**MASARYKOVA UNIVERZITA**
**Filozofická fakulta**
**Ústav hudební vědy**

Teorie interaktivních médií

# Live coding

Bakalářská diplomová práce

Kryštof Bernard Hájek
UČO: 420916

Vedoucí práce: PhDr. Martin Flašar, Ph.D.

2016

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s využitím uvedených pramenů a literatury.

V Praze dne 12. května 2016

.....

Kryštof Bernard Hájek

***„Umění musí držet krok s vědou."***

*Edgar Varèse*

Zde bych rád poděkoval několika lidem. Především Mgr. Martinu Flašarovi, Ph.D. za jeho ochotu, trpělivost a vedení práce, a mým rodičům za jejich neutuchající podporu během mého náročného studia.
Velký dík patří také pivovarům Staropramen, Březňák, Klášter, Kozel a Braník, a četným českým i zahraničním čokoládovnám za jejich výrobky, které mne udržovaly ve stavu psychické vyrovnanosti a umožnily mi tak nerušeně psát a tvořit.
Děkuji také všem mým blízkým kamarádům, kteří do poslední chvíle nepřestali věřit, že svou práci úspěšně dokončím.
Speciální poděkování pak patří mému příteli, jenž při mně stál ve chvílích nejtěžších a bez nějž bych tuto práci nikdy nedokončil.

Kryštof Bernard Hájek

# Obsah

Úvod....5

1. Kultura softwaru....7

2. Počítačová hudba....9

2.1. Počátky generativní hudby - Illiac Suite....9

2.2. Počátky zvukové syntézy - Max Mathews....11

3. Algoritmická kompozice....12

3.1. Úvod....12

3.2. Algoritmická kompozice bez počítače....12

3.3. Algoritmická kompozice s užitím počítače....13

3.4. Současné techniky algoritmické kompozice....15

4. Live coding....17

4.1. Úvod....17

4.2. Historie živého kódování....18

4.3. The Slub....20

4.4. Další live codingová vystoupení a umělci....21

4.5. TOPLAP....22

4.6. Vystoupení....23

5. Textová programovací prostředí....27

5.1. SuperCollider....27

5.2. Impromptu....28

6. Ostatní programovací prostředí....31

6.1. Grafické programovací prostředí PureData....31

6.2. Personalizovaná prostředí vytvořená v hnutí TOPLAP....33

Závěr....34

Resumé....35

Summary....36

Seznam zdrojů a použité literatury....37

# Úvod

Jako téma své bakalářské práce jsem si zvolil umění živé algoritmické hudební kompozice nazývané jako *live coding.* Když jsem začal tohoto termínu užívat v okruhu svých známých a přátel, většina z nich nebyla s to jej identifikovat nebo alespoň oborově zařadit. Není divu, jde o obor stále poměrně mladý a bohužel také nadále převážně okrajový.

Live coding nebo také umění živého kódování je název pro činnost, kdy je psaní a manipulování se softwarem generující audio a/nebo vizuální výstupy bráno jako hlavní součást živého vystoupení. Tím jsou zdůrazňovány a vyzdvihovány výrazové prostředky poskytované různými programovacími jazyky coby prostředky pro tvorbu a upravování zvuku i vizuálu.

Ač samozřejmě v živém kódování hudby zanechává svou viditelnou stopu médium k němu užívané, jde v zásadě především o záležitost stylistickou.

Má práce začíná širším přehledem zahrnujícím pohled na kulturu softwaru, v jejímž prostředí mělo živé kódování možnost vzniknout. Značný prostor je věnován také historii počítačové hudby, jenž umožňuje live coding zařadit do kulturně sociálního kontextu. V první kapitole **Historie** se věnuji prvotní staré myšlence o propojení hudby a počítačů, a přibližuji první odvážlivce, kteří se na tehdy těžko tonážních a pomalých strojích snažili vytvořit něco tak krásného a křehkého, jako je hudba. Dostatek prostoru dostane jak slavná skladba z těchto počátků počítačové hudby *Illiac Suite,* tak první programy schopné generování či upravování hudby.

Celá druhá kapitola se zabývá jedním ze základních stavebních kamenů live codingu, a sice **algoritmickou hudební kompozicí.** Zde zabíhám opět i do historie, protože hudební skladby pracující s jednoduchým algoritmickým principem je možné najít již v 18. století. Větší prostor ovšem samozřejmě dostává moderní algoritmická kompozice, tedy ta využívající počítač. Představím zde osobnosti počítačové hudby jako *Lejarena Hillera, Johna Cage* nebo „skládajícího architekta" *Iannise Xenakise*, a vysvětlím rozdíly mezi různými přístupy k algoritmické tvorbě hudby.

Na konec ještě přibližuji některé techniky využívané v současné algoritmické kompozici. Záměrně jen některé, poněvadž jich existuje skutečně velké množství, a navíc se dnes nejčastěji používají kombinace několika těchto systémů.

Kapitola třetí je už celá věnována čistě jen umění **live codingu**. Vysvětlím, co si přesně pod tímto pojmem představit, do jaké formy umění vlastně spadá, nebo jak o něm smýšlejí členové live codingové komunity. O co je historie živého kódování kratší, o to důležitější je zmínit každou událost, která tento obor formovala a dala mu dnešní tvář. Proto zde píšu o prvním veřejném live codingovém vystoupení či prvních uskupeních, z nichž nejznámější a nejdůležitější je skupina *Slub***,** proto je jí, i jejím třem členům, věnována samostatná podkapitola.

Následuje přiblížení osobností a prací dalších význačných osobností živého kódování jako *Juliana Rohrhubera* nebo *Andrewa Sorensena,* a fungování organizace TOPLAP.

Důležitou a podle toho obsáhlou podkapitolou je **Vystoupení,** detailně popisující průběh vystoupení, práci s programovacím jazykem nebo úroveň propojení s posluchači, potažmo diváky.

Poslední dvě kapitoly **Textová** a **Grafická programová prostředí** jsou pak texty převážně technickými, popisujícími jednotlivá programovací prostředí využívaná pro live coding, jejich nástroje a možnosti, uživatelskou přívětivost a hlavní myšlenky jejich tvůrců.

Cílem mé práce je zmapování a přiblížení live codingu a kultury jej obklopující. Kladu důraz na jeho historický kontext, na to, co mu předcházelo, z čeho vykrystalizoval a kde se vůbec v lidech brala touha vytvářet hudbu živě pomocí algoritmů. Zajímají mě osobnosti, jež zasvětili velký díl svého životního času právě tomuto odvětví a vidí v něm budoucí potenciál. Stejnou pozornost věnuji technické stránce věci, která je samozřejmě v live codingu velice důležitá a zaslouží si být dopodrobna prozkoumat. Myslím si, že umění živého kódování je hodnotná a pozoruhodná umělecká forma, která by měla být přiblížena co největšímu počtu lidí.

Přesto, že se publikováním článků o něm zabývá mnoho lidí, nikde jsem nenašel takový text, který by shrnoval všechny informace jakkoli se týkající live codingu; úplnými prvopočátky algoritmického umění počínaje a současnými umělci tohoto odvětví konče. A jsem si jist, že live coding si to plně zaslouží.

Materiál tuto práci jsem čerpal převážně z elektronických zdrojů, jenž byly pro toto téma často jediné dostupné. Pro první kapitolu jsou stěžejní myšlenky z předmluvy knihy Lva Manoviche *Software Takes Command.* Pro druhou kapitolu, zabývající se historií počítačové hudby, byla důležitým zdrojem publikace pánů Hillera a Isaacsona *Experimental Music: Composition With an Electronic Computer,* a také články italského hudebního teoretika Alexe di Nunzia z jeho webové stránky zabývající počítačovou hudbou *Musica informatica.* Jím podepsaná pojednání jsou jedním z podkladů i další kapitoly o algoritmických kompozicích. Kromě nich tato část čerpá dostatek informací z publikace Adama Alperna *Techniques for algoritmic composition of music* a také z knihy německého skladatele a teoretika Gerharda Nierhause *Algorithmic Composition: Paradigms of Automated Music Generation.*

Pro kapitolu věnující se čistě live codingu byly nejdůležitější webové stránky organizace TOPLAP a díla umělců live codingem se zabývajících. Mezi ty patří práce Andrewa Sorensena a Andrewa R. Browna *aa-cell In Practice: An Approach To Musical Live Coding,* a *Live Algorithm Programming And A Temporary Organisation For Its Promotion* napsaná v kolektivu Adriana Warda, Juliana Rohrhubera a mnoha dalších programátorů a hudebníků. Pro podkapitolu *Vystoupení* byla pak také nezbytné pojednání Nicka Collinse a Alexe McLeana *Live coding in laptop performance*, publikované v časopise *Organised Sound.*

Poslední kapitola věnující se programovacím prostředím pak čerpá přímo z textů jejich tvůrců - Andrewa Sorensena nebo Millera Pucketteho, a také z jiné bakalářské práce *PureData: Nástroj pro algoritmické umění* Ivana Floreše.

