naší tónové soustavy je 7 základních (celých) tónů v tónové řadě: c, d, e, f, g, a, h (b). Od těchto tónů vznikají snížením a zvýšením tóny odvozené. Pro hudební skladby, popř. nahrávky, je také typické určité předznamenání v podobě křížků a béček (trvale snížených a zvýšených tónů), které spolu s prvním a posledním tónem skladby určují tóninu, ve které se skladba odehrává. Tónina je definována jako soustava tónů a vztahů mezi nimi. V některých případech je použita tzv. modulace a skladba tak pro určitou část přechází do jiné tóniny. Je však zvykem, aby se začátek skladby hrál ve stejné tónině jako její konec. Z dané tóniny vyplývají použité akordy (souzvuky více tónů). Soustava akordů ve skladbě tvoří její harmonii.[7] S těmito parametry nahrávky také souvisí frekvenční rozsah. Ten může být závislý na použitých hudebních nástrojích, na jejich instrumentaci, na nahrávacím řetězci a na použité ekvalizaci během nahrávání či během mixu.

Počet kanálů, panorama

Standardně dnes většina hudebních nahrávek vzniká ve stereu, je však možné vytvořit mono nahrávku či nahrávku pro surroundové systémy. Mono (monaurální) nahrávka obsahuje pouze jeden kanál. Jednotlivé zdroje zvuku jsou umístěny ve středu pomyslného

prostoru nahrávky. Mono mixy jsou proto náchylnější k tomu, že se nástroje mohou maskovat, a to buď hlasitosně nebo barevně. Intenzivně se mono nahrávky používaly do 50. let 20. století.[19]

Stereo nahrávky poskytují možnost rozdělit kanály při mixu prostorově mezi pravou a levou stranu. Kromě celkového hudebního vjemu poskytují posluchači informaci o směru, ze kterého zdroj zvuku přichází. V současné době se jedná o nejběžnější formu výroby nahrávek.

Prostorový zvuk označovaný jako surround sound nebo jen surround se do komerční sféry dostal v roce 1969. V této době se však ještě jednalo o prostorový zvuk založený na kvadrofonii. Čtyři reproduktory byly umístěny do rohu pomyslného čtverce okolo posluchače. Časem se však ukazuje, že toto rozmístění je matoucí. Z psychoakustického hlediska platí, že posluchač vnímá nejlépe zdroj zvuku umístěný v rovině přímo před ním. V kvadrofonickém uspořádání tento zdroj zvuku chyběl. Proto se pojmem surround sound začalo označovat uspořádání s šesti či více kanály známé jako 5.1, 6.1 a další. Ve verzi 5.1 se jedná o pět kanálů přenášejících audio a jeden kanál přenášející basovou a efektovou složku.[19]

Vzorkovací frekvence

Základním předpokladem pro správné vzorkování je dodržení Nyquistova teorému, který říká, že vzorkovací frekvence musí být minimálně dvakrát vyšší než maximální frekvence vzorkovaného signálu.

$$f_{vz} \geq 2f_{max} \tag{1.4}$$

Při užití nevhodné vzorkovací frekvence dochází k aliasingu (překrytí frekvenčního spektra), což má za důsledek znehodnocení vzorkovaného signálu. Vzhledem k maximální slyšitelné frekvenci (cca 20 kHz) musí být f_{vz} minimálně 40 kHz. Nejnižší běžně používaná vzorkovací frekvence je 44,1kHz. Tato hodnota byla zvolena kvůli možnosti použití audia na video zařízeních, které mají snímkovací frekvenci 25 fps (frames per second). Běžně se pro audio signál vzorkuje také ve 48 kHz nebo ve vyšších frekvencích.

Počet bitů

Jedná se o počet bitů, kterými je popsán jeden vzorek. Pro CD kvalitu se používá typicky 16 bitů na vzorek neboli 2^{16} úrovní mezi maximální hlasitostí nahrávky a tichem, respektive šumem. Běžně se užívají také vyšší počty bitů na jeden vzorek.

Formát a způsob uložení

Jedná se o formát uložení dat v souboru. Způsoby uložení se dají rozdělit podle toho zda nahrávku komprimují nebo ne. K nekomprimovanému uložení se používá kódování

PCM (pulzně kódová modulace). PCM je typický pro zápis na CD či DVD, zaznamenaná nahrávka je téměř shodná s původní analogovou. Komprese bývá dvojího typu: v časové oblasti a v kmitočtové oblasti. Datové kontejnery (formáty) je pak možné dělit na standardizované a proprietární (užívané jedním výrobcem).

2 HISTORIE VIZUALIZACE ZVUKU

2.1 Umělecké zobrazení parametrů zvuku

Z oblasti umění lze za vizualizaci zvuku považovat některé formy tance. Americká společnost Merriam-Webster, která se zabývá tvorbou slovníků, definuje pojem vizualizace zvuku hned dvakrát. Za prvé, vizualizací zvuku je balet nebo forma moderního tance, která plně vychází z hudebního doprovodu. Za druhé, vizualizací zvuku je tanec, který představuje přímý překlad hudby do pohybu. Z těchto tvrzení vyplývá, že tanec je historicky nejstarší formou vizualizace zvuku v reálném čase. Tanec divákovi obvykle zprostředkovává vizualizaci tempa, nálady či frekvenčního složení doprovodné nahrávky.

2.2 Technické zobrazení parametrů zvuku

V technické oblasti nesahají pokusy o zobrazení zvuku, tak daleko do lidské historie jako v případě umění. První pokusy o vizualizaci zvuku se datují již do 2. poloviny 19. století, tedy paralelně s prvními pokusy o zvukový záznam. Na přelomu 19. a 20. století docházelo také k rozvoji měřící techniky. Na poli zobrazování parametrů zvuku se jednalo především o osciloskopy a později také o hlukoměry.

Počátky zobrazování parametrů zvuku

Speciálním případem techniky zobrazující průběh zvukového signálu byl fonoautograf. Jednalo se prakticky o mechanického předchůdce analogových osciloskopů. První zařízení tohoto typu sestrojil v roce 1857 E. L. Scott de Martinville. Na vstupu přístroje bylo zvukové vlnění, na výstupu pak nakreslená zvuková stopa. Převodu se dosahovalo použitím jehly připevněné k membráně. Jehla se na druhém konci volně dotýkala skleněného válce pokrytého sazemi. Zvukové vlnění se pak na membráně měnilo v kinetickou energii, která zapříčinila pohyb jehly v sazích a vykreslila tak průběh zvukového vlnění. V roce 1909 toto zařízení zdokonalil americký fyzik D. C. Miller tím, že jehlu nahradil optickou čočkou a zrcadlem, které bylo připevněno k membráně a reagovalo tak na její pohyb. Světlo bylo nasměrováno skrz čočku na pohyblivé zrcadlo, které jej odráželo na fotografický film poháněný motorem. Zařízení tak bylo schopno vykreslit na film poměrně přesně odlišné průběhy zvuku produkovaného různými akustickými nástroji.[21]

Dalším historickým přístrojem, který by se dal považovat za mechanického předchůdce osciloskopu byl Koenigův manometrický plameník. Stejně jako fonoautograf obsahoval membránu, která přenášela zkoumané kmity. Za membránou byla kapsle naplněná plynem. Skrz něj se kmity přenášely až na plamen. Pokud plamen neindikoval žádné

kmity, hořel klidně, svítivým plamenem. Při indikaci kmitů se zřetelně měnila jeho svítivost a délka. Toto zařízení se používalo především k vyhledávání uzlů stojatého vlnění v uzavřeném akustickém poli. Kromě Koenigova manometrického plameníku existovalo více různých přístrojů pracujících na podobném principu. Jak plameník, tak fonoautograf - obě tato zařízení stála na počátku výzkumu A. G. Bella, ze kterého nakonec vzešel první mikrofon a první telefon.[13]

Měřící technika

Počátky vývoje analogových osciloskopů se datují do poloviny 19. století, kdy W. Crookes vynalezl zobrazovací techniku CRT (Cathode Ray Tube). První osciloskop, obsahující obrazovku CRT, sestrojil v roce 1897 německý fyzik Karl Braun. První osciloskopy však nebyly vybaveny zesilovačem a kvůli malému napětí signálu byl zobrazený průběh extrémě malý. Běžnou praxí bylo, průběh fotografovat a fotografii následně zvětšit, což však znamenalo, že mohla být zaznamenána pouze určitá, krátká část signálu. V roce 1922 začala firma Western Electric s komerční výrobou osciloskopů. O deset let později již byl do osciloskopů přidán generátor pilového průběhu sloužící jako časová základna, což umožnilo kontinuální zobrazení průběhu signálu. Během 2. světové války pracují vědci najmutí armádou na radarové technice. Rozvoj této techniky napomohl i dalšímu zdokonalení osciloskopů - zpřesnilo se měření amplitudy signálu a dále se zlepšuje také vestavěný zesilovač. Do roku 1970 se z osciloskopů stává nejpoužívanější elektronická měřící technika, a to především díky firmám Cossor a Tektronix. Dále byly do osciloskopů přidány také integrované obvody, digitální vstupy a zobrazování průběhů a parametrů signálu bylo barevně odlišitelné.[8]

Dalším stupněm ve vývoji osciloskopů byl přechod na digitální zpracování dat. U digitálních osciloskopů je v signálové cestě po zesilovači zařazen A/D převodník. Analogový signál je navzorkován, vzorky se ukládají do paměti, jsou zpracovány a zobrazeny. Odebírání vzorků je řízeno časovou základnou. Existují různé varianty digiálních osciloskopů: od nejběžnějších DSO (Digital Storage Oscilloscope) po přístroje typu MSO (Mixed Signal Oscilloscope), kde je v jednom zařízení kombinován osciloskop s logickým analyzátorem.[20]

V posledních letech se rozvíjí také softwarové měření parametrů zvuku. Jelikož se ve většině případů jedná o kombinaci programu a zvukové karty, je toto řešení po ekonomické stránce pro koncového uživatele zdaleka nejdostupnější. Existuje nepřeberné množství jednoduchých freeware programů, které fungují jako osciloskop, generátor a spektrální analyzátor v jednom. Nabízenými funkcemi se však značně liší. Tyto programy stojí buď samostatně nebo se ve formě pluginů a toolboxů připojují k jiným programům. Příkladem mohou být například toolboxy Wavelet nebo Wavelab v programu Matlab, které dokáží vyhodnotit a zobrazit parametry audio signálu.

Digital Audio Workstation

Jedná se o systém několika komponent řízený počítačem. Počítačem lze ovládat jak proces nahrávání, přehrávání, editaci, tak zpracování. Celý systém také podporuje MIDI a lze synchronizovat přes časový kód nahrávky. První generace DAW se skládala z jednoho nebo více počítačů, které zabezpečovaly veškeré digitální zpracování v 16, 20 nebo 24 bitech. Dnes jsou pod pojmem DAW označovány komplexní systémy, které kompletně řeší zpracování audia a jsou dále rozšiřitelné o pluginy. Vývoj těchto systémů začal již v roce 1983, tedy v době, kdy už je veřejně dostupný osobní počítač Atari ST s grafickým i MIDI rozhraním. O šest let později, Steinberg vydává první verzi svého DAW - Cubase a v roce 1991 se objevuje také první verze konkurenčních Pro Tools společnosti Digidesign.[22]

2.3 Technika pro zábavní průmysl

S rychlým rozvojem počítačů v 2. polovině 20. století dochází v určité oblasti kultury k propojení techniky a umění. Dalo by se říci, že úkolem techniky je podpořit umělecké sdělení a pozitivně ovlivnit vjem diváka. Pódiová technika (světla apod.), která do té doby stála odděleně, se synchronizuje se zvukem a je závislá na jeho parametrech.

Osvětlení

Jak je již zmíněno výše, práce se světlem se při živých aplikacích se zvukem dlouhou dobu nespojovala. Ačkoliv historie osvětlování pódií sahá až do roku 1580, kdy bylo v Itálii při divadelní hře poprvé použito nasvícení pódia pomocí svícnů, osvětlení zůstávalo velmi dlouho pouze pasivním scénickým prvkem. V 2. polovině 19. století se začalo používat osvětlení s barevnými filtry, scénická technika tedy mohla lépe podkreslit náladu uměleckého představení. V polovině 20. století byla společností Century Lighting představena první ovládací konzole obsahující deset presetů. Velká změna přišla v roce 1975, kdy byl na Broadwayi poprvé použit elektronický systém osvětlení s tzv. Computer-Assisted Memory konzolí. Poté, co byla každá scéna vytvořena manuálně, ji bylo možné uložit do paměti a během představení ji znovu vyvolat. V 80. letech 20. století byly zavedeny protokoly MIDI a DMX512, což vedlo v osvětlovací technice k velkému posunu. Jednotlivé scény lze naprogramovat, spojit a synchronizovat se zvukem. Osvětelní tak může při živých aplikacích efektně podpořit nejen náladu představení, ale také rytmus či intenzitu produkovaného zvuku.

Barevné hudební nástroje

Speciálním typem vizualizace zvuku pomocí barevného světla jsou barevné hudební nástroje. V tomto případě vychází idea spojení barev a zvuku z psychologického jevu, který

se nazývá synestezie. Jedná se o jev při kterém dochází k samovolnému sdružení vjemů více lidských smyslů.[18] Barevné hudební nástroje produkují primárně zvuk a podle různých teorií jsou k jednotlivým zvukům přiřazeny barvy. Primárně má tak divák vjem sluchový a ten je doprovázen vjemem zrakovým. Do skupiny barevných hudebních nástrojů (color organs) se řadí jak hudební nástroje produkující světla i zvuky v reálném čase, tak i ty, které světelnou složku přidávají k hudbě z playbacku. Jev sdružování lidských vjemů je znám už ze starověkého Řecka. Pojmenován byl však až v období renesnce (17. a 18. století), kdy se vědci i umělci začali zabývat možností vytvářet nástroje produkující barevnou hudbu.[2] Mechanické typy těchto hudebních nástrojů vznikaly již v průběhu 18. století (*clavecin oculaire*, *piano optophonique*). K velkému rozvoji v této oblasti vizualizace došlo až na konci 19. století, a to především díky vzrůstající popularitě abstraktního umění.[3] Mechanické nástroje vystřídaly začátkem 20. století nástroje elektrické. Jedním z nich byl clavilux Thomase Wilfreda. Jednalo se o zařízení, které prostřednictvím čoček směrovalo paprsky barevného světla na velké plátno. Ovládání bylo zajištěno instalací tří manuálů (klaviatur), každá klávesa mohla být nastavena do jedné ze sta pozic. Wilfred navrhl také malé, typově podobné přístroje pro domácí použití. Éru těchto nástrojů víceméně ukončil rozvoj osvětlovací techniky a speciálních efektů.[23]

Speciální efekty (SFX, SPFX)

Skupina speciálních efektů se během historie velmi rozrostla. Jejich použití v souvislosti s hudební produkcí je nejznámější z živých aplikací (koncerty, divadelní vystoupení, taneční vystoupení, světelná show apod.). Podstatou jejich použití je podpořit a oživit dění. V dnešní době prakticky neexistují žádná umělecká pravidla pro používání této skupiny efektů. Omezením tak jsou pouze bezpečnostní směrnice, technické řešení a ekonomická stránka věci. Vývoj speciálních efektů začal ve filmovém průmyslu, a to už ke konci 19. století. V hudebním průmyslu se začaly efekty objevovat až s příchodem hard rocku, tedy v 70. letech. Experimentoval s nimi Ron Volz, který na počátku 70. let pracoval pro amerického hudebníka Alice Coopera. Volz je považován za prvního odborníka přes speciální efekty vůbec. Z počátku se jednalo pouze o používání dýmovnic na tvorbu kouře. Šlo o manuálně řízené efekty, které sloužily k oživení a podpoření nálady dané hudební produkce. Později se začala místo dýmovnic používat pyrotechnika, velkoplošné obrazovky, laserové instalace a hydraulika. Řízení některých těchto efektů přešlo na počítače. Programy na řízení pyrotechniky fungují na podobném principu jako programy zajišťující funkci osvětlení. Vytváří se jednotlivé scény, které se skládají za sebe. Odpalování jednotlivých nálož se řídí časovým kódem nahrávky. Příkladem typického prostředí může být program FireOne. Co se týče laserů, poprvé se objevily v roce 1975 na živé produkci britské rockové skupiny Led Zeppelin. Další zdokonalení efektové techniky přišlo s projekty Pink Floyd, Kiss, Michael Jackson a Madonna, ta v roce 1990 na své koncertní tour

poprvé použila velkoplošné obrazovky. V poslední době se na pódiích začínají objevovat také obrazovky IMAX, při své multimediální show nazvané Delirium je použila kanadská zábavní společnost Cirque du Soleil.[17]

Softwarové vizualizace

Na počátku softwarové vizualizace stály analogové video syntezátory. Jejich érou byla 70. léta. V principu šlo o zpracování analogového signálu z kamery nebo o tvorbu signálu nového. Výhodou video syntezátoru byla práce v reálném čase. Nevýhody však u tohoto přístroje značně převažovaly: složitá konstrukce, velikost a vysoká cena. To vedlo k tomu, že tyto přístroje byly v hudebním světě spíše raritou a také to byl důvod, proč se s nimi živě vystupovat prakticky nedalo. Z analogově-digitálních hybridních zařízení se více prosadilo GROOVE (Generated Realtime Output Operations on Voltage-controlled Equipment). Jednalo se o malý počítač, který řídil analogové zařízení s video výstupem. Měl několik různých vstupů: posuvné, otočné potenciometry, joysticky a klaviaturu. Program byl tvořen v jazyce FORTRAN IV a assembleru DAP 24. Dalším známým počinem byl VAMPIRE (Video and Music Program for Interactive Realtime Exploration/Experimentation) vyvinutý Laurií Spiegel mezi lety 1974 a 1979. V principu se jednalo o program, který na základě časového kódu řídil video i audio složku. Prvním digitálním zařízením byl vizualizér Atari Video Music, představený v roce 1978. Nedlouho poté Atari Inc. vyvinulo herní konzoli Atari 2600, která byla schopna na základě audio signálu vytvořit obraz na televizním monitoru. Prakticky veškerá audiovizuální technika je v dnešní době digitální. Výroba vybavení pro VJs (video jockeys) stále roste a audiovizuální tvorba je v porovnání s „pouze“ hudební tvorbou DJs zdrojem vyšších příjmů, tudíž je i z ekonomického hlediska podstatnější. Základním vybavením pro VJs jsou softwary, a to buď pouze video generátory jako ArKaos, Motion Dive Tokyo nebo integrovaná prostředí pro audio i video, například Max nebo PureData s GEM (Graphic Enviroment for Multimedia) o kterých bude pojednáno později. Běžně se také zavádí podpora MIDI protokolu pro komunikaci s programy nebo pluginy určenými primárně pro audio, naopak v hudebně-editačních programech (Pro Tools, Logic) je zavedena podpora videa. Jinou možností je využít prostředí speciálního programovacího jazyka Scheme (např. prostředí Impromptu).[4]

Mimo živou produkci vzniklo během historie několik významných audiovizualizérů za různým účelem. V roce 1985 S. Malinowski představil svůj projekt *Animation Music Machine*. Cílem bylo zobrazení notového zápisu tak, aby byl srozumitelný i lidem bez předchozího hudebního vzdělání. V první verzi programu bylo nutné přidělit každému taktu určitou hodnotu. Pozdější verze již k synchronizaci hudby a animace používaly protokol MIDI. Animace je tvořena z řad různě barevných bloků, které se posouvají v tempu hudby. Jeden blok odpovídá jedné notě. V případě souzvuku jsou bloky poskládány pod

sebou. Délka bloku se řídí délkou noty, barva bloku odlišuje různé nástroje nebo různé rejstříky. Momentálně přehrávaný blok je v řadě zvýrazněn. Umístění bloků na pozadí se řídí výškou (frekvencí) na ose y a časem na ose x. Díky použitému grafickému zobrazení jsou divákovi vizuálně zprostředkovány také vztahy mezi tóny a intervaly, které vycházejí z harmonie. Jedná se o tzv. analytickou animaci. Grafické uspořádání použité v *Animation Music Machine* je v obměnách používáno v programech i dnes. Výuková hra Synthesia, jejímž primárním účelem je pomocí grafického zobrazení not zjednodušit výuku hry na klávesové hudební nástroje. Synchronizace animace a zvuku je závislá na protokolu MIDI, grafická stránka je zpracována v OpenGL. Dalším typově podobným programem je videohra Guitar Hero, kde se jedná o animovanou kytarovou tabulaturu regující na vstup z ovladače, kterým je nahrazena reálná elektrická kytara.[1]

Významně do oblasti vizualizace zvuku zasáhla také americká společnost Animusic, jejímž zakladatelem je umělec Wayne Lytle. V principu se jedná o 3D počítačovou animaci synchronizovanou se zvukem pomocí MIDI. Použité animace zobrazují unikátní hudební nástroje (např. jeden hudební nástroj složený z více známých drnkacích nástrojů), ze kterých je zvuk vyluzován mechanicky pomocí elastických předmětů či trsátek nebo je nástroj ovládán roboticky. První projekt společnosti vznikl již v roce 1990. Jednalo se především o kód v jazyce C, kam byla importována MIDI data z externího sekvenceru. Později byly práce vyvíjeny v softwaru 3ds Max (animace, vykreslování) a MIDImotion (pohyby hudebních nástrojů). Později společnost vyvinula vlastní software nazvaný Animusic|studio, jako sekvencery byly použity programy Cubase SX a Nuendo.[14]

Vizuální podkreslení přehrávaného zvuku poskytuje i většina programů určených pro manipulaci s audio soubory v počítači. Tyto produkty jsou určeny pro širokou veřejnost, to znamená, že musí být schopny přehrát a reagovat na velkou škálu různě kvalitních nahrávek různých hudebních stylů. Z tohoto důvodu bývají softwarové přehrávače značně univerzální. Vizuální část programu reaguje nejčastěji na dynamické změny v nahrávce (dynamické vrcholy, peaky, beaty), na frekvenční rozsah či alespoň na dobu trvání dané nahrávky.

Hologramy

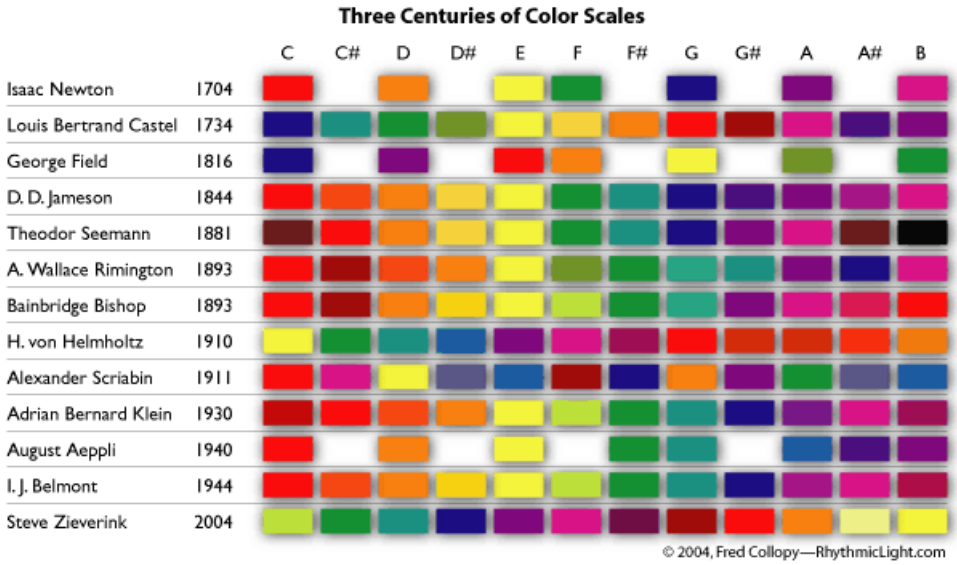
Holografie se zabývá tvořením specifických fotografických obrazů bez použití čoček. Předmět zobrazený touto metodou se nazývá hologram. V principu se jedná o dva svazky koherentního světla, které spolu interferují a vytváří tak interferenční proužky. Vhodným nasvícením těchto proužků pak vzniká trojrozměrný obraz. Vizualizace zvuku pomocí hologramu se objevila teprve poměrně nedávno a typická je především pro živá hudební vystoupení. První zmínky o využívání hologramů sahají až do roku 1862. V té době byl jednoduchý holografický trik využíván k zobrazení duchů v divadelních hrách (např. hra Charlese Dickense The Haunted Man, režisér John Pepper). Zjednodušený princip holo-

grafie byl již však poprvé popsán kolem roku 1590 v dokumentu Magica Naturalis od B. G. D. Porta. V moderní živé produkci se využívají pohyblivé hologramy synchronizované s hudbou. V roce 2006 byla tato technologie použita na předávání cen Grammy. Jednalo se o vystoupení zpěvačky Madonny, jejímž doprovodem byla britská virtuální skupina Gorillaz v holografické podobě. V roce 2007 spojením společností Yamaha, Sega, Sony Music a Crypton Future Media vznikl projekt Hatsune Miku. Jedná se o zpívajícího avatara doprovázeného živou kapelou. Vizuální podoba je řešena sestavou 3D holografických projektorů (portál Inventor-strategies.com uvádí, že běžně bývá použito několik tuctů těchto projektorů). Díky tomu si obraz zachovává stejnou podobu pod všemi úhly. Hudební složka avatara je řešena pomocí syntezátoru Vocaloid (Yamaha). Přestože se jedná o syntetický objekt, koncertuje jak v Japonsku, tak v zahraničí a nezřídka se stává, že tyto koncerty bývají vyprodány.[15]