# 1. Kultura softwaru

Celá naše společnost je, krom jiných, propojena a ovládána jedním důležitým prvkem. Prvkem ne moc starým, do civilizace začal pronikat teprve před pár desítkami let. S každým rokem se však stává mocnějším, a jeho propojení společnosti těsnějším a dokonalejším. Zcela nenápadně a ve vší tichosti se mu podařilo ovládnout většinu aspektů našeho života. Ráno nás dokáže probudit, uvařit kávu podle přání, vypočítá, jaké zpoždění očekávat při dnešní cestě do práce. Někteří s ním pak i tráví celou pracovní dobu, ba dokonce by bez něj někteří ani žádnou práci neměli. Když v ní pak skončí a žádají zábavu, opět se obrátí o pomoc k němu. Tedy samozřejmě k softwaru. Ten je tou nehmotnou hybnou silou současného světa, již někteří vnímají více, někteří méně, někdo vůbec. Zasahuje však do životů všech.

Termín Lva Manoviche „kulturní software" je zcela na místě. Kulturní proto, že, jak tento asi neznámější teoretik nových médií vysvětluje, *„that it is directly used by hundreds of millions of people and that it carries "atoms" of culture (media and information, as well as human interactions around these media and information) – is only the visible part of a much larger software universe."*[1]

O revoluce v informačních technologiích (coby původci kultury softwaru), byly již popsány stohy papíru a zaplněny terabajty počítačových uložišť. Svůj pohled na věc sdělili filosofové, sociologové, psychologové, programátoři a samozřejmě i teoretici nových i starých médií. Dokonce díky této revoluci vznikly nové obory výzkumu, jako třeba kyberkultura nebo teorie nových médií. Zvláštní však je, že hnací sílu celého tohoto moderního směru - software - většina těchto vědců a teoretiků vytrvale opomíjí. Vzhledem k dnešní situaci, kdy jsou lidé skutečně zahlceni nejrůznějšími aplikacemi a neumí si bez nich svůj život představit, je to vskutku paradoxní.

Výjimkou je právě zmiňovaný rusko-americký teoretik Lev Manovich, jenž pochopil, jakou silou působí software na naši společnost, a bohatě tento aspekt rozebírá ve svých knihách *The Language of New Media* a *Software Takes Command.*

V úvodu druhé jmenové publikace zmiňuje slova svého „předchůdce" Marshalla McLuhana o tom, že abychom pochopili logiku nových médií, musíme na ni nahlížet pomocí počítačových věd, a posunout se od studií médií k něčemu, co se dá nazývat studii softwarovými. McLuhan tímto staví počítačovou vědu na pozici jakési absolutní pravdy, kterou se musí řídit vše okolo a která nám vysvětlí, jak v dnešní společnosti softwaru funguje kultura. *„But computer science is itself part of culture"*[2], namítá Manovich. Tím pádem se musí softwarová studia zkoumat jak roli softwaru coby nástroje formujícího současnou kulturu, a sociální, kulturní a ekonomické vlivy, jež formují vývoj samotného softwaru.

Zároveň však Manovich netvrdí, že se dnešní kultura plně identifikuje s aplikačním softwarovým vybavením, a že mohou její výtvory vznikat pouze s jejich pomocí. Základem zkoumání softwarových studií je fakt, že se právě díky softwaru na konci 20. století radikálně proměnil obsah kultury celé naší civilizace.

---

[1] MANOVICH, Lev. *Cultural Software.* [online] 2011. Dostupné z: http://manovich.net/content/04-projects/070-cultural-software/67-article-2011.pdf

[2] Ibid.

Software ovlivňuje způsoby naší komunikace, tvorby a sdílení. Zdařilo se mu proniknout do všech kulturních vrstev a zásadně je změnit, v čemž samozřejmě stále pokračuje. A jak to shrnuje profesor Manovich, *„adding software to culture changes the identity of everything which a culture is made from.“*[3]

[3] MANOVICH, 2011.

# 2. Počítačová hudba

## 2.1. Počátky počítačové hudby - Illiac Suite

Počítače a hudba. Spojení, jež by ve čtyřicátých letech, v počátcích moderních počítačů, očekával asi málokdo. Natož konstruoval počítače přímo pro tento účel, pro něco tak nevědeckého, ba dokonce určeného k zábavě.
Avšak když se podíváme na dějiny hudby, zjistíme, že je její produkce je již od prehistorických dob spjata s nějakým nástrojem. Vždyť nejstaršímu nalezenému hudebnímu nástroji - flétně vyrobené ze supí kosti - je jejími nálezci připisováno stáří více než 40 000 let.[4] Pokud nástroje doprovází hudbu po desítky tisíc let, pak toto vlákno nemůže být jen tak přerušeno, a na počítače je pak třeba pohlížet jen na jako další kapitolu v tomto dlouhém vývoji.
Spojení hudby a matematických výpočtů je navíc zcela v souladu s filozofickým konceptem antického Řecka o *hudbě sfér.* Ten se, mimo jiné, zmiňuje o metafyzickém principu „ [...] *that mathematical relationships express qualities of tones of energy which manifest in numbers, visual angles, shapes and tones - all connected within a pattern of proportion."* [5]

O to, aby se počítače mohly změnit ze strojů určených téměř beze zbytku k řešení složitých matematických na stroje schopny tvořit i umění, se přičinil zejména americký elektronik a matematik *Claude E. Shannon*. Tento „otec teorie informace", jak bývá často nazýván, napsal v roce 1948 studii *A mathematical theory of information,* v níž vysvětluje, že počítačové systémy by dle něj mohly fungovat jako platforma pro digitální zpracování jakékoli myslitelné informace, od psaného jazyku po televizní vysílání. Tímto se mu podařilo rozšířit oblast počítačového výzkumu novým směrem, v němž již byl prostor i pro zpracování hudby.[6]
Zcela prvním počítačem, jež tento prostor využil, a z jehož chatrných reproduktorů tak lidé poprvé zaslechli digitálně zpracovanou hudbu, byl stroj vytvořený duem australských inženýrů Trevora Pearceyho a Mastona Bearda, zvaný *CSIRAC.* Tento vůbec první australský digitální počítač, pracující s děrnými pásky, přehrál v 1951 před užaslým publikem nahrávku britského vojenského pochodu *Colonel Bogey March.* Nahrávka se však nezachovala, a i kdyby ano, šlo vlastně o pouhou reprodukci bez ambicí jakkoliv rozšířit pole působnosti hudební vědy i praxe.

---

[4] OWEN, James. Bone Flute Is Oldest Instrument, Study Says. *National Geographic News.* [online] cit. 29. 4.. 2016 Dostupné z: http://news.nationalgeographic.com/news/2009/06/090624-bone-flute-oldest-instrument.html

[5] WEISS, Piero; TARUSKIN RIchard. *Music in the Western World: a history in documents.* Cengage Learning, 2008. op. cit. s. 3. ISBN: 978-0-534-58599-0.

[6] SHANNON, Claude. A Mathematical Theory of Communication. *The Bell System Technical Journal,* 1948.s. 379–423. Dostupné z: http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf

O první výtvor splňující tato kritéria se tak na podzim 1951 postaral počítač *Ferranti Mark 1*, komerční verze prvního programovatelného počítače známého jako *Baby machine,* stvořeného na Victoria University of Manchester. Pomocí programu od počítačového vědce Christophera Stracheyho dokázal počítač přehrát své verze klasické britské říkanky *Baa Baa Black Sheep*, poté *God Save the King* a nakonec zkrácenou verzi slavného díla Glenna Millera *In the Mood*. A díky tehdy přítomného štábu z BBC máme navíc tyto první počítačem přehrané skladby navždy zachovány.[7,8]

**Illiac Suite**

I když už byly počítače schopny svými vlastními způsoby přehrávat hudbu, stále nešlo o hudbu počítačem vytvořenou. K tomu se však ještě v padesátých letech podařilo přiblížit skladatelům působícím na *University of Illinois*, jednom z center tehdejšího výzkumu počítačové hudby, Lejarenu A. Hillerovi a Leonardu Isaacsonovi, když pomocí počítače poprvé vygenerovali hudební data, která následně přepsali do partitury. Tato skladba s názvem *Illiac Suite for String Quartet* vznikla v roce 1957 a jméno dostala podle superpočítače, jež ji pomohl vytvořit - Illinois Automatic Computer, zkráceně ILLIAC.

Jak sami tvůrci uvádějí ve své knize *Experimental Music,* nešlo jim o vytvoření nějaké přelomové monumentální skladby, spíše výzkum pojali jako experiment. Samotná píseň se pak skládá ze čtyř různých experimentů, které je z hlediska hudební terminologie možno označit jako hudební věty.[9]
První věta obsahuje *cantus firmus,* tedy princip raného vícehlasu, druhá pak generuje čtyřhlasé části, každou s rozdílnými pravidly. Třetí se zabývá rytmem po vzoru tehdy moderních kompozičních technik (např. serialismem), dynamikou a instrukcemi pro hraní, a konečně poslední věta pracuje s různými modely a pravděpodobnostmi pro generativní gramatiku a Markovovy řetězce, tedy se stochastickými technikami.[10]

Nemůžu nezmínit fakt, že ač jde podle tvůrců o experiment, Illiac Suite jej až tak úplně nesplňuje. Experiment je obecně definován jako soubor jednání a pozorování, jehož účelem je vyvrátit nebo ověřit hypotézu či poznatek. Když se podíváme na první dvě věty skladby, můžeme stěží říci, že by obsahovaly něco nového a originálního, třetí věta pak už vůbec ne; pouze pracuje s v tehdejší době již známými hudebními styly. Není tedy snadné říci, co vlastně tento počin představuje, když nesplňuje parametry experimentu, a jako o hudebním díle o

---

[7] FILDES, Jonathan. 'Oldest' computer music unveiled. *BBC News.* [online] cit. 29. 4. 2016 Dostupné z: http://news.bbc.co.uk/2/hi/technology/7458479.stm

[8] First Recorded Computer Music. *YouTube.* video [online] Dostupné z: https://www.youtube.com/watch?v=xLwjz0UR5_A

[9] DI NUNZIO, Alex. Illiac Suite. *Musica informatica.* [online] cit. 29. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/illiac-suite.php

[10] HILLER, Lejaren; ISAACSON Leonard M. *Experimental Music: Composition With an Electronic Computer.* McGraw-Hill, New York, 1959, 2. vyd. s. 5-7. ISBN: 978-0-313-22158-3.

něm odmítají mluvit i sami jeho tvůrci. O čem však není pochyb, je fakt, že jde o dílo, které skvěle reprezentuje techniky hudební kompozice konkrétní doby, a je tak důležitým mezníkem ve vývoji počítačové hudby.

## 2.2. Počátky zvukové syntézy - Max Mathews

Zatímco na University of Illinois sklízeli obdiv pánové Hiller a Isaacson, na východním pobřeží Spojených států se o velký pokrok v symbióze mezi počítači a hudbou postaral Max Mathews. Jako zaměstnanec Bellových laboratoří zde napsal první program na světě umožňující přes přímou syntézu generovat digitální zvukové vlny, lapidárně pojmenovaný MUSIC.
Jeho prostřednictvím přehrál v roce 1957 na sálovém počítači IBM 704 asi 17 vteřin notového záznamu, jehož tón *„byl podobný zvuku hodinek nebo mobilního telefonu."*[11] Vzhledem k tehdejším kapacitám počítačů šlo o úctyhodný počin, neboť ke zpracování pár tónů bylo zapotřebí několika hodin. Náročnost i kvalita tónu zde však hlavní roli nehráli, neboť jak vzpomíná sám Mathews: *„The timbres and notes were not inspiring, but the technical breakthrough is still reverberating."*[12]
Především šlo totiž o dokázání možnosti digitalizace zvuku a jeho následné uložení, což se podařilo dokázat dokonale. Tento úspěch inspiroval Mathewse k vytvoření dalších programů z řady *music n,* do nichž postupně přidával další části a funkce.

Ani tímto však jeho role v počítačové hudbě nekončí, ba naopak, po celý zbytek 20. století zůstává jedním z hlavních leaderů v této oblasti. Za příklad jeho tvůrčích schopností může sloužit program ze sedmdesátých let *Groove (*zkratka skrývající slova *Real-time Generated Operations On Voltage-controlled Equipment),* vytvořený ve spolupráci s kolegou Richardem Moorem. Jejich počin byl prvním zcela vyvinutým systémem hudební syntézy pro interaktivní kompozici a *real-time* performance, a stal se rychle jedním z nejpopulárnějších hybridních systémů určených pro syntézu zvuku a její ovládání v reálném čase.[13]

---

[11] FLAŠAR, Martin. *Elektroakustická hudba.* Masarykova Univerza, Brno, 2015. op. cit. s. 81. Dostupné z: http://is.muni.cz/do/rect/el/estud/ff/ps15/eah/web/media/Elektroakusticka_hudba.pdf

[12] Remembering Max - A tribute. *cSound.* [online] cit. 29. 4. 2016 Dostupné z: http://www.csounds.com/mathews/max_ideas.html

[13] DI NUNZIO, Alex. Groove. *Musica informatica.* [online] cit. 29. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/groove.php

# 3. Algoritmická kompozice

## 3.1. Úvod

Live coding se dá, kromě několik dalších definic, vysvětlit jako spojení dvou faktorů - lidského a technického. Lidský zde zastupuje schopnost improvizace, ten technický pak počítačové umění algoritmické kompozice.
Pojem *algoritmická kompozice*, někdy je také užíváno názvu *zautomatizovaná kompozice*, značí „ [...] *process of using some formal process to make music with minimal music intervention."* [14]
Tento proces je nejčastěji tvořen na počítači pomocí různých *formalismů*, jako náhodné generace čísel, *rule-based* systémů a dalších různých algoritmů.

## 3.2. Algoritmická kompozice bez počítače

Nejprostší forma algoritmické kompozice se objevuje již v dobách, kdy by ji asi každý neočekával, a to v roce 1787, kdy Wolgang Amadeus Mozart nedokončuje pouze Dona Giovanniho, ale také návod na speciální hudební hru zvanou *Musikalisches Würfelspiel,* neboli „hudební hru v kostky".[15] Myšlenka hry spočívá v sestavování řady krátkých hudebních fragmentů a jejich náhodném spojování dohromady pomocí čísel vržených na kostce, čímž samozřejmě pokaždé vznikne zcela nové dílo.

U principu náhody zůstaneme, neboť věrni tomuto nejzákladnějšímu a zároveň i nejjednoduššímu algoritmickému principu zůstali i další experimentátoři ve vodách zautomatizované kompozice. V čase se však posuneme do nám již bližších dob, přesněji do roku 1968, kdy se slavný americký skladatel *John Cage*, ve spolupráci s dalšími hudebníky, pouští do práce na skladbě, či přesněji už performance, *Reunion.* Ta využívá ke spouštění hudby hraní šachu na speciální šachovnici vybavené foto receptorem. Pohyby figurkami spouští různé předem vytvořené hudební fragmenty a zvuky, díky čemuž je dílo při každém provedení jiné.