3 VÝVOJOVÁ PROSTŘEDÍ

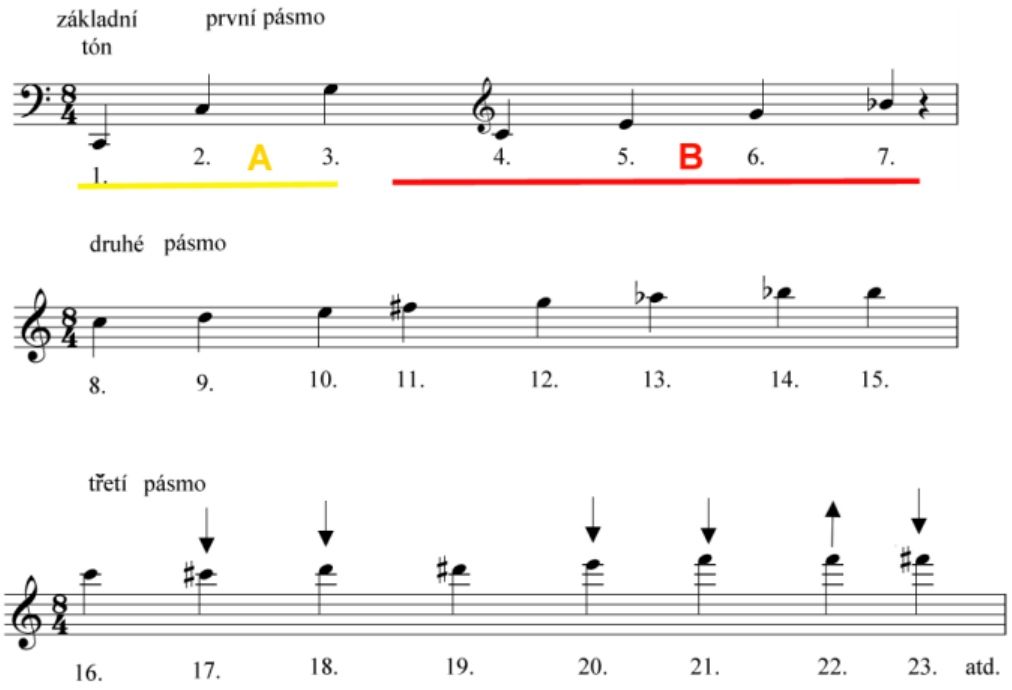
3.1 Propojení zvuku, barev a tvarů

Jak je již zmíněno výše, propojením zvuku a barev se zabývalo mnoho umělců, a to v různých érách kulturních dějin. Nejvíce je toto spojení rozebíráno v souvislosti s jevem synestezie, kdy dochází ke sdužení vjemů více lidských smyslů. Jelikož se však jedná o psychologický jev, je jeho vnímání do jisté míry u různých osob subjektivní. Zkoumáním ideálního propojení barev a hudebních tónů proto vznikl nespočet teorií jak vědeckých, tak uměleckých. (Obr. 3.1) V pramenech často zmiňovanou teorií je teorie Issaca Newtona pocházející z roku 1704. Jeho cílem bylo přiřadit sedm tónů na klaviatuře piana k sedmi barvám duhy (teorii pojmenoval Collopy).



Obr. 3.1: Teorie umělců a vědců přidělující tónům barvy [1]

V případě Newtona byly barvy přiděleny tónům pouze podle tonální výšky. Je však možné se také setkat s přidělením barev podle akustické barvy tónů. Tato barva je závislá na počtu a velikosti amplitud vyšších harmonických složek daného tónu. (Obr. 3.2) V zásadě platí, že první pásmo harmonických tónů, generuje zvuk s barvou hebkou, dutou a neprůraznou. Druhé pásmo naopak generuje zvuk se sytou, sonorní a průraznou barvou. V případě třetího pásma je pak zvuk silně průrazný, ostrý a zvonící. Také platí, že zvuky s převahou sudých harmonických složek jsou měkké a naopak ty, s převahou lichých jsou ostré.[9]



Obr. 3.2: Harmonická tónová řada [9]

Přiřazení barev, které by bralo v úvahu akustickou barvu tónu provedl ruský malíř a umělecký teoretik W. Kandinsky.

- Žlutá: barva vyjadřující sílu, ale zároveň inklinující k tónům s vyšší výškou. Z hudebních zvuků nejvíce podobná trubce hrající vysoko laděné fanfáry.
- Oranžová: barva znějící podobně jako zvon se střední výškou či bohatě znějící kontraalt.
- Červená: zvučná barva připomínající tubu, která hraje fanfáru - to znamená houževnatým, vytrvalým a energickým tónem.
- Purpurová: označována jako barva mládí, svěžesti a ryzího štěstí. Je srovnatelná s vysokým, čistým a zpěvným zvukem houslí nebo malých zvonů.
- Fialová: odpovídá hlubším tónům anglického rohu nebo šalmaje. Nejtemnější odstíny pak hlubším tónům fagotu.
- Modrá: je rozdělena na odstíny. Světlé odstíny odpovídají zvuku flétny, střední odstíny zvuku cella, temnější odstíny kontrabasu a nejtemnější odstíny odpovídají hlubokým polohám varhan.
- Zelená: je symbolem klidu, odpovídá proto dlouhým, meditativním tónům houslí.[6]

Přímo Kandinského barevné asociace nebo její verze byly přejaty mnoha dalšími umělci. Ruský skladatel Alexander Scriabin použil barevné asociace pro některá svá díla, například *Prometheus: The Poem of Fire*. Toto dílo by určeno pro symfonický orchestr, klavír,

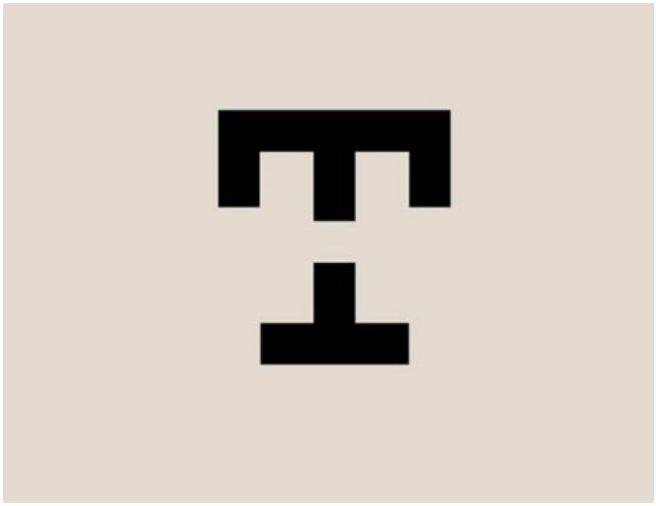
sbor a barevný hudební nástroj zvaný *clavier à lumières*. Sám Scriabin sestavil systém ve kterém přiděloval barvy jednotlivým tóninám.[11]

V oblasti vizualizace zvuku existuje několik dalších doporučení. Jelikož se zároveň jedná o oblast uměleckou, tato doporučení nefungují jako pravidlo, užívají se však ve většině případů a divák je na ně zvyklý.

- Sytost použitých barev: doporučuje se, aby byla závislá na amplitudě zvukového signálu. Syté a zářivé barvy by měly znázorňovat hlasité zvuky. Naopak pro zvuky tišší je vhodnější volit tlumené odstíny s vyšším podílem černé nebo bílé barvy.
- Odstín: světlé odstíny si člověk běžně spojuje s tóny na vyšších frekvencích. Ty temné naopak se zvučnými hlubokými tóny.[6]
- Velikost: pokud je ve vizualizaci použitý tvar či objekt, doporučuje se, aby se jeho velikost přizpůsobila intenzitě produkovaného zvuku. Čím větší intenzita, tím větší objekt. Větší objekty často také znázorňují tóny v hlubších polohách, které člověk vnímá jako masivnější.[10]
- Ostrost: pro hudební kompozice v rychlém tempu se doporučuje volit vizualizaci, která bude obsahovat ostré tvary.

3.2 Chápání prostoru ve vztahu ke zvuku

Volba a využití prostoru může značně ovlivnit výsledný vzhled vizualizace. Dobře zvolený a rozvržený prostor výsledný vjem obrazu a zvuku více sjednotí. V opačném případě působí jako rozdělující prvek. Důležité při tom je, aby se obraz divákovi jevil jako podmanivý.



Obr. 3.3: Pozitivní a negativní prostor [5]

V teorii pro malíře a jiné umělce pracující s prostorem se prostor dělí na negativní a

pozitivní. Jako základní příklad tohoto rozdělení se používá bílé plátno s černým objektem uprostřed (Obr. 3.3). V tomto případě je černý objekt považován za pozitivní část, bílé pozadí je negativní. Má se za to, že pokud se divák zaměří na pozitivní část, vnímaný prostor se celkově propojí. Vnímá-li negativní část, jasně si uvědomuje hranice jak prostoru samotného, tak hranice pozitivní části v něm.[5]

3.3 Možnosti herních enginů

Herní enginy jsou méně používaným prostředím pro realizaci vizualizace zvuku. Jejich výhodou je nepřeberné množství možností, co se týče zpracování grafické části vizualizace. Je zde také možnost zdroj zvuku v prostoru přesně lokalizovat, pohybovat s ním, či pracovat najednou s více zdroji zvuku. Samotné zpracování vzorků audia je však spíše slabinou. První pokus o realizaci této bakalářské práce proběhl v prostředí Unreal Engine 4, které jsem kombinovala s Visual Studiem 2015. Propojením prostředí získá uživatel možnost manipulovat s vlastním kódem grafických objektů a upravovat jejich vlastnosti. Vlastní kód grafických objektů je psán v C++ a doplněn o speciální funkce enginu. Možnosti zpracování zvukových stop v Unreal Engine jsou však značně omezeny a uzpůsobeny čistě pro práci v rámci vývoje počítačových her.[24] Je zde možnost připojení knihovny FMOD, která například rozšíří škálu použitelných audio formátů, ale pro zpracování vzorků rovněž uzpůsobena není. Mým cílem proto bylo připojit speciální knihovnu jako plugin a zpracování vzorků audia provádět v něm. Realizace vizualizace touto cetou se však ukázala natolik problematická, že bylo nakonec nutné přistoupit na jinou metodu. Nutno podotknout, že samotná kompilace kódu mezi prostředími Unreal Engine a Visual Studio je výpočetně náročná a čas potřebný na vývoj se násobně prodlužuje.

V dalším, momentálně nejrozšířenějším, herním vývojovém prostředí Unity je do určité míry vizualizace zvuku proveditelná. Jsou zde dostupné knihovny s potřebnými algoritmy a funkcemi (funkce oken pro analýzu apod.).

3.4 Tvorba vizualizace v softwarech pro multimédia

Tvorba ve většině multimediálních softwarech probíhá prostřednictvím nějakého programovacího jazyka. V zásadě platí, že co software, to jiný přístup k tvorbě. Zástupcem softwaru, kde se jako prostředek k tvorbě uplatňuje psaný kód, je SuperCollider. Jedná se jak o prostředí, tak o programovací jazyk, který je uzpůsoben k syntéze zvuku v reálném čase, k tvorbě hudby z algoritmů, k interaktivnímu programování či k tzv. live codingu (živé vystoupení, kdy programátor/umělec vytváří zvukové či audiovizuální dílo přímo před diváky).