---

[14] ALPERN, Adam. *Techniques for algoritmic composition of music.* Hampshire College, 1995. s. 1 cit. 30. 4. 2016 Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9364&rep=rep1&type=pdf

[15] CHUANG, John. Mozart's Musikalisches Würfelspiel. *SunSITE Austria at the University of Vienna.* [online] cit. 29. 4. 2016 Dostupné z: http://sunsite.univie.ac.at/Mozart/dice/

## 3.3. Algoritmická kompozice s užitím počítače

Ač to bude nejspíše překvapivé, musím se na úvodu počítačem podporované algoritmické kompozice vrátit až do první polovině 19. století, a to k anglické matematičce Adě Lovelace, známé především díky jejímu detailnímu popisu fungování *analytického stroje* Charlese Babbage. Naši pozornost si však zasluhuje především díky tomu, že nejspíše jako první člověk na světě rozpoznala potenciál budoucích počítačů na jiném poli, než je jen zpracování čísel. Ve svých poznámkách z roku 1843 totiž píše:

*„Supposing, for instance, that the fundamental relations of pitched sound in the signs of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."* [16]

Jak hraběnka z Lovelace předpověděla, tak sestalo. Počítače svedly dohromady vědce a hudebníky, aby ti následně mohli tvořit ony *„elaborate and scientific pieces of music".*

O jedněch z hlavních pionýrů počítačové kompozice jsem se již zmínil výše, byli to totiž právě pánové Hiller a Isaacson s jejich slavnou Illiac Suite.[17] Lejaren Hiller se však do dějin počítačové hudby nezapsal pouze tímto dílem; zmínku si zaslouží i výsledek jeho spolupráce s kolegou Robertem Bakerem z přelomu padesátých a šedesátých let. Výsledek jejich snažení dostal jméno *MUSICOMP (MUsic Simulator Interpreter for COMpositional Procedures),* a stal se jedním z prvních počítačových systémů pro zautomatizovanou kompozici.[18] Díky tomu, že byl program koncipován jako knihovna *podprogramů*[19]*,* proces psaní programů na hudební kompozici se značně zjednodušil; skladatel/programátor totiž tuto knihovnu mohl použít v jakémkoliv dalším programu, který mu pro jeho tvorbu nejvíce vyhovoval.

Tento přístup Hiller využil v následujících letech, kdy zorganizoval několik svých různých prací do jedné knihovny kompozičních formalismů. Tato myšlenka vytváření malých, jasně definovaných kompozičních funkcí (v tomto případě podprogramů) a jejich následného sestavování dohromady se ukázala jako velmi efektivní a dovolující systému značnou flexibilitu, což z ní udělalo velmi oblíbený přístup, dodnes používaný v mnoha systémech pro algoritmickou kompozici.

Na tomto místě se ještě opět zmíním o Johnu Cageovi, který již začíná experimentovat i s počítači, objevuje při tom potenciál Hillerových programů, a rovnou se s ním pouští do spolupráce, z níž vzejde slavné dílo *HPSCHD,* skladba pro *„7 cembal hrajících náhodně zpracovanou hudbu Mozarta a dalších*

---

[16] ALPERN, 1995, s. 2

[17] viz kapitola 2.1. Počátky počítačové hudby - Illiac Suite. s. 9-10

[18] DI NUNZIO, Alex. Musicomp. *Musica informatica.* [online] cit. 30. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/musicomp.php

[19] Podprogram - v programování jde o část programu, kterou je možné z různých míst kódu opakovaně volat. pozn. aut.

*skladatelů, 51 audiokazet s nahrávkami počítačem generovaných zvuků, více než 5 000 dia pozitivů s motivy vesmíru a abstraktními tvary, a několik filmů".*[20]

Dalším, a pravděpodobně i nejznámějším, jménem v oblasti užívání počítačů k tvorbě hudby je řecko-francouzský skladatel *Iannis Xenakis*. Užíval vysokorychlostních výpočtů počítače k propočítávání různých pravděpodobnostních teorií. S pomocí svých programů následně tvořil kompozice notových zápisu pro celý živý ansámbl, využívajíce při tom statistických a pravděpodobnostních metod. Jeho program fungující na *stochastickém* principu tvořil notové zápisy na základě seznamu hustoty not a pravděpodobnostních hodnot, dodaných uživatelem, nechávajíce při tom určitá rozhodnutí zcela na generátoru náhodných čísel.

Xenakis sám sebe co by skladatele kdysi popsal téměř jako hrdinu dobové sci-fi: *„With the aid of electronic computers, the composer becomes a sort of pilot: he presses buttons, introduces coordinates, and supervises the controls of a cosmic vessel sailing in the space of sound* [...]*."* [21]

Vždy si přál ovládat hudbu pomocí jazyka vzešlého z matematiky a logiky. A možná právě tato snaha o „vyšší estetiku" je důvodem, proč byla jeho práce, ač fascinující, vždy více zaměřená na objevování procesu kompozice, než na výsledek samotný.

Zmínil jsem, že Xenakis využíval *stochastického principu*. Společně s *rule-based* systémem, tedy systémem založeným na pravidlech, tvoří dvě základní metodologie tvorby počítačově generované algoritmické kompozice.
*Stochastický princip* je termín půjčený z matematiky, jež byla ostatně Xenakisovi, coby původně vystudovanému architektovi, velmi blízká, a označuje proces *„in which a sequence of values is drawn from a corresponding sequence of jointly distributed random variables".*[22] Jde tedy o princip pracující s *náhodností,* spadá sem například prosté generování náhodných sérií not, jaké je k vidění u Mozartovy *Musikalisches Würfelspiel* nebo u Johna Cage. Může se však stát i složitějším principem, to v případě využití statistické teorie a Markovových řetězců.

Druhá hlavní metodologie - *systémy založené na pravidlech* - se soustřeďuje na různé testy a pravidla, dle nichž program postupně hudbu vytváří. Tyto kroky jsou nejčastěji stvořeny do takové podoby, že teprve splnění jedné podmínky či pravidla vede k možnosti splnit další. Asi nejznámějším algoritmickým umělcem často využívajícím těchto metod byl Lejaren Hiller. Liší se tak třeba od Xenakise, jenž, ač experimentoval i s tímto přístupem, dával přednost principu stochastickému. Jejich názory se navíc nestřetávaly pouze v otázce používání hudebních programů, nýbrž i v celkovém pohledu na jejich díla.

---

[20] DI NUNZIO, Alex. HPSCHD. *Musica informatica.* [online] cit. 30. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/hpschd.php

[21] XENAKIS, Iannis. *Formalized Music: Thought and Mathematics in Composition.* Pendragon Press, 1994. s. 144. ISBN: 2234015103

[22] Stochastic. *Your Dictionary.* [online] cit. 30. 4. 2016 Dostupné z: *http://www.yourdictionary.com/stochastic*

Zatímco řecký avantgarda věřil počítačům natolik, že jeho výstupy nijak neupravoval a sám se nechal slyšet, že 75% celé jeho tvorby má na starosti počítač, a teprve zbytek jeho vlastní prací, Hiller naopak říkal, že pokud se tvůrci zdá algoritmický výstup jakkoliv nedokonalý, je třeba program pozměňovat, dokud nebude výstup přesně podle umělcových představ.[23]

## 3.4. Současné techniky algoritmické kompozice

Techniky druhé poloviny 20. století nemají tendenci být ani čistě stochastickými, ani založenými na pravidlech, spíše se stávají jejich hybridy, a navíc je kombinují s dalšími prvky, například z teorie umělé inteligence, komplexního systému, dokonce i umělého života.

### Model generativní gramatiky

S generativní gramatikou byl v algoritmické kompozici vytvořen mocný systém formálních pravidel pro generování hudebních struktur. Původně vyvinut v kontextu lingvistiky, dovoluje produkci struktur, jež jsou i v dlouhých pasážích citlivé na kontext. Dle hierarchie vytvořené Noamem Chomskym, otcem teorie generativní gramatiky, stojí generativní gramatika na poli algoritmické kompozice vysoko nad všemi ostatními technikami.

Tento přístup rozlišuje gramatické typy různé výrazovosti - ty s vyšší výrazovostí je snadnější zpracovat, avšak ve vztahu k těm nižším mají zase horší generativní kapacitu. Gramatický závěr procesu umožňuje automatickou generaci přepisování pravidel z *těla* - základní vrstvy, což znamená, že mohou být v jednom okamžiku zahrnuty různé hloubky souvislosti hudebního materiálu.[24]

### Transition networks

Neboli model přechodných sítí může být v algoritmické kompozici aplikován na celou řadu úkolů. Tento formalismus pro reprezentování a zpracování hudebních informací použil například americký skladatel a vědec David Cope pro svůj program *EMI*. Ten generuje imitace hudebního stylu poté, co analyzuje dostatečný počet existujících skladeb konkrétního žánru. Copův systém aplikuje komplexní strategii v rekombinaci hudebních segmentů, zvažujíce při tom jejich formální význam na různých úrovních hierarchie.

---

[23] ROADS, Curtis. *The Computer Music Tutorial.* MIT Press, 1996. s. 89. ISBN: 978-0262680820

[24] NIERHAUS, Gerhard. *Algorithmic Composition: Paradigms of Automated Music Generation.* SpringerWienNewYork, 2009. [verze Kindle] s.4. Získáno z Amazon.com

Přechodné sítě existují v různých formách, a zohledňují adaptaci k rozdílným hudebním úkolům.[25]

**Genetické algoritmy** a genetické programování jsou vyhledávací techniky založené na pravděpodobnosti a vytvořené na základě modelu Darwinovy evoluční teorie. Úkoly jsou zde řešeny ve virtuálním biologickém prostředí aplikováním kvazi-biologických procedur. Genetické operace jako křížení nebo mutace produkují v každé generaci nové jedince, kteří jsou následně ohodnoceni podle kvalitativního měřítka, jehož parametry musí splnit. Specifické chování genetických algoritmů podporuje zejména ty kompoziční přístupy, jež tvoří hudební materiály jako je *time-flow*, tedy proces neustálé změny.

**Samoučící se systémy** jsou programy, které nemají žádné předem naprogramované znalosti žánru hudby, s níž pracují. Místo toho sbírají informace sami ze vzorových materiálů dodaných uživatelem či jejich programátorem. Výsledný materiál je poté zpracován do hudební skladby podobné dodaným vzorům. Tato metoda algoritmické kompozice je silně propojena s algoritmickým modelováním stylu, improvizací strojů, a také studiemi o kognitivních vědách a neutronových sítích.

**Hybridní systémy** začaly vznikat z toho důvodu, že programům založeným na jednom algoritmickém modelu se podaří zřídkakdy dosáhnout esteticky uspokojivých výsledků. Proto se v současné době často používá několik algoritmů různých typů dohromady, s cílem ideálně nakombinovat silné stránky každého z nich, a minimalizovat ty slabé.[26]

---

[25] NIERHAUS, s.5.

[26] FERNÁNDEZ, José David; VICO Francisco. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research.* č. 48, 2013. s. 528- 532.

# 4. Live coding

## 4.1. Úvod

Live coding, někdy také nazýván jako *programování za chodu* nebo *programování právě včas*, je kreativní technika založená na improvizovaném *interaktivním programování*.[27] Pod pojmem 'interaktivní programování' se rozumí proces psaní částí programu, který je už zároveň v chodu. Hlavním uživatelským rozhraním se tak stává samotný psaný kód programu, čímž se proces odlišuje od *interaktivního zpracování,* kdy je program navržen ve vývojových cyklech a použit až poté. Dá se tedy říci, že use už přímo akt psaní programu stává součástí samotného programu.
V protipólu k interaktivnímu programování pak stojí *batch processing (dávkové zpracování),* kde se neděje interaktivní cestou ani psaní programu, ani jeho užívání.[28]

V rámci umění spadá live coding do performativního umění, tj. umění, při nichž vystupují před publikem sami umělci. Samozřejmě, že v tomto případě není umělcovým médiem jeho hlas či tělo, jak tomu bývá zvykem, nýbrž jeho schopnost improvizace, která je v tomto případě stěžejním procesem celého vystoupení. Díky ní se z tato performance stává pomíjivou, působící pouze v daném okamžiku na daném místě.
Live coding se nedá snadno zařadit do určitého uměleckého směru, naopak si bere inspiraci z mnoha zaráz, ať už z konceptuálního umění a serialismu, nebo z music concrète, minimalismu, noisu i elektronické taneční hudby. Stejně tak osoba *live codera* se pohybuje mezi několika termíny zároveň a dala by se popsat jako kombinace performera, programátora a skladatele.

Jak o umění živého kódování přemýšlejí sami jeho tvůrci, dobře ilustruje výňatek některých hesel z manifestu organizace TOPLAP, sdružující umělce zabývající se právě tímto algoritmickým uměním:

- *Live coding může být doprovázen působivým zobrazením manuální zručnosti a oslavováním psacího rozhraní.*
- *Kód má být viděn i slyšen, základní algoritmy zobrazeny stejně jako jejich vizuální výstup.*
- *Live coding není založen na nástrojích. Motorové pily jsou nástroje. Algoritmy jsou myšlenky. Z toho důvodu je někdy obtížnější si všimnout algoritmů, než motorových pil.*

---

[27] SORENSEN, Andrew; BROWN, Andrew R. aa-cell In Practice: An Approach To Musical Live Coding. *Proceedings of the International Computer Music Conference.* Kodaň, Dánsko, srpen 2007. Dostupné z: http://impromptu.moso.com.au/extras/aa-cell-icmc07.pdf

[28] Batch processing. *PC Mag.* [online] cit. 30. 4. 2016 Dostupné z: http://www.pcmag.com/encyclopedia/term/38460/batch-processing

- *Tmářství je nebezpečné. Ukažte nám vaše obrazovky.*
- *Dejte nám přístup k umělcově mysli, dáte nám přístup k celému lidskému nástroji.*
- *Programy jsou nástroje, jež se mohou sami měnit.*
- *Nepoužíváme zálohy.* [29]

## 4.2. Historie živého kódování

Ač je live coding přirozeně zcela spjat s počítači, bez nichž by tento proces nebyl realizovatelný, základní principy v něm využívané lze najít již v dávnějších experimentech nevyužívajících jakékoli elektronické zařízení, natož počítač.
Takovéto rané období sahá až do dob antického Řecka, kde je možno si všimnout jistých prvků dnešního *kódování naživo* v tehdejších debatách filozofů. Samozřejmě, že nemůže být řeč o algoritmech či hudbě, avšak jakousi vzdálenou podobnost lze najít mezi dnešním psaním kódu a tehdejším vytváření názorů a myšlenek, které se rovněž dělo „za chodu".

Dalším předchůdcem z dob před jakýmikoliv interpretovanými jazyky mohou být *„veřejné matematické turnaje, které jsou známé již ve starověkém Egyptě, ale svého největšího rozmachu dosáhly během renesance."* [30] I zde jde však samozřejmě o pouhé náznaky základních technik živého kódování.

Prvním dílem, které obsahuje skutečné prvky *live codingu*, se stala v roce 1975 skladba švédského experimentálního hudebníka Clicka Nilsona. Ač je jeho dílo s názvem *An Instructional Game for 1 to many musicians* stále „nepočítačové", pracuje již na principu za chodu vytvářeného algoritmu. Hra spočívala v tom, že každý hudebník na začátku obdržel místo klasické partitury seznam instrukcí, jež si interpretoval dle svého, a navíc je mohl v průběhu představení kdykoliv upravovat. V seznamu bylo celkem šest možných změn, jež šlo aplikovat každou minutu - *změna, smazání, výměna, rozšíření povinností, sonifikace* a *ukončení.*[31]

Některým skupinám pokročilých live coderů se podařilo hru vymodelovat i v počítači, díky čemuž Nilsonův odkaz stále žije. Hra nemá žádné časové omezení, proto se mezi příznivci live codingu říká, že sám Nilson, poté co ztratil intrukce 1 a 6 (změna, ukončení), stále svou hru někde hraje.[32]

Dostáváme se do 80. let, v oblasti kódování známých jako *éra jazyku FORTH.* Prostředí tohoto programovacího jazyku, jenž vytvořil již v roce 1969 Charles H.

---

[29] ManifestoDraft. *TOPLAP.* [online] cit. 30. 4. 2016 Dostupné z: http://toplap.org/wiki/ManifestoDraft

[30] FLOREŠ, Ivan. *Pure Data: Nástroj pro algoritmické umění.* Bak. dip. práce, Masarykova Univerzita, Brno, 2011. s. 17. Dostupné také z: http://is.muni.cz/th/256351/ff_b/pure_data_ivan_flores_2011.pdf

[31] V originále *Change, Delete, Exchange, Proliferate Duties, Sonification, Termination.* pozn. aut.