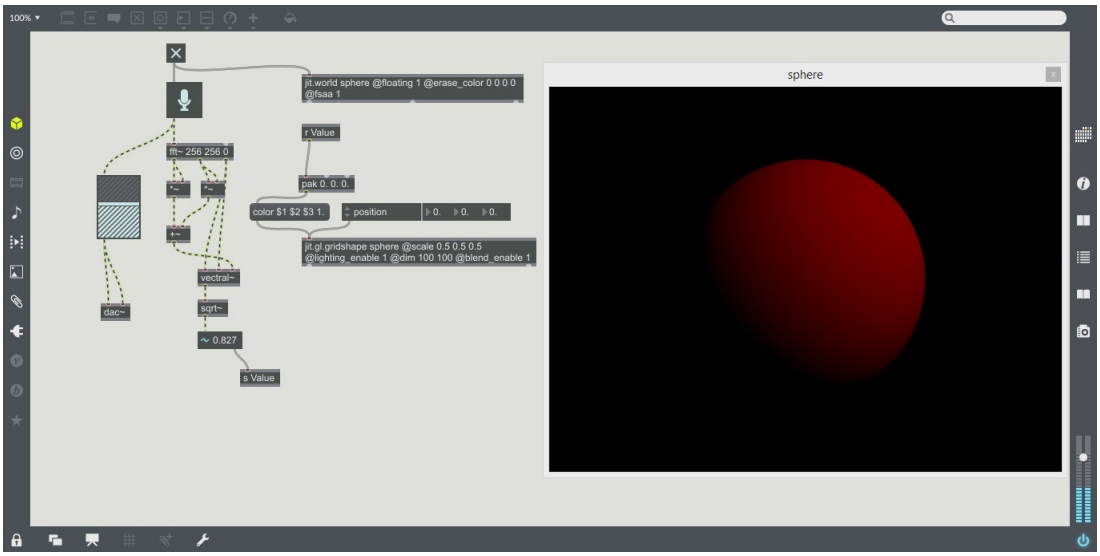
Jinou skupinou jsou již zmíněné softwary Max a Pure Data (Pd), ve kterých uživatel

tvoří prostřednitcvím grafického programovacího jazyku. Co se týče vývoje těchto dvou softwarů, jejich historie se schází v jednom bodě. Autorem prostředí Pd je americký programátor Miller Puckette, který také dříve pracoval na vývoji prostředí Max. Programovací jazyk a vývojové prostředí jsou si proro velmi podobné. V obou případech vytváříme na bílém plátně (*canvas*) soustavu propojených objektů. Max a Pd se liší především licencí. Zatímco Pd jsou šířeny zdarma jako open source, Max je komerční software. Tento rozdíl je příčinou všem ostatním rozdílům v těchto softwarech.

Komunita programátorů věnující se vývoji Pd dnes již postupně zaniká. Komerční Max proto Pd předčí téměř ve všem. V Pd například nelze vyvořit k danému projektu (projekt čili *patch*) žádné uživatelské rozhraní a bez přítomnosti Pd na počítači není možné žádný vytvořený projekt spustit. Max možnost vytvoření uživatelského rozhraní poskytuje a z projektu je možné dostat samostatně stojící aplikaci, která ke spuštění nevyžaduje přítomnost Maxu. Pd jako open source se rovněž potýkají s nejednotností, kterou je možné vidět, jak v samotném zdrojovém kódu jednotlivých funkcí, tak i v případě nápovědy. Každá nově přidaná funkce (nejčastěji ve formě tzv. objektu) by jistě měla mít dostupný patch s nápovědou. U většiny objektů tomu tak sice je, ale ne každý autor k tomu přistupoval stejně zodpovědně. Dalším rozdílem je samotné uživatelské rozhraní, které je v případě Maxu lépe graficky řešené a více intuitivní. Při volbě objektu jsou uživateli automaticky nabídnuty abecedně řazené objekty s informací o jejich funkci. Uživatel tedy nemusí znát všechny názvy objektu přesně a daný objekt může přesto rychle dohledat. V Pd tato funkce neexistuje. „Kamenem úrazu“ je zde však také samotná dokumentace. Při tvorbě v Pd je pro uživatele mnohdy lepší dohledávat informace ve fórech než v samotné dokumentaci. Nasvědčuje tomu i název fóra Pd - *Comments which should be Documents* (Komentáře, které měli být v dokumentaci).[25]

Na další rozdíl narazí uživatel u běžně používaných funkcí. V případě Maxu je evidentní snaha o maximální zjednodušení práce. Pokud uživatel například potřebuje vstup z mikrofonu, Max celý patch zapouzdří a uživateli nabídne předpřipravený blackbox s ikonou mikrofonu. Totéž patí i pro práci s grafikou - Max nabízí zvlášť objeky zajišťující vytvoření grafického okna, rendering, vytvoření geometrického útvaru apod. Je však také možné zvolit objekt s názvem jit.world, který všechny tyto funkce zapouzdří do jednoho objektu. V Pd je nutné zvlášť vytvořit všechny objekty, automaticky se nespustí ani sekce zpracovávající audio signál (sekce DSP). Je zde však, stejně jako v prostředí Max, možnost zapouzdřit tyto objekty ručně do tzv. subpatche - tedy objektu se vstupem a výstupem, který se nachází fyzicky na hlavním plátně a uvnitř skrývá další plátno (*canvas*) se skrytými objekty.

Na obrázku 3.4 je ukázka jednoduché vizualizace audio signálu z mikrofonu. Na základě jednoho parametru vstupního signálu dochází k obarvení objektu. Se zvyšující se hodnotou parametru se zvyšuje hodnota jasu červené barvy. Z obrázku je vidět prostředí Maxu a jeho úspornost na množství nutných objektů.



Obr. 3.4: Příklad jednoduché vizualizace v prostředí Max

3.5 Pure Data

Pro tvorbu vlastní vizualizace jsem si zvolila toto prostředí. Jak jsem již zmínila výše, jedná se o open source prostředí určené pro audiovizuální tvorbu. Není proto těžké dostat se ke zdrojovým kódům a vytořit vlastní objekt. Programovacím jazykem je grafický jazyk Pure Data (Pd). Ten se skládá z objektů (*objects*), zpráv (*messages*), komentářů (*comments*) a speciálních objektů (*GUI boxes*). Objekt i zpráva mohou mít vstup (*inlet*) a výstup (*outlet*), popřípadě více inletů i outletů. Propojením těchto jednotlivých částí uživatel získává patch. Programovací jazyk Pd má omezené možnosti, co se týče užití iterací, větvení kódu nebo podmínek. V podstatě se jedná o jednoduchý princip shodný s principy objektového programování. Objekt má svou metodu, která je volána daty a vrací zpět pouze opět data. Pro zjednodušení programování jsou před uživatelem skryty některé úlohy, např. dealokace smazaných objektů nebo řízení a plánování úloh pro interpreta objektu DSP (Digital Signal Processing object).[26]

Základní pracovní plochou je plátno (někdy nazýváno kanvas), na které pomocí lišty či klávesových zkratek vyvoláváme prázdné objekty či zprávy. Teprve po vepsání názvu objektu prohledají Pd příslušné adresáře a objekt vytvoří. Automaticky dojde k prohledání adresáře, ve kterém jsou uložena Pd a adresáře, ve kterém je uložený patch, pro který objekt vyvoláváme. Další adresáře je možné nastavit ručně v okně Předvolby.

Pd pracují s datovými toky. Data prochází skrz propojení a objekty, kde jsou zpracovávána. Výstup jednoho procesu přichází na vstup dalšího, a tento krok se může mnohokrát zopakovat.

Objekty

Objekty bývají napsány v jazyce C, C++ a nebo v jazyce FORTRAN a jsou dvojího typu. Rozlišujeme objekty (*internals*), které jsou součástí jádra Pd a objekty (*externals*), které jsou uloženy zvlášť a obsahují rozšíření interních funkcí. Kolekce externích objektů je načítána v podobě knihovny, pro Windows, na kterých byla tato bakalářská práce zpracovávána jsou to soubory s příponou .dll - tedy dynamicky připojované knihovny.[25]

Dále rozlišujeme mezi objekty zpracovávající data a objekty zpracovávající signály. Fakt, že se jedná o objekt zpracovávající signál, musíme zohlednit již v názvu - ten musí končit tildou. Datové i signálové objekty mohou mít různý počet inletů i outletů a také různý počet argumentů, které se uvádí do kolonky za název. Název určuje třídu objektu. Argumentem může být číslo i symbol. Počet inletů a outletů může být dán samotnou třídou a nebo je volitelný pomocí argumentů (např. objekty pack/unpack). Pd rozlišují mezi *hot* a *cold* inlety. Jako hot inlet je označován vstup nacházející se na objektu nejvíce vlevo. Skrze hot inlet přijímá objekt data, která spouští vnitřní funkci a výsledkem jsou zpracovaná data na výstupu. Skrze cold inlet je možné aktualizovat stav hodnot uvnitř funkce, ale přímý vliv na výstup nemá.

Propojení

Propojení mezi jednotlivými objekty je řešeno graficky a má podobu propojovacích kabelů (*cords, wires*), které se v patchi mohou libovolně křížit. Úzká, černá propojení značí, že jsou přenášena běžná data. Tlustší, modrá propojení ukazují, že jde o přenost audio signálu. Jednotlivé typy přenášených dat bohužel barevně odlišeny nejsou.[26]

Data

Pokud se nejedná o audio signál jsou data přenášena ve tzv. zprávách. Každá zpráva obsahuje dvě části *selector* a *atoms*. Atomy určují typ přenášené hodnoty:

- *A_FLOAT*: číselná hodnota; číselné hodnoty jsou vždy typu float, a to i tehdy, když je lze zapsat jako integer
- *A_SYMBOL*: symbolická hodnota (string)
- *A_POINTER*: ukazatel.

Každý symbol je uložen ve vyhledávací tabulce. Příkaz *gensym* vyhledá v tabulce příslušný řetězec znaků a vrací adresu nalezeného symbolu.

Selector definuje typ zprávy:

- `bang`: spouštěcí událost, přítomen je pouze selector, bez atomů
- `float`: číselná hodnota, obsahuje selector a jeden atom *A_FLOAT*
- `symbol`: symbolická hodnota, obsahuje selector a jeden atom *A_SYMBOL*
- `pointer`: ukazatel, obsahuje kromě selectoru jeden atom *A_POINTER*

- `list`: obsahuje list tvořený jedním nebo více atomy libovolného typu

V kódu je možné ukázat na adresu symbolu i přímo - `&s_bang`, `&s_float` atd.

Zprávy

Zprávy interpretují uživatelem vložený text jako zprávu, která se odesílá pokaždé, když je kolonka zprávy aktivována - to může udělat sám uživatel myší nebo lze připojit jiný objekt, který bude zprávu aktivovat (bang, loadbang apod.).

GUI boxes

Jedná se o specializované objekty, může se jednat o čísla, přepínače, posuvníky atd. Hodnoty těchto specializovaných objektů se mohou měnit v reálném čase, a to buď na základě zaslaných zpráv z jiných objektů a nebo může jejich hodnotu měnit uživatel myší.[25]

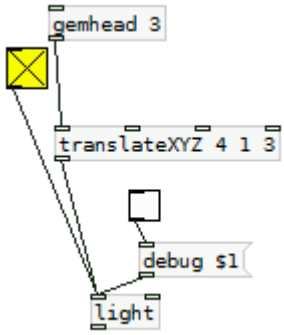
4 VIZUALIZACE ZVUKU V PD

Mým cílem je vytvořit objekt reagující na parametry nahrávky, a to v prostředí Pure Data. Popisu samotné tvorby se budu věnovat ve čtyřech bodech: parametr - amplituda, parametr - barva, parametr - tempo a obrazová část.

4.1 Obrazová část

Grafické prostředí je v Pd řešeno prostřednictvím setu knihoven GEM (Graphics Environment for Multimedia), které jsou uzpůsobeny pro audio-vizuální kompozice v reálném čase. Jedná se o zprostředkování OpenGL grafiky pro Pd. Objekty obsahují funkce zajišťující vykreslování 2D a 3D objektů, světlo, mapování textury, zpracování obrazu a pohyby kamery.

Na začátku řetězce tvořící obrazovou část je objekt *gemwin*, který spravuje výstupní okno GEM a umožňuje uživateli měnit jeho parametry. Přijímá zprávy předcházející tvorbě okna (např. *dimen* stanoví rozměry okna), určené k tvorbě okna (*create* a *destroy*) a zprávy následující po vytvoření okna (např. *lighting* spouští a vypíná rendering světel). Gemwin má jeden argument, a to fps (frames per second, počet snímků za sekundu), který je standardně nastaven na hodnotu 20. Aby došlo k vytvoření okna zároveň se spuštěním patche, používám objekt *loadbang*, který posílá při načtení patche zprávu bang a tím aktivuje příkaz k vytvoření okna. K zavření okna dochází automaticky se zavřením patche a nebo ručně aktivací zprávy *destroy*. Červený spínač toggle v okně spustí a vypne rendering.