[32] HistoricalPerformances. *TOPLAP.* [online] cit. 1. 5. 2016 Dostupné z: http://toplap.org/wiki/HistoricalPerformances

Moore, umožňovalo spouštět více procesů zaráz, a současně v něm mohlo nezávisle na sobě pracovat až 64 uživatelů. Na základě FORTH později vzniká objektově orientovaný hudební jazyk *HMSL - Hierarchical Music Specification Language,* následně široce užíván pro živou kompozici a performance.[33]

Přestože profesor hudby na New Yorské *Columbia University* George Lewis tvrdí, že se živě programovalo již od počátku 80. let,[34] dělo se tak nejspíš jen za zavřenými dveřmi experimentátorských prostor, a nemáme o něm tedy žádné dochované informace, natož třeba záznamy. Proto se dnes za skutečně první plnohodnotnou live coding performance považuje ta, kterou předvedl v roce 1985 americký umělec Ron Kuivila. Stalo se tak na festivalu organizace STEIM - Centra pro výzkum a vývoj nových hudebních nástrojů - v Amsterdamu. Předvedená skladba dostala název *Water Surface,* a jak vzpomíná na místě tehdy přitomný hudební teoretik Curtis Roads, předtím, než stihla na stage zaznít nějaká zajímavá hudba, se Kuivilův software pokazil. Celá performance tak sestávala pouze ze sledování tvoření kódu.[35]

Ve stejném roce vzniká komunita live codingu také v samotném rodišti moderních počítačů - kalifornském Silicon Valley. Ze staršího hudebního uskupení *The League of Automatic Music Composers* se vyčleňují dva jeho členové John Bischoff a Tim Perkis, a již pod názvem *The Hub* se účastní mini festivalu *THE NETWORK MUSE - Automatic Music Band Festival.* Slovo *hub* (střed, centrum) odkazuje k přímo k jejich technice; používali totiž malého počítače jako 'poštovní schránky', do níž posílali data používaná k ovládání jejich vlastního hudebního systému. Díky schránce byla data ihned přístupná dalšímu uživateli, který s nimi mohl jakkoliv a kdykoliv nakládat.[36] Dodám ještě, že jejich systém pracoval na základě již zmíněné platformy FORTH.

Na přelomu let 80. a 90. na scénu živého kódování pronikají systémy s grafickým programovacím prostředím, které nebylo všemi umělci úplně uznáváno, neboť nejde o plnohodnotný programovací jazyk, ale o jazyk vizuální, kde k hudební produkci dochází prostřednictvím manipulace 'patchů'. Za nejznámější a zároveň nejpoužívanější software této éry se dá považovat dílo Millera Pucketta *MAX/MSP.*[37]

Zlatá éra tohoto umění přichází až s přelomem milénia. Přesněji přímo kulatý rok 2000 byl pro live coding i další způsoby tvorby počítačové hudby přelomový. Výkony osobních počítačů strmě stoupají, jejich ceny i rozměry naopak kvapem

---

[33] HMSL - Heirarchical Music Specification Language. *SoftSynth.* [online] cit. 1. 5. 2016 Dostupné z: http://www.softsynth.com/hmsl/

[34] HistoricalPerformances. *TOPLAP.* [online] cit. 1. 5. 2016 Dostupné z: http://toplap.org/wiki/HistoricalPerformances

[35] WARD, Adrian; ROHRHUBER Julian; a kol. A Historical Perspective. Live Algorithm Programming And A Temporary Organisation For Its Promotion. *Runme.org* [online] cit. 1. 5. 2016 Dostupné z: http://art.runme.org/1107861145-2780-0/livecoding.pdf

[36] BROWN, Chris. The first Hub. *Crossfade - Walker Art Center.* [online] cit. 1. 5. 2016 Dostupné z: http://crossfade.walkerart.org/brownbischoff/

[37] HistoricalPerformances. *TOPLAP.* [online] cit. 1. 5. 2016 Dostupné z: http://toplap.org/wiki/HistoricalPerformances

klesají, tvorba počítačové hudby se tak stává dostupnou pro velké množství uživatelů.

## 4.3. The Slub

V červnu téhož roku navíc zakládají britští hudebníci Alex McLean a Adrian Ward live codingové uskupení *Slub*. Již po prvním vystoupení ve známém londýnském klubu Foundry se skupina ihned stává stěžejním subjektem live codingové scény, neboť tvoří hudbu výhradně pomocí jejich vlastního generativního softwaru, jehož prostředí je při představení promítáno na zeď, takže může hudebníkům každý „koukat pod prsty" a performance je tak divácky mnohem přitažlivější. Jejich zvuk se pohybuje od klasického Detroit techna až po meditativní akustické studie. Ať však hrají cokoliv, společným jmenovatelem jejich akcí je vysoká kvalita a osobitost, zaručená výraznou kontrolou nad jejich vlastním softwarem.

Zatímco v počátcích se v jejich představení vyskytovaly i prvky nenáležící vyloženě do live codingu, od roku 2005, kdy se k uskupení přidává ještě Dave Griffiths, jsou jejich performance výhradně live codovými. Využívají při tom celou škálu jejich vlastních jazykových prosředí, mezi něž patří např. *Pure Events*, JavaScriptové prostředí podobné trackeru[38]; *feedback.pl*, prostředí pro automatickou editaci programování v jazyku Perl; *Fluxus*, herní engine jazyku Scheme; a *Tidal*, čistě funkční DSL jazyk zasazený do jazyku Haskell.

Jak jsem již zmínil, všechna jejich představení jsou doprovázena promítáním programovacího prostředí, což nabádá k myšlence, že představení jsou tím pádem určena primárně pro odborníky, kteří zobrazenému kódu rozumí. Třetí člen Slubu, Dave Griffiths, k tomu však říká, že: *„It's not important that the audience understand the code. It's to show them that you're really doing something."* [39]

A že všichni tři skutečně 'něco dělají', dokazuje nejen zmíněný fakt, že si píšou své vlastní programy, nýbrž i neustálé upravování a vylaďování jejich kódu, experimentování s novými programovacími jazyky, a celkové improvizování se softwarem, stejné jako s hudbou. Sám Adrian Ward se ohledně jejich novátorství nechal slyšet, že je zvyklý svůj aktuální zdrojový kód vyhazovat každých dvanáct měsíců.

Díky Slubu byla live codingová a hlavně klubová scéna obohacena o další styl hudebního koncertu - *algorave.* Výraz složený ze slov „algorithmic" a „rave" vymysleli v roce 2011 Nick Collins a Alex McLean, když, dle svých slov, zrovna mířili na svůj další koncert v Nottinghamu.

---

[38] Tracker - hudební program používaný pro vytváření hudebních skladeb. pozn. aut.

[39] ARMITAGE, Tom. Slub: Making music with live computer code. *Wired.co.uk* [online] cit. 24. 9. 2009 Dostupné z: http://www.wired.co.uk/news/archive/2009-09/25/making-music-with-live-computer-code-.aspx

*„We tuned into a pirate station playing happy hardcore, and we thought it would be good to program some rave music,"* dodává Collins.[40] Od té doby začali tyto akce, jejichž základní ideou je předvádět „živé kódování repetitivních akustických algoritmů v temných místnostech"[41], pořádat po celém světě. Častými lokacemi událostí jsou třeba německé Karlsruhe, Londýn nebo Mexico City. Plnou definici pojmu *algorave* si tvůrci vypůjčili z Britského trestního zákoníku z roku 1994, který obsahuje tzv. *anti-rave* zákon a *rave* je zde popsán jako „sounds wholly or predominantly characterised by the emission of a succession of repetitive conditionals".[42]

Při těchto vystoupeních se využívají takřka všechny programy, ve kterých je možné live coding provádět, ať už vizuální jako *PureData* a *MAX/MSP,* tak ty 'klasické', nevizuální jako *Super Collider, Impromptu* nebo *Fluxus.* Díky nim se k syrovým zvukům britského *raveu* z 90. let přidávají algoritmy vytvořené futuristické beaty a rytmy, a vzniká tak zcela nový a jistě stále neotřelý zvuk.