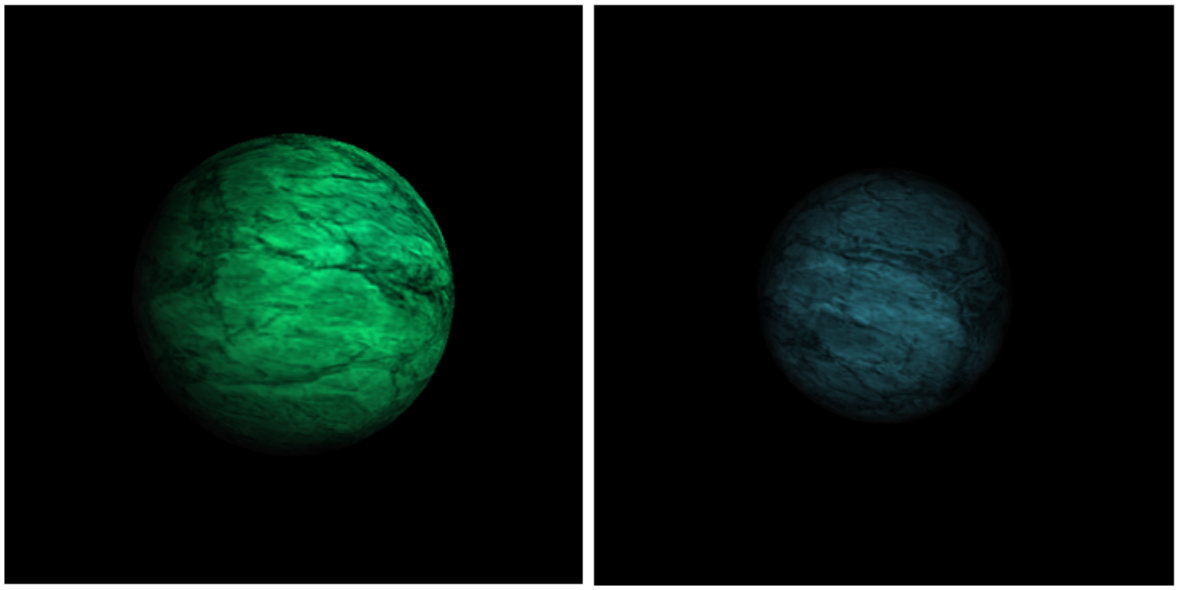


Obr. 4.1: Řetězec objektů pro nasvícení

Další nezbytnou součástí je objekt *gemhead*. Za něj se připojují všechny ostatní GEM objekty - gemhead je propojí se správcem grafického okna a zprostředkuje příkaz ke spuštění či zastavení renderingu. Argumentem objektu je číslo určující pořadí, ve kterém do-

stanou jednotlivé gemheady rendrovací příkaz. Objekty typu gemhead jsem použila celkem čtyři - tři pro vytvoření světel a jeden pro vytvoření vlastního tělesa.

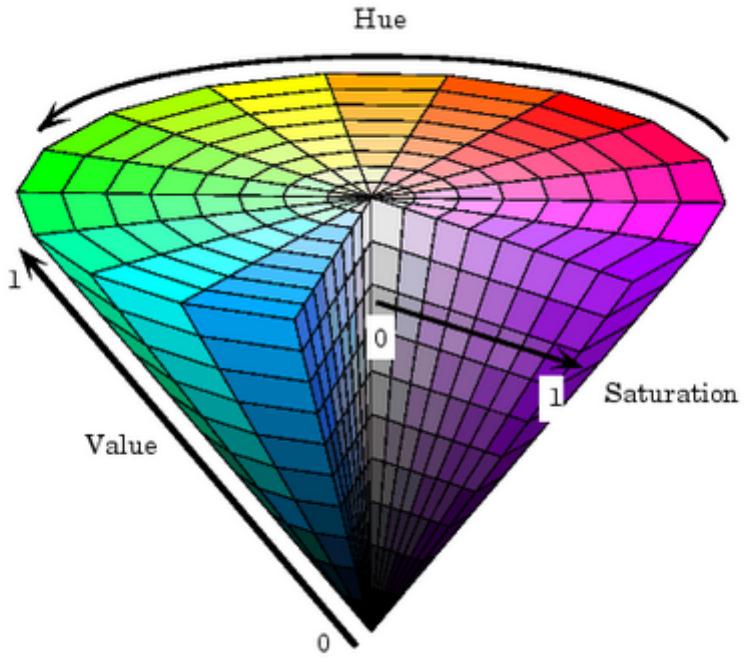
K nasvícení tělesa jsem využila tři objekty *light*. Knihovny GEM nabízí, stejně jako prostředí Max, možnost využít univerzální nasvícení pomocí objektu *world_light*. Toto světlo je nekonečně vzdáleno od všech těles v prostoru - světelné paprsky jsou proto uspořádány paralelně, což snižuje nároky na výpočet. Mým cílem bylo vytvořit v prostoru těleso připomínající planetu. S jedním zdrojem světla však nebyl výsledek zcela uspokojivý - scéna byla příliš tmavá. Proto k nasvícení planety používám tři objekty light. Objekt light má dva inlety - pravý inlet slouží pouze k nastavení barvy světla, levý inlet přijímá rendrovací příkaz. V patchi jsem k levému inletu připojila i zprávu *debug*. Tato zpráva usnadňuje umístění bodového zdroje světla do prostoru - při přijetí se zdroj vyrendruje jako viditelný bod. Umístění světel je řešeno pomocí objektu *translateXYZ*. Vektor posunutí tělesa lze měnit průběžně, a to přivedením zprávy float na jednotlivé cold inlety (v pořadí zleva doprava: x, y, z; obr. 4.1) a nebo mohou zůstat hodnoty statické. Pak se ve stejném pořadí zapisují jako argumenty k objektu. Hot inlet přijímá tzv. gemlist - jedná se o skupinu všech objektů GEM, začínající gemheadem a končící většinou rendrovaným tělesem. Objekty typu translate tedy mění všechny vektory v prostoru o dané posunutí.



Obr. 4.2: Nasvícení třemi bodovými zdroji (vlevo) a jedním zdrojem (vpravo)

Objekt translate se uplatňuje také v řetězci, který tvoří těleso - planetu. Kromě něj je zde použit také objekt *rotateXYZ*. Stejně jako translate přijímá gemlist a mění vektory o specifikovanou rotaci. Narozdíl od objektu translate však využívám u rotace možnost

měnit hodnoty otočení vůči osám v reálném čase. Aby při rotaci planety nebyly viditelné švy na textuře, natočila jsem planetu o 90° vůči ose *x* a švy se tak nachází na pólech planety. Kvůli natočení planeta rotuje kolem osy *z*. Objektem *color* je možné nastavit barvu pro všechny objekty v řetězci. V prvním inletu přijímá gemlist a; o druhým inletem se nastavuje barva v modelu RGBA (kanály: red, green, blue, alpha - červená, zelená, modrá a průsvitnost). Jelikož chci barvu objektu měnit v reálném čase, je pro tento účel vhodnější model HSV. Proto mám před objektem color předřazený převodník *hsv2rgb*, který přijímá list hodnot v pořadí hue (barva), saturation (sytost) a value (změna odstínu barvy).

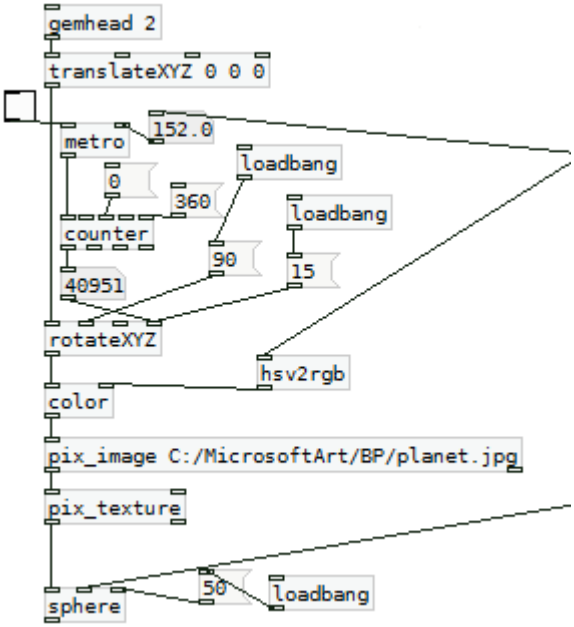


Obr. 4.3: Barevný model HSV [27]

Dalším bodem v řetězci planety je textura. Aplikace textury na těleso probíhá ve dvou bodech. Adresář, ve kterém je textura uložena se zadává jako argument objektu *pix_image*. Tento objekt texturu pouze načte do patche. Pro OS Windows je možné použít textury ve formátu TIFF a JPEG. Následující objekt *pix_texture* mapuje texturu na dané těleso. Jde však pouze o umístění textury na těleso, samotný rendering provádí až geometrický objekt na konci řetězce. Standardní barevný model užívaný v GEM je RGBA. GEM integruje zpracování pixelů a polygonální modelování geometrických těles. Polygonální model je síť složená z plošek (často trojúhelníků), která se tvarem přibližuje popisovanému objektu.[29] V případě použitého geometrického objektu - koule dojde prakticky k obalení tělesa texturou. U textury je upravena průsvitnost. Barva, která je aplikovaná na

těleso proto obarvuje i texturu. Změnou barvy polygonálně vytvořeného geometrického objektu dojde tedy i k barevné změně textury.[28]

Posledním článkem grafického řetezce je geometrický objekt *sphere*. Dva cold inlety nastavují průměr koule a počet bodů ze kterých bude koule zobrazena. Hot inlet přijímá gemlist a na základě informací v něm obsažených (z předcházejících objektů) se koule obarvuje, rotuje nebo se obaluje texturou.



Obr. 4.4: Řetězec objektů - planeta

4.2 Parametr - amplituda

Prostředí Pd dokáže zpracovat formáty WAV, AIFF a přímý vstup z mikrofonu. K načtení nahrávky je nutné sestavit řetězec. Prvním objektem bude *openpanel*, který otevře adresář a uživatel bude moci zvolit nahrávku. Otevření adresáře se spustí stisknutím tlačítka bang. Adresář je možné specifikovat pomocí zprávy směřované do hot inletu. Po vybrání souboru z adresáře se název souboru objeví na výstupu objektu. Za objektem openpanel bude následovat zpráva *open* pro otevření souboru. Nyní, když je soubor otevřen je nutné jej přečíst jako audio soubor. Toto zajišťuje objekt *readsf~*. Tilda za názvem značí, že jde o objekt operující s audio signály. Využívá pouze vzorky o 2, 3, maximálně 4 bytech. K aktivaci signálového výstupu je nutné do hot inletu nasměrovat zprávy *start* a *stop*. Argumentem objektu je float určující očekávaný počet kanálů. Objekt disponuje dvěma outputy - jak jsem již zmínila, jeden je signálový. Druhý output je datový a vyšle zprávu bang po ukončení čtení zvukového souboru. Pd nabízí více možností jak přečíst zvukový

soubor, např. pomocí objektu *soundfiler*. Narozdíl od signálového readsf jsou zde však omezení. Soundfiler načítá soubory do velikosti $4 \cdot 10^9$ vzorků, což při vzorkovací frekvenci 44100 Hz znamená cca 90 sekund délky souboru, což je pro většinu skladeb příliš málo.

Zde se signál větví a směřuje do objektů, kde probíhá zpracování a do výstupního objektu *dac~*. Počet argumentů objektu značí počet kanálů zvukové karty, do kterých bude signál směrován.

Pro výpočet amplitudy signálu jsem zvolila multifunkční objekt *fiddle~*. Jelikož se jedná o poměrně složitý objekt, bude mu věnována celá následující sekce, kde bude rovněž zmíněn i princip, kterým fiddle~zjišťuje hodnotu amplitudy. Jedná se prakticky o skutečnou špičkovou hodnotu (true peak) v dB. Tuto hodnotu posílám prostřednictvím objektu *send* do prostředního inletu objektu sphere. Tento geometrický objekt renderuje kouli a na jeho prostředním inletu lze pomocí zprávy float měnit průměr koule v reálném čase. Aby změny velikosti koule nebyly příliš skokové, v řetězci je vřazen objekt *average* s argumentem 10. Výsledná hodnota amplitudy je počítána jako průměr deseti hodnot a díky tomu dojde k mírnému vyhlazení náhlých změn ve velikosti koule.

4.3 fiddle~

Jedná se o signálový multifunkční nástroj dostupný v prostředích Max a Pure Data. Hlavní funkcí je odhad základní frekvence v reálném čase u zvuku s nekonstantní výškou.

Objekt poskytuje jeden vstupní inlet a pět výstupních outletů. Tilda v názvu objektu značí, že vstup je signálový, výstupy datové. Jednotlivé výstupy zleva doprava zprostředkovávájí tyto funkce:

- výstup float odhadující výšku základní frekvence nalezených stabilních tónů
- výstup bang reagující na nalezení stabilního tónu, pro určení základní frekvence
- výstup list poskytující informaci o hlasitosti a frekvenci zpracovávaného audia
- výstup float, amplituda signálu v dB
- výstup list, amplituda a frekvence harmonických složek signálu.

Volitelné argumenty objektu zleva doprava:

- velikost okna pro analýzu (128 - 2048, standardně 1024)
- počet výstupů, na kterých se objeví zjištěná frekvence (1 - 3, standardně 1)
- počet hledaných peaků (1 - 100, standardně 20)
- počet peaků směřovaných do výstupu (standardně 0)

Prvním krokem analýzy je diskretizace signálu z časové domény do domény frekvenční. To se běžně děje za použití diskrétní Fourierovy transformace (DFT). Jelikož tento způsob může být náročný na výpočet, nejčastější metodou je rychlá Fourierova transfor-

mace(FFT). Zdrojový kód objektu však napovídá, že zde byla transformace řešena jiným způsobem. Jde o vysoce optimalizovanou verzi Hartleyho transformace (DHT), která byla poprvé popsána v roce 1942, ale její vztah k DFT a FFT byl objeven až v roce 1983. Kód obsažený v objektu fiddle~je implementací DHT vytvořenou Ronem Mayerem v roce 1993.

Příchozí signál je členěn na segmenty o N vzorcích v rozmezí 256 až 2048. Nová analýza spektra proběhne každý $N/2$ vzorek. Pro každou analýzu je vzorek N doplněn nulami na $2N$ a voleno je obdélníkové okno. Výpočet probíhá ve třech krocích - vlastní výpočet, interpolace spektra a užití okna ve frekvenční doméně.

Vlastní výpočet

Máme reálný signál $x[n]$ o délce N, který modulujeme komplexní exponenciální funkcí $e^{-j\frac{\pi}{2N}n}$ a získáme

$$x_{mod}[n] \quad = \quad x[n]e^{-j\frac{\pi}{2N}n}. \tag{4.1}$$

Z tohoto modulovaného signálu spočítáme DFT jako

$$X_{mod}[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}n}e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}(4k+1)n} \tag{4.2}$$

pro $k \in [0, N-1]$. Nyní původní signál $x[n]$ doplníme o nuly na délku $4N$. DFT signálu $x_{nul}[n]$ bude

$$X_{nul}[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{4N}kn} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}kn} \tag{4.3}$$

pro $k \in [0, 4N-1]$. Je zřejmé, že

$$X_{mod}[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}(4k+1)n} = X_{nul}[4k+1] \tag{4.4}$$

a

$$X^*_{mod}[N-k-1] = \sum_{n=0}^{N-1} x[n]e^{j\frac{\pi}{2N}[4(N-k-1)+1]n} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{\pi}{2N}(4k+3)n} = X_{nul}[4k+3] \tag{4.5}$$

pro $k \in [0, \frac{N}{2}]$. Lze tedy spočítat lichou část spektra pouze se signálu X_{mod}, který je kratší než X_{nul}. K získání sudé části spektra je nutné přejít k dalšímu kroku - interpolaci.

Interpolace spektra

Pokud má DFT N členů posloupnosti na úseku k o $M \leq N$ členech signál $x[n]$ je dán

$$X[k] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}kn}. \tag{4.6}$$

Máme-li

$$X[k-l_\iota] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}(k-l_\iota)n} \tag{4.7}$$

$$X[k+l_\iota] = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}(k+l_\iota)n} \tag{4.8}$$

kde l_ι je kladná hodnota (z liché části spektra), pro součet signálů platí

$$\begin{aligned} X[k-l_\iota] + X[k+l_\iota] &= \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}(k-l_\iota)n} + \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}(k+l_\iota)n} \\ &= \sum_{n=0}^{M-1} x[n]\left(e^{-j\frac{2\pi}{N}(k-l_\iota)n} + e^{-j\frac{2\pi}{N}(k+l_\iota)n}\right) \\ &= 2\sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}kn} cos\left(\frac{2\pi}{N}l_\iota n\right). \end{aligned} \tag{4.9}$$

Suma všech hodnot l_ι je

$$\begin{aligned} \sum_{\iota=1}^{L} X[k-l_\iota] + X[k+l_\iota] &= 2\sum_{\iota=1}^{L}\sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}kn} cos\left(\frac{2\pi}{N}l_\iota n\right) \\ &= 2\sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}kn}\sum_{\iota=1}^{L} cos\left(\frac{2\pi}{N}l_\iota n\right). \end{aligned} \tag{4.10}$$

Když

$$\sum_{\iota=1}^{L} a_\iota cos\left(\frac{2\pi}{N}l_\iota n\right) = \frac{1}{2} \tag{4.11}$$

pro $n \in [0, M-1]$ a množina a_ι je poté

$$\sum_{\iota=1}^{L} a_\iota (X[k-l_\iota] + X[k+l_\iota]) = \sum_{n=0}^{M-1} x[n]e^{-j\frac{2\pi}{N}kn} = X[k]. \tag{4.12}$$

Je tedy možné spočítat hodnotu DFT signálu $X[k]$ jako superpozici hodnot DFT na každé straně $X[k]$. Aby byla splněna rovnice (4.10) musí existovat množina hodnot a_ι, a to taková, aby se po vynásobení s funkcí kosinus a sečtením sumy rovnala hodnota levé strany $1/2$.Pokud ale za n dosadíme počet vzorků 512, budou všechny funkce kosinus rovny nule a pravá strana tak hodnoty $1/2$ nedosáhne. Pokud však funkci kosinus zpozdíme o $\frac{\pi}{4}l_\iota$ rovnice (4.10) bude vypadat

$$\sum_{\iota=1}^{L} a_\iota cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)l_\iota\right] = \frac{1}{2}. \tag{4.13}$$

Nyní lze najít takové hodnoty a_ι, které budou téměř v souladu s rovnicí (4.12)

$$\begin{aligned} \mathbf{a} &= [a_1 a_2 \dots a_L]^T \\ \mathbf{b}(n) &= \left[cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)l_{\iota_1}\right] \quad cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)l_{\iota_2}\right] \dots \quad cos\left[\left(\frac{2\pi}{N}n - \frac{\pi}{4}\right)l_{\iota_L}\right]\right]^T \end{aligned} \tag{4.14}$$

Vzniklou chybu E definujeme jako

$$E = \sum_{n=0}^{M-1} \left[\mathbf{a}^T \mathbf{b}(n) - \frac{1}{2} \right]^2 = \mathbf{a}^T \left[\sum_{n=0}^{M-1} \mathbf{b}(n) \mathbf{b}^T(n) \right] \mathbf{a} - \mathbf{a}^T \sum_{n=0}^{M-1} \mathbf{b}(n) + \frac{1}{4}. \tag{4.15}$$

Derivaci rovnice (4.15) podle **a** položíme rovno nule

$$2 \left[\sum_{n=0}^{M-1} \mathbf{b}(n) \mathbf{b}^T(n) \right] \mathbf{a} - \sum_{n=0}^{M-1} \mathbf{b}(n) = 0 \Rightarrow \mathbf{a} = \frac{1}{2} \left[\sum_{n=0}^{M-1} \mathbf{b}(n) \mathbf{b}^T(n) \right]^{-1} \sum_{n=0}^{M-1} \mathbf{b}(n). \tag{4.16}$$

Pro objekt fiddle~s parametry $M = 512$, $N = 2048$ a $\{l_1, l_2, l_3, l_4, l_5\} = \{1, 3, 5, 7, 9\}$ vychází z rovnice (4.16) optimální parametr **a**

$$\mathbf{a} = [0.614965 - 0.154600 \quad 0.051131 - 0.013506 \quad 0.002033]^T. \tag{4.17}$$

V originálním kódu jsou tyto koeficienty téměř shodné. Je u nich však komentář autora, který říká, že těchto koeficientů bylo dosaženo metodou pokus - omyl. Podle autora článku How fiddle~Works (2011) Nathana Whetsella jsou koeficienty získané výše uvedeným výpočtem přesnější variantou.[30]

Užití okna ve frekvenční doméně

Ze spektra $X[k]$, na které bylo použito obdélníkové okno a bylo doplněno o nuly faktorem c lze získat spektrum $X_H[k]$, kde je užito Hannovo okno

$$X_H[k] = \frac{X[k]}{2} - \frac{X[k-c] + X[k+c]}{4} = (W_H * X)[k], \tag{4.18}$$

kde $W_H = [-\frac{1}{4}, 0, \ldots, 0, \frac{1}{2}, 0, \ldots, 0, -\frac{1}{4}]$. Po provedení zpětné diskrétní Fourierovy transformace, získáme Hannovu funkci

$$IDFT\{W_H[k]\} = \sum_{k=-c}^{c} W_H[k] e^{j\frac{2\pi}{M}kn} = -\frac{1}{4} e^{-j\frac{2\pi}{M}cn} + \frac{1}{2} - \frac{1}{4} e^{j\frac{2\pi}{M}cn} = \frac{1}{2} - \frac{1}{2} cos\left(\frac{2\pi}{M} cn\right). \tag{4.19}$$

Odhad frekvence výrazných spektrálních špiček

Odhad frekvence výrazných špiček ve spektru řeší fiddle~následujícím vzorcem:

$$\omega = \frac{\pi}{N} \left(k + re \left[\frac{X[k-2] - X[k+2]}{2X[k] - X[k-2] - X[k+2]} \right] \right). \tag{4.20}$$

Jedná se o speciální případ obecnějšího vzorce

$$\omega_{odhad} = \frac{2\pi}{N} \left(k + re \left\{ \frac{\beta(X[k-1] - X[k+1]) - 2\gamma(X[k-2] - X[k+2])}{2\alpha X[k] - \beta(X[k-1] + X[k+1]) + 2\gamma(X[k-2] + X[k+2])} \right\} \right) \tag{4.21}$$

kde α, β, γ jsou podmínky pro Blackmanovo okno. Platí $\{\alpha, \beta, \gamma\} = \{1, 0, -\frac{1}{4}\}$.

Amplituda signálu je dána jako nejsilnější špička ve vzorkovací periodě. Jelikož byl signál doplněn o nuly nebude se amplituda lišit od skutečné špičkové hodnoty (true peak) o více než 1 dB. Odhad fundamentálních frekvencí probíhá pomocí statistické věrohodnostní funkce - likelihood $L(f)$, kde f je frekvence. Přítomnost špiček poblíž nebo přímo na násobcích frekvence f zvyšuje hodnotu funkce $L(f)$, a to tak, aby se projevila amplituda a frekvence špičky. Funkce $L(f)$ se zapíše jako

$$L(f) = \sum_{i=0}^{k} a_i t_i n_i, \tag{4.22}$$

kde k je počet špiček ve spektru, a_i je faktor závisející na amplitudě i-té špičky, t_i závisí na vzdálenosti i-té špičky od násobku frekvence f, a n_i závisí na tom, zda je špička blíže celočíselným násobkům f nebo celočíselným podílům f. Přesná závislost těchto faktorů na frekvenci f je dána empiricky.[31]

4.4 Parametr - barva

Hudební pojem - barva tónu závisí na složení harmonického spektra a na poměrech mezi jeho jednotlivými složkami. Mým cílem je vytvořit vlastní objekt, který využije posledního outletu objektu fiddle~a na základě frekvencí a velikostí vyšších harmonckých složek zvuku vybere barvu, kterou se obarví rendrované těleso. Jde tedy o objekt, který bude mít na vstupu frekvenci a velikost jednotlivých harmonických (prvních pěti) a na výstupu bude poskytovat tři hodnoty hue, saturation a value (podle modelu HSV). K obarvení tělesa budu používat deset barev, které stanovil umělecký teoretik W. Kandinskij (viz 3.1 Propojení zvuku, barev a tvarů), kromě purpurové barvy, kterou naznačil tóny v nejvyšší části frekvenčního rozsahu. Empiricky jsem ale dospěla k tomu, že v tomto zobecněném případě, působí z estetického hlediska lépe bílá barva. Z jeho popisu je jasné, že barvy nástrojů odděloval jak frekvenčně, tak i co se týče poměrů jednotlivých harmonických složek. Například nejtmavší odstíny modré jsou reprezentovány temným zvukem varhan. Do řady za varhany je pak umístěn kontrabas, cello a nakonec flétna s nejsvětlejším odstínem modré. Stejně tak u dřevěných dechových dvouplátkových nástrojů je fagot v nižší části frekvenčního rozsahu reprezentován tmavě fialovou barvou a anglický roh ve vyšší části rozsahu světlým odstínem fialové. Podobná tendence je i u žesťů, kde basová tuba znamená červenou barvu a vyšší žestě ve fanfárách žlutou barvu.