## 4.4. Další live codingová uskupení a umělci

Mezi jednu z nejvýznamnějších postav světa live codingu patří *Julian Rohrhuber*. Po vystudování oboru vizuální komunikace na Univerzitě výtvarných umění v Hamburgu pracoval třeba na výzkumu týkajícím se algoritmické akustiky nebo přednášel o počítačových vědách na univerzitě v Kolíně a hudební informatice Univerzitě hudby v Karlsruhe. Od roku 2009 je členem vědecké skupiny zabývajícími se médii *Situierte Medien* v Siegenu, a v současné době je profesorem hudební informatiky a teorie médií na Univerzitě hudby a médií Roberta Schumanna v Düsseldorfu.[43]

Do povědomí live coderů se zapsal především jako jeden z hlavních vývojářů open-source prostředí SuperCollider. Od roku 2000 začal v rámci uměleckých i vědeckých projektů programovat algoritmy pro zvukovou syntézu. Zabývá se také vyvíjením nových metod konverzačního programování a síťové hudby jako je např. *JITLib (Just In Time Library).* Live codingem se přirozeně zabývá i aktivně, společně s pěti dalšími kolegy z univerzity v Karlsruhe tvoří uskupení *Powerbooks Unplugged.* Jejich vystoupení se dají popsat jako minimalistická,

---

[40] CHESHIRE, Tom. Hacking meets clubbing with the 'algorave'. *Wired.co.uk* [online] 23. 8. 2013 Dostupné z: http://www.wired.co.uk/magazine/archive/2013/09/play/algorave

[41] McLEAN, Alex. Algoravers. *TOPLAP.* [online] 2. 11. 2012 Dostupné z: http://toplap.org/algoravers/

[42] Odst. 1b. § 63. Powers to remove persons attending or preparing for a rave. Část V. Powers in relations to raves. Criminal Justice and Public Order Act 1994. *Legislation.gov.uk.* Dostupné také z: http://www.legislation.gov.uk/ukpga/1994/33/part/V/crossheading/powers-in-relation-to-raves#term-music

[43] Prof. Julian Rohrhuber. *Institute For Music And Media.* [online] cit. 3. 5. 2016 http://musikundmedien.net/en/studies/faculty/julian-rohrhuber/

neboť všechen vzájemný přenos kódu probíhá přes wi-fi a jediným zdrojem výsledného zvuku jsou pouze reproduktory jejich notebooků.[44]

Důležitou osobností živého kódování je také počítačový vědec a umělec, skladatel i aktivní performer elektronické hudby *Andrew Sorensen*. V současnosti působí jako výzkumník na Queensland University of Technology v australském Brisbane. Jeho zájmy leží na pomezí počítačových věd kreativní praxe, zabývá se totiž vývojem i praktickým aplikováním programovacích jazyků v počítačovém umění. Mezi jeho největší počiny zde patří vytvoření programů *Impromptu* a příbuzného *Extempore.* Sorensen své počítačové práce také rád vytváří před živým publikem, vyvíjejíce přitom software jako nedílnou součást představení. Bývá častým hostem live codingových session nejen v Austrálii, ale také v německém Karlsruhe nebo Londýně.[45] [46]

Již několikrát zmíněné Karlsruhe je vyloženým epicentrem současného live codingu. V roce 2009 v něm vzniklo live codingové uskupení *Benoît and the Mandelbrots*, pojmenované po francouzsko-americkém matematikovi Benoît Mandelbrotovi, jenž se proslavil zejména jako zakladatel fraktální geometrie. Její členové Juan Romero, Holger Ballweg, Patrick Borgeat a Matthias Schneiderbanger se potkali během studií na Hochschule für Musik Karlsruhe, a jsou známí díky svému improvizování s programova- cími jazyky a *algorave* koncertům. Na těch používají oblíbený SuperCollider a výsledný zvuk se řadí k tvrdším formám hudby a pohybuje se někde mezi technem a noisem. V roce 2012 jejich činnost ocenil i prestižní institut nových médií Ars Electronica v Linzi a na počátku letošního roku na CD i gramofonových deskách vydali své první album, prostě pojmenované Benoît and the Mandelbrots.

## 4.5. TOPLAP

Už dříve jsem se zmínil o organizaci *TOPLAP*. Její vznik má na svědomí především Alex McLean a jak sám detailně popisuje, zrodila se prý v jednu hodinu ráno v neděli 15. února 2004 v jednom zakouřeném hamburském baru. Název je akronym s mnoha možnými výklady, z nichž nejčastěji užívaným je „Temporary Organisation for Proliferation of Live Algorithm Programming" a jako

[44] Powebooks Unplugged. ZXZW independent culture. [online] cit. 3. 5. 2016 Dostupné z: http://past.incubate.org/2007/acts.php?id=119

[45] SCHUSTER, Werner. Andrew Sorensen on Real Time Programming, Live Coding Music, Memory Management. *InfoQ.* [online] 13. 12. 2014 Dostupné z: http://www.infoq.com/interviews/sorensen-extempore-realtime

[46] Andrew Sorensen. *Pioneers.* [online] cit. 1. 5. 2016 Dostupné z: https://pioneers.io/blog/people/andrew-sorensen/

hlavní cíle této organizace jsou uváděny propagace, podporování a výzkum live codingu.[47]

TOPLAPem vydaný „manifest live codingu", jehož některé body uvádím v kapitole 4.1. Úvod[48], nemá plnit funkci nějakých závazných pravidel, jimiž by se všichni codeři museli řídit. Snaží se pouze do světa živého kódování vnést pár zdravých a rozumných principů otevřenosti a kvality, a podpořit tím tak nové objevy v tomto mladém odvětví. Protože, jak dodávají jeho tvůrci, *„profundity of insight into code and representation is to be applauded."* [49]

Že se organizce soustředí i na praktický výzkum, dokazuje to, že pod její hlavičkou vniklo několik osobních vývojových prostředí. 'Osobních' nebo ještě lépe 'personalizovaných' proto, že byly většinou vytvořeny pro vlastní účely jediným programátorem. Patří mezi ně například *feedback.pl* Alexe McLeana, *The Thingee* Amy Alexander nebo *fluxus* Davida Griffithse.

## 4.6 Vystoupení

V předchozích kapitolách o historii a prvních uskupeních jsem stále detailně nevysvětlil, jak vlastně takové live codingové představení probíhá, co je k němu zapotřebí a jak má výsledek tohoto improvizovaného programování vypadat, což je nyní třeba napravit.

Stěžejním bodem živého kódování je *proces,* který při tomto vystoupení probíhá. Aby umělci jasně zdůraznili, že k postupnému budování hudebních struktur dochází až v průběhu performance, na jejím samém začátku vidí posluchači/diváci projekci obrazovky s prázdným vývojovým editorem. Z toho důvodu je počátek vystoupení také doprovázen tichem, v němž live coder teprve sestavuje základní stavební prvky kódu, jehož úplnost je vytvářena kontinuálně během show.

Vystoupení v reálném čase na umělci přirozeně vyžaduje vysokou míru koncentrace. Postupem času totiž narůstá složitost kódu, díky čemuž se může programátorova soustředěnost začít vytrácet, navíc je od něj vyžadováno, aby měl přehled o další řadě paralelně běžících procesů. Zmíněnou složitost či lépe komplexnost je třeba vyvážit i ve vzniknuvší hudbě, kde musí tvůrce dbát na udržení rovnováhy mezi simplicitou a komplexností. Výrazovými prostředky zjednodušujícími výslednou hudbu v živém kódování jsou *opakování* a *náhodnost,* tedy již rozebírané prosté stochastické metody. Komplexnost je možné nuancovat zvýšením stupně abstrakce při použití již předem napsaného kódu. Při užívání předem připravených knihoven (podprogramů) je však nutné pamatovat na to, že by mělo být vyvážené se zřetelem na dostatečnou transparentnost kódu. Ačkoli jsou tyto abstrakce v programování značně

---

[47] WARD, Adrian; ROHRHUBER Julian; a kol. A Historical Perspective. Live Algorithm Programming And A Temporary Organisation For Its Promotion. *Runme.org.* s. 246 [online] cit. 1. 5. 2016 Dostupné z: http://art.runme.org/1107861145-2780-0/livecoding.pdf

[48] viz strana 17

[49] viz [47] s. 246

významné co se efektivního zápisu kódu týče, 'odhání' programátora od tvorby digitálního zvuku s nízkou úrovní konstrukce, což má za následek zpřetrhání jeho přímého propojení s médiem kódu.

Jednou ze stěžejních, a také nejdiskutovanějších, záležitostí live codingového vystoupení je umělcova schopnost zvládat v průběhu vystoupení náhlé obměny kódu. Jejich příprava a následná realizace v textovém rozhraní je beze sporu náročným úkolem, poněvadž se hudební kompozice zpravidla vyvíjí s tím, jak je krok za krokem stavěn kód, jímž je generována.

Praktická vystoupení tak žádají perfektní znalost nástrojů, jinak by se mohl účinkující jen stěží plně zaměřit na estetickou hudební stránku improvizace. V případě výtečné znalosti daného programovacího jazyka se proces kódování může pro umělce změnit v intuitivní prostředek vyjadřující jeho muzikální obrazotvornost.