Pro představu toho, jak na výstupu objektu fiddle~vypadají spektra zvuků, které psychoakusticky popsal Kandinskij jsem využila databanku samplů, kterou pro své syntezátory využívá firma Korg. Výsledky pro jednotlivá spektra jsou zohledněny v kódu objektu *kandinskij*. Pro potřeby komplexní nahrávky je však nutné kód velmi zobecnit. Především

nahrávky moderního charakteru téměř nedisponují typickou barvou dřevěných dvouplátkových nástrojů a také inharmonické spektrum typické pro zvon se objeví jen zřídka. Bez zobecnění popisu jednotlivých barev dosáhneme toho, že se geometrické těleso obarví jen zřídka a na velmi krátkou dobu. Při první implementaci kódu jsem pomocí podmínky `if` zjistila frekvenci na které se nachází základní frekvenční složka a v daném frekvenčním rozsahu jsem dále kód větvila pomocí dalších podmínek `if` na zjištění barvy, které momentální spektrum zvuku odpovídá. Kvůli specifickým spektrům nástrojů bylo však nutné ke každé části frekvenčního rozsahu přidat i podmínku `else`. Pokud barva zvuku neodpovídala žádnému nástroji, který Kandinskij ve své teorii popsal obarvovalo se těleso na odstíny šedé barvy. Pro komplexní zvuk nahrávky to však znamenalo, že bude těleso šedé po většinu času nahrávky. Frekventovaněji se objevila pouze žlutá barva znázorňující plné a postupně klesající spektrum podobné žesťům.
Pro implementaci se ukázala vhodná varianta natolik zobecněných barev, že bylo možné větvit kód v dané části frekvenčního rozsahu bez použití podmínky `else`. Nejdůležitějším parametrem je tedy frekvence základní harmonické složky, na dané části rozsahu se pak hledá vhodná barva na základě velikosti či poměrů harmonických složek. Závislost barev na základní frekvenci signálu:

- 0 až 60 Hz - černá
- 60 až 100 Hz - temná modrá a tmavá fialová (do určité velikosti prvního peaku ve spektru)
- 100 až 200 Hz - tmavá modrá a červená (v případě, že je třetí peak ve spektru je větší než druhý)
- 200 až 400 Hz - oranžová a světlá fialová (v případě, že je čtvrtý peak ve spektru větší než první)
- 400 až 700 Hz - zelená
- 700 až 1200 Hz - žlutá a světlá modrá (do určité velikosti prvního peaku ve spektru)
- vyšší než 1200 Hz - bílá

Objekt je napsán v jazyce C a jedná se o dynamicky připojovanou knihovnu (DLL). Pro tvorbu nových rozšíření poskytuje Pd univerzální header `m_pd.h`, který slouží jako rozhraní a obsahuje všechny předdefinované typy, metody apod. Většina autorů dodržuje v kódu svých objektů jednotné uspořádání. To v případě Pd znamená, že kód začíná definicí třídy a struktury, jejíž proměnná je typu `t_object`. Tento typ ukládá vnitřní vlastnosti objektů - jejich grafickou reprezentaci a informace o inletech a outletech. V mém případě bude mít objekt jeden inlet a jeden outlet. Pokud se jedná o inlet typu hot, není nutné jej ve struktuře deklarovat.

```
#include "m_pd.h"

static t_class *kandinskij_class;

typedef struct _kandinskij {
```

```
  t_object x_obj;
  t_outlet *t_outH;  //deklarace outletu
}t_kandinskij;
```

Vlastní jádro, kde pomocí podmínek if probíhá zjištění hodnot HSV, se nachází ve funkci `kandinskij_float`. Argumenty této funkce jsou všechny vstupní hodnoty a ukazatel na proměnnou x typu `t_object`. Jelikož objekt operuje s šesti vstupními a třemi výstupními hodnotami, lze jej řešit dvěma způsoby. První možností je vytvořit objekt s šesti inlety a třemi outlety s tím, že přijímaná a odesílaná zpráva bude typu float. Druhou možností je vytvořit objekt s jedním inletem a jedním outletem s tím, že přijímaná a odesílaná zpráva bude typu list. Při implementaci objektu používám právě tuto druhou možnost.

```
static void kandinskij_float(t_kandinskij *x, t_floatarg firstHz,
  t_floatarg firstAmp, t_floatarg secondAmp,
  t_floatarg thirdAmp, t_floatarg fourthAmp,
  t_floatarg fifthAmp)
{
  t_atom argv[3];       //výstupní list o třech hodnotách
  float h = 0, s, v;

if ((firstHz >= 0) && (firstHz < 60)) {
  h = 0;
  s = 1;
  v = 0;
}
if ((firstHz >= 60) && (firstHz < 100)) {
   if (firstAmp > 0.06){ h = 0.69; s = 1; v = 0.19; }
   if (firstAmp <= 0.06){ h = 0.84; s = 1; v = 0.25; }
}
if ((firstHz >= 100) && (firstHz < 200)) {
  if (secondAmp >=  thirdAmp) { h = 0.69; s = 1; v = 0.63; }
  if (secondAmp < thirdAmp) { h = 0; s = 1; v = 0.63; }
}
if ((firstHz >= 200) && (firstHz < 400)) {
  if (fourthAmp >= firstAmp) { h = 0.84; s = 0.44; v = 0.66; }
  if (fourthAmp < firstAmp) { h = 0.05; s = 1; v = 0.88; }
}
if ((firstHz >= 400) && (firstHz < 700)) {
  h = 0.55; s = 0.7; v = 0.6;
}
if ((firstHz >= 700) && (firstHz < 1200)) {
    if (firstAmp <= 0.25 ) { h = 0.5; s = 1; v = 1; }
    if (firstAmp > 0.25 ) { h = 0.15; s = 1; v = 1; }
}
if (firstHz >= 1200) {
    h = 0.5;
    s = 0;
    v = 1;
}
SETFLOAT(&argv[0], h);        //přiřazení hodnot jednotlivým parametrům argv
SETFLOAT(&argv[1], s);
SETFLOAT(&argv[2], v);
outlet_list(x->t_outH, &s_list, 3, argv);   //nasměrování listu do výstupu
}
```

Ostatní funkce objektu jsou „formality“ nutné pro funkčnost objektu, jeho vstupů a výstupů.

```
static void *kandinskij_new(void)   //konstruktor, funkce je zavolána při vytvoření objektu
{
  t_kandinskij *x = (t_kandinskij *)pd_new(kandinskij_class);
      x->t_outH = outlet_new(&x->x_obj, 0);

  return(x);
}

void kandinskij_free(t_kandinskij *x)   //funkce free slouží k uvolnění
{
outlet_free(x->t_outH);     //alokovaného místa po smazání objektu
}

void kandinskij_list(t_kandinskij *x, t_symbol *sym, int argc, t_atom *argv)
{
    if(argc >= 6){            //zde se definuje list vstupních hodnot
    float firstHz = atom_getfloat(&argv[0]);
    float firstAmp = atom_getfloat(&argv[1]);
    float secondAmp = atom_getfloat(&argv[2]);
    float thirdAmp = atom_getfloat(&argv[3]);
    float fourthAmp = atom_getfloat(&argv[4]);
    float fifthAmp = atom_getfloat(&argv[5]);
    kandinskij_float(x, firstHz, firstAmp, secondAmp, thirdAmp, fourthAmp, fifthAmp);
    }
}

void kandinskij_setup(void)
{
  kandinskij_class = class_new(gensym("kandinskij"), (t_newmethod)kandinskij_new,
                               (t_method)kandinskij_free,
    sizeof(t_kandinskij), CLASS_DEFAULT, 0);
  class_addlist(kandinskij_class, (t_method)kandinskij_list);
}
//funkce setup se volá pouze jednou - při načtení knihovny
//musí obsahovat deklaraci všech tříd a jejich vlastností
```

Argumenty funkce `class_new` jsou v pořadí zleva doprava: jméno třídy, konstruktor, destruktor, velikost dat pro rezervování paměti, grafická reprezentace objektu, další argumenty.

V patchi se pro seřazení hodnot do zprávy typu list uplatňuje objekt *pack*, pro rozbalení listu objekt *unpack*. Před objekt pack jsou předřazeny objekty typu trigger, aby se zajistilo, že zpráva float přijde na všechny inlety objektu pack ve stejný okamžik.

4.5 Parametr - tempo

Ke zjištění attacků v nahrávce využívám signálový objekt *bonk~*. Při vzorkovací frekvenci $f_vz = 44.1kHz$ je velikost okna pro analýzu 256 vzorků (= 5.8 msec). Prvním krokem analýzy je podvzorkování pomocí FIR filtrů. Je-li velikost okna N vzorků, $M = N/2$ a $x[n]$ je vstupní signál, kde $n = 0, \ldots, N-1$. Dále pak ω odpovídá střední frekvenci filtru

a δ je šířka pásma daného filtru. Pro energii signálu platí

$$P(\omega, \delta) = \left| \sum_{m=-T}^{T-1} \exp i\omega m \frac{1 + cos(\delta m)}{2} x[M + m] \right|^2 \tag{4.23}$$

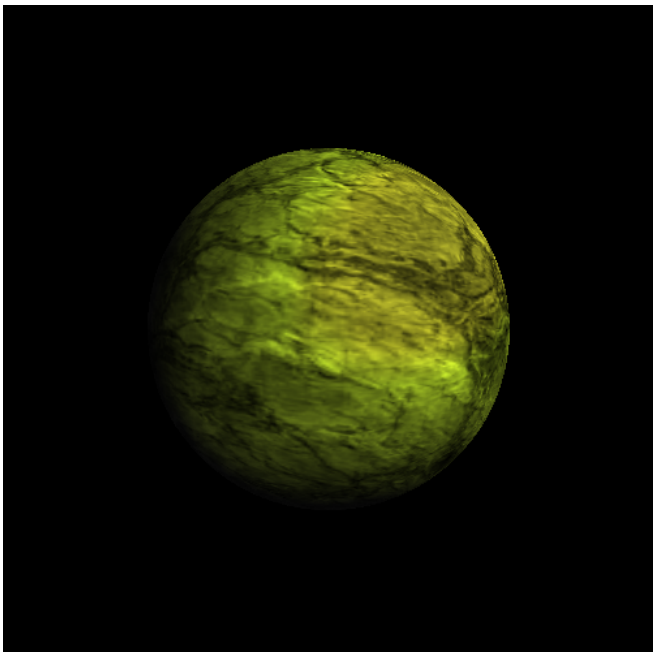
kde $T = [\frac{\pi}{\delta}]$.

Bonk~detekuje pouze ostré změny ve spektru signálu, tzn. rozsáhlá změna energie ve spektru nezamaskuje hledané attacky. To se může stát, pokud nahrávka obsahuje cinkání nebo zvonění - zvuk tohoto typu způsobí nárůst ve spektru a v případě detekce attacků pomocí sledovače obálky, tak dochází k maskování. Bonk~na celém frekvenčním rozsahu používá k analýze 11 filtrů. Pomocí zprávy je možné pro objekt bonk~nastavit threshold, kterým určíme nejvyšší hodnotu, kterou musí ostrý nárůst překročit a nižší hodnotu, na kterou musí následně energie signálu spadnout, aby se změna detekovala jako attack.[31] Threshold tedy závisí na charakteru nahrávky a ideálně by se měl přenastavit při každé změně skladby. Při výpočtu časového rozdílu mezi attacky používám univerzální hodnoty thresholdu a časový rozdíl průměruji z 10 hodnot.

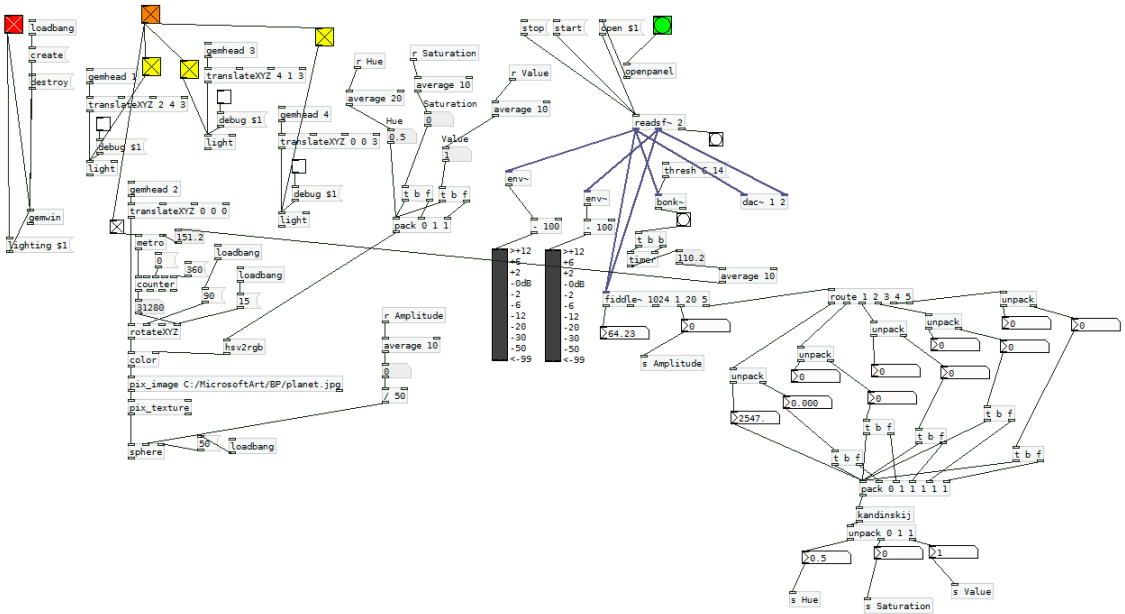
4.6 Výsledky

Pure Data je prostředí, které je uzpůsobeno speciálně pro vývoj vizualizací, syntezátorů a podobných aplikací. Mínusem pro Pure Data jsou omezenější možnosti zpracování 2D a 3D objektů. Herní enginy v tomto ohledu poskytují nepřeberné množství nástrojů. Ovšem podstatnou částí vizualizace je také zpracování audia. To je „kámen úrazu“ pro herní enginy. Enginy poskytují spíše specifické možnosti zpracování audia - zvuk lze rozmístit libovolně v 3D prostoru, simulovat jeho vzdálenost od posluchače apod. Co se týká získávání parametrů z nahrávky, poskytují enginy pouze omezené možnosti (Unity) a nebo nejsou kompatibilní s žádnou vhodnou knihovnou obsahující tyto funkce (Unreal Engine). V tomto ohledu je prostředí Pure Data ideální pro vývoj vizualizace. Ve vytvořeném patchi získávám tři parametry - amplitudu (true peak), barvu a tempo skladby. Ve výsledku všechny získané hodnoty průměruji, aby došlo k mírnému vyhlazení změn v obrazové části. Nejedná se tedy o přesnou vizualizaci parametrů nahrávky, jak je tomu v případě měřící techniky. Přesto by pro manipulaci s některými parametry bylo vhodné vyhledat sofistikovanější metodu. Pro dosažení větší estetičnosti v obrazové části by bylo vhodné v objektu *kandinskij* implementovat místo podmínek if vzorec. Tím by se zajistily plynulé změny barvy, což by bylo především u nahrávek v pomalejším tempu příjemnější.

Výstupem obrazové části je koule se strukturou planety, která mění svoji velikost, rotuje a mění barvu. V důsledku výpočtu průměru nastává při spuštění vizualizace krátké opoždění rotace. Aby k tomuto jevu nedocházelo, bylo by nutné nastavit threshold objektu bonk~pro každou nahrávku zvlášť. Při dobře nastaveném thresholdu je průměrování zbytečné.



Obr. 4.5: Obrazový výstup



Obr. 4.6: Řešení vizualizace v Pd

5 ZÁVĚR

V této práci se věnuji teorii zvuku a jeho šíření, jak z uměleckého, tak z technického hlediska a parametrům hudební nahrávky, rovněž z obou hledisek. Dále shrnuji vývoj v oblasti vizualizace hudby, a to od tance - tedy historické umělecké formy po první mechanické či elektronicky řízené vizualizace. Zmiňuji také měřící techniku, což je jediná oblast parametrické vizualizace zvuku, ve které se neklade důraz na umělecké zpracování, ale na technickou stránku řešení, k dosažení co nejpřesnějších hodnot. Pokouším se také komplexně obsáhnout širokou škálu forem parametrické vizualizace zvuku dnes. Přestože většina z nich je spojena se zábavním průmyslem a hudební produkcí, jednotlivé formy se od sebe liší jak technickým zpracováním, tak možnostmi samotné vizualizace. Co se týče právě možností vizualizace, vynikají značně softwarová řešení, protože lze v reálném čase zobrazit více parametrů najednou a to různými způsoby. Před samotným technickým řešením věnuji pozornost také umělecké teorii pojednávající o vhodných spojeních mezi barvami a zvuky, a také doporučením ohledně použitých tvarů a využití prostoru. Posledním tématem je technická stránka projektu a jeho realizace. Nejpreve zde krátce popisuji pokus o řešení zadání v herním enginu. Od tohoto záměru jsem však upustila, jelikož získání parametrů z nahrávky nebylo v možnostech daného enginu. Dále popisuji postup tvorby vizualizace v prostředí Pure Data a principy, kterými se získávají parametry z nahrávky.

V rámci práce se zabývám třemi parametry - amplitudou, barvou a tempem nahrávky. Pro řízení vizualizace barvy, jsem vytvořila jednoduché rozšíření pro Pd, které převádí hudební barvu nástroje do barevného modelu HSV podle návrhu uměleckého teoretika W. Kandinského. Samotná vizualizace probíhá ve zobrazovacím okně na modelu planety. Změna amplitudy mění průměr planety, změna barvy mění její obarvení a změna tempa mění rychlost její rotace.

Možné rozšíření práce by mohlo spočívat ve vytvoření vzorce pro převod hudebních barev do modelu HSV. Změna barvy by tak probíhala plynule, což by mohlo zlepšit estetický dojem na diváka. Dalším bodem by mohlo být vytvoření módů vizualizace pro různé hudební žánry. Parametr tempo je závislý na thresholdu, který je nastaven na univerzální hodnoty. Ideálně by se však měly hodnoty thresholdu měnit s žánrem skladby, aby se zpřesnilo odečítání attacků z nahrávky a tím pádem i rychlost a stabilita rotace planety.

LITERATURA

[1] BAIN, Matthew. *REAL TIME MUSIC VISUALIZATION: A STUDY IN THE VISUAL EXTENSION OF MUSIC* [online]. The Ohio State University, 2008 [cit. 2016-11-29]. Dostupné z: https://etd.ohiolink.edu/rws_etd/document/get/osu1213207395/inline

[2] BERMAN, Greta. Synesthesia and the Arts. *Leonardo* [online]. 1999, **32**(1), 15-22 [cit. 2016-12-03]. DOI: 10.1162/002409499552957. ISSN 0024-094X. Dostupné z: http://postcog.ucd.ie/files/b1_SynesthesiaAndTheArts.pdf

[3] BROUGHER, Kerry, Jeremy STRICK, Ari WISEMAN a Judith ZILCZER. *Visual Music: Synaesthesia in Art and Music Since 1900*. 1.vydání. London: Thames and Hudson, 2005. ISBN 978-0500512173.

[4] COLLINS, Nicholas a Julio D' ESCRIVAN RINCÓN. *The Cambridge Companion to Electronic Music*. 1. edition. Cambridge: Cambridge University Press, 2007. ISBN 9780521688659.

[5] FULKS, Michael. Negative Space Photography In Photo Composition. In: *Apogee Photo Magazine* [online]. London, GB: Apogee Photo Magazine, 2016 [cit. 2016-12-03]. Dostupné z: http://www.apogeephoto.com/negative-space-photography-in-photo-composition/

[6] GERSTNER, Karl. *Forms of Color: The Interaction of Visual Elements*. 1. edition. Cambridge, Massachusetts: The MIT Press, 1990. ISBN 978-0262570817.

[7] GRIGROVÁ, Věra. *Všeobecná hudební nauka*. První. Olomouc: ALDA, 2003. ISBN 80-85600-46-3.

[8] HOGG, Kirsten. *Timeline of the development of the oscilloscope* [online]. První. Sydney: School of Physics, The University of Sydney, 2006 [cit. 2016-11-09]. Dostupné z: http://science.uniserve.edu.au/school/curric/stage6/phys/stw2002/hogg.pdf

[9] JIRÁSEK, Ondřej. *Zvukové vlny - prezentace k předmětu Zvukové aspekty interpretace* [online]. Výzkumné centrum JAMU [cit. 2016-11-27]. Dostupné z: http://www.audified.com/projekt/vavcjamu/vyuka/page95/files/I_zvukove_vlny.pdf

[10] KEPES, Gyorgy. *Language of Vision* [online]. 2.vydání, dotisk. Courier Corporation, 1995 [cit. 2016-12-01]. ISBN 978-0486286501. Dostupné z:

https://books.google.cz/books/about/Language_of_Vision.html?id=zDWhvpEekFAC&redir_esc=y

[11] POPPER, Frank. *Origins and Development of Kinetic Art*. the University of Michigan, USA: New York Graphic Society, 1968. ISBN 978-0289795927.

[12] SCHIMMEL, Jiří. *Elektroakustika* [online]. První. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací, Brno [cit. 2016-12-03]. ISBN 978-80-214-4716-5. Dostupné z: https://www.vutbr.cz/www_base/priloha.php?dpid=71624

[13] ZAHRADNÍČEK, Josef. Kmitající plaménky. *Časopis pro pěstování matematiky a fysiky*. Praha: Union of Czech Mathematicians and Physicists, 1933, **62**(2.), 1-3. ISSN 1802-114X.

[14] *Animusic* [online]. Ithaca, NY., USA: Animusic, LLC., 2012 [cit. 2016-11-29]. Dostupné z: http://animusic.com/

[15] *Crypton* [online]. Sapporo, Japan: Crypton Future Media, 2016 [cit. 2016-11-27]. Dostupné z: http://www.crypton.co.jp/

[16] *ESSENTIA* [online]. Barcelona, Spain: Universitat Pompeu Fabra, 2016 [cit. 2016-12-08]. Dostupné z: http://essentia.upf.edu/

[17] How concerts shifted from songs to spectacles. *The Washington Post* [online]. The Washington Post, 2014 [cit. 2016-11-24]. Dostupné z: https://www.washingtonpost.com/entertainment/music/how-concerts-shifted-from-songs-to-spectacles/2014/05/22/ca521340-d6ce-11e3-8a78-8fe50322a72c_story.html?utm_term=.413da8b88e01

[18] *Merriam-Webster* [online]. Springfield, Massachusetts, USA [cit. 2016-11-20]. Dostupné z: https://www.merriam-webster.com/

[19] Monophonic, Stereophonic and Surround Sound Differences. *Lifewire* [online]. [cit. 2016-11-08]. Dostupné z: https://www.lifewire.com/mono-stereophonic-surround-sound-3134845

[20] Oscilloscope Types. *Radio-Electronics* [online]. [cit. 2016-11-15]. Dostupné z: http://www.radio-electronics.com/info/t_and_m/oscilloscope/oscilloscope_types.php

[21] *Sound Visualization and Analysis in the Pre-Electronic Era - Visualization* [online]. Harvard College, 2005 [cit. 2016-11-21]. Dostupné z: http://www.seas.harvard.edu/

[22] *Steinberg* [online]. Steinberg Media Technologies, 2016 [cit. 2016-12-01]. Dostupné z: https://www.steinberg.net

[23] The Dream of Color Music, And Machines That Made it Possible. *Animation World Network* [online]. AWN, 2015 [cit. 2016-11-20]. Dostupné z: http://www.awn.com/mag/issue2.1/articles/moritz2.1.html

[24] *Unreal Engine* [online]. EPIC GAMES, 2016 [cit. 2016-12-07]. Dostupné z: https://www.unrealengine.com/

[25] *Pure Data* [online]. Graz, Austria: Institute of Electronic Music and Acoustics, c2016 [cit. 2017-04-29]. Dostupné z: https://puredata.info/

[26] CHOW, Ming a Paul LEHRMAN. *Starting with Pure Data* [online]. In: . s. 5-18 [cit. 2017-05-25]. Dostupné z: http://tuftsdev.github.io/MusicAppsOnTheIpad

[27] HSV model. In: *Shaeod* [online]. [cit. 2017-06-07]. Dostupné z: http://shaeod.tistory.com/165

[28] DANS, Mark. *Real-time Image and Video Processing in GEM* [online]. 1997 [cit. 2017-06-01]. Dostupné z: https://puredata.info/downloads/gem/documentation/manual/pub/danks1997realtime.pdf

[29] BROOKSHEAR, Glenn. *Informatika* .První. Brno: Computer Press, Albatros Media, 2013 [cit. 2017-06-07]. ISBN 0132569035.

[30] WHETSELL, Nathan. *How fiddle~Works* [online]. [cit. 2017-06-01]. Dostupné z: http://www.music.mcgill.ca/~ich/classes/mumt621_11/final/20projects/final/Report.pdf

[31] PUCKETTE, Miller. qtextitReal-time audio analysis tools for Pd and MSP [online]. [cit. 2017-06-01]. Dostupné z: http://crca.ucsd.edu/~tapel/icmc98.pdf