Perfektní zvládání programovacího jazyka je zcela nezbytné zejména při používání interpretovaného programovacího jazyka. Pokud se totiž tvůrci „podaří" do interpretovaného kódu vepsat byť jen jedinou chybu, hrozí, že spadne celý program a následně také ihned skončí celá hudební produkce. Z takového živého vystoupení tak programátor musí zcela vypustit jakékoli testování, dolaďování či dokonce opravování chyb.[50]

Protože by se tím pádem náročnost live codingových vystoupení stala extrémní, některá prostředí pro živé kódování tvůrcům malé zádrhely a překlepy tolerují. V případě práce s integrovaným vývojovým prostředí[51] tak může navíc umělec využít nástrojů jako automatického dokončování řádků kódu, kontroly skladby zápisu jazyka nebo používání fragmentů kódu coby hlavních stavebních jednotek celé algoritmické kompozice.

Nabízí se také možnost využívání předem hotových jednotek kódu, a to jak v podobě pouze jednotlivých funkcí, tak celých souborů v podobě knihoven. V tom případě však tvůrci zase hrozí snížení schopností pozměňování kódu, protože se vytrácí příležitost přímého spojení s kódem. Tohoto problému se blízce dotýká jeden ze základních požadavků live codingu - potřeba *transparentnosti* zobrazovaného kódu. Místo toho, aby uživatel vyvolával funkci pomocí pouhé reference jejího skutečného názvu, je žádoucí s ní pracovat pomocí jejího kompletního rozvrhu. S tímto přístupem je totiž k následnému upravování poskytnut celé její *tělo,* což je i přes spojená negativa - nárůst složitosti algoritmu a vyšší pravděpodobnost objevování se chyb - v živém kódování jasně žádoucí.[52]

---

[50] COLLINS, Nick; McLEAN Alex, a kol. Live coding in laptop performance. *Organized Sound. roč. 8, vyd. 3., 2003. s. 321-330.* Dostupné také z: http://users.sussex.ac.uk/~nc81/research/livecoding.pdf

[51] *Integrated Development Environment* - software usnadňující práci programátorů. pozn. aut.

[52] WARD, Adrian; ROHRHUBER Julian; a kol. Live Algorithm Programming And A Temporary Organisation For Its Promotion. *Runme.org* [online] cit. 1. 5. 2016 Dostupné z: http://art.runme.org/1107861145-2780-0/livecoding.pdf

S moderní formou live codingu nastoupivší po roce 2000 je již nerozlučně spjato promítání obrazovky účinkujících, na které je zobrazen kód tvořící to, co v danou chvíli všichni poslouchají. Tím se vlastně z pasivních posluchačů stávají aktivní posluchači i diváci zároveň, což byl také i jeden z původních záměrů live coderů - reagovat na všeobecnou diváckou pasivitu při běžných koncertech elektronické hudby. To potvrzují slova Adriana Warda z uskupení Slub:

*„Livecoding was originally a reaction to a passiveness you get with standard computer performances. The idea is to increase the improvisation and 'live-ness'. The danger of making mistakes which are audible by everyone gives a certain tension that the audience can feel."* [53]

Další současný live coder, i když přesněji live coderka, Amy Alexander k tomuto tématu ještě říká: *„Electronic music tends to remove the kinetic aspects of performance, the idea of seeing a performer perform a visible action that has an audible reaction."*[54]

Jistě, sledovat, jak hudebník desítky minut zírá do obrazovky počítače, není zrovna velká vizuální ani jiná slast, poněvadž jeho posluchači/diváci nemají nejmenší tušení, jak jeho pohyby prstů po klávesnici ovlivňují aktuálně produkovanou hudbu. Navíc, klasický software pro tvorbu hudby by nevypadal pro živé vystoupení dvakrát zajímavě.

Díky veřejnému zobrazení zdrojového kódu tak v live codingu ke zmíněnému „odstranění kinetického aspektu" nedochází, protože divák je schopen si vytvořit bezprostřední vazbu mezi změnami, jež vidí na promítané obrazovce a změnami v produkovaných zvucích. Jistěže ve značném množství případů nejsou lidé s to viděnému zcela porozumět, je pro ně však jistě zajímavé prováděné změny pozorovat a všímat si jejich návaznosti na změnu hudby.

Že s programátorskou gramotností obecenstva nepočítají ani sami live codeři, jsem již zmínil v kapitole o skupině Slub.[55] Navíc se potvrzuje, že navzdory neschopnosti detailně porozumět psanému kódu, publikum oceňuje možnost sledovat právě vznikající práci a užívají si možnost jakési spolupráce v rozpoznávání symbolických řad.

Kromě toho se tvůrci snaží přihlížejícím interpretaci kódu usnadňovat tím, že pro různé funkce používají takové názvy, jejichž poselství lidé rozeznají a pomůže jim s interpretací kódu. Jako názvy, které nikdy nezklamou, uvádí Andrew Sorensen například termíny „outrageous kick" nebo „grunge-it-up", a dodává:

*„Regardless of the audiences' level of understanding, code projection highlights to the audience that structures are coded during the performance. This*

---

[53] ARMITAGE, Tom. Slub: Making music with live computer code. *Wired.co.uk* [online] 24. 9. 2009 Dostupné z: http://www.wired.co.uk/news/archive/2009-09/25/making-music-with-live-computer-code-.aspx

[54] ANDREWS, Robert. Real DJs Code Live. *Wired.* [online] 3. 7. 2006 Dostupné z: http://www.wired.com/2006/07/real-djs-code-live/

[55] viz kapitola 4.3. Slub, s. 20

*is particularly important at this early stage in the development of live coding as a musical practice."* [56]

Dodám ještě, že kromě sugestivních názvů jsou dalšími nástroji usnadňujícím divákovi jeho orientaci v zobrazovaném kódu například tučný, jasně viditelný kurzor nebo pestrobarevné rozdělení jednotlivých částí kódu.

Při koncertech živého programování jsou nejčastěji přítomni dva účinkující; první z nich se stará o tvorbu kódu, kdežto druhý jeho výsledky pozorně sleduje, aby mohl v případě potřeby opravit nastalé chyby. Tomuto stylu se říká *párové programování.* Není však pravidlem, že má každý z umělců po celou dobu vystoupení přiřazenu pouze jednu danou roli, naopak je zvykem si je po napsání určitého bloku kódu prohodit. Dá se říci, že si tím vlastně druhý hráč vyzkouší hraní na „nástroj", jenž vytvořil jeho kolega. Pokud bychom takovýto způsob hraní srovnali s duem hudebníků hrajících na klasické hudební nástroje, zjistíme, že takovéto míry flexibilnosti přirozeně nemohou nikdy dosáhnout.

Kromě zmíněného *párového programování*, kdy si účinkující vzájemně pomáhají, označovaného jako *kolaborativní,* existuje ještě programování *soutěžní.* Jak je jistě patrné, v tomto případě spolu live codeři nespolupracují, naopak se snaží výtvor druhého umělce překonat, čímž se performance začíná velmi podobat již mnohem starším hudebním soubojům ve freestyle rapu.[57]

Při live codingové performance se mohou diváci dočkat ukázky perfektního zvládnutí programovacího editoru pomocí vynikajících psacích schopností. Středobodem celé show je tak de facto textový editor, potažmo nástroj k ní, tedy klávesnice. K té je nutno říci, že zde může být užívána dvěma způsoby; zaprvé pro zápis kódů hudebních funkcí a zadruhé jako přímý spouštěč konkrétních zvuků.

Ač se všichni ohledně live codingu zmiňují o umělcově nutné schopnosti rychlé reakce a improvizace, přece je v živém kódování prostor i pro pomalé kódování. Tato skupina live codingového společenství, o níž se zmiňuje Click Nilson, si říká *slow code movement*, a mezi její stoupence patří například známý coder Julian Rohrhuber. Ihned po prvním vyzkoušení „pomalého" kódování, jej okomentoval: *„Slow coding is my thing. I'd like to do live coding for airports."*[58] Přirovnání v druhé větě je narážkou na „otce ambientu", britského skladatele Briana Ena, jenž kdysi označil svou hudbu jako *„music for airports".*

---

[56] SORENSEN, Andrew; BROWN, Andrew R. aa-cell In Practice: An Approach To Musical Live Coding. *Proceedings of the International Computer Music Conference,* Kodaň, Dánsko, srpen 2007. Dostupné z: http://impromptu.moso.com.au/extras/aa-cell-icmc07.pdf

[57] COLLINS, Nick; McLEAN Alex, a kol. Live coding in laptop performance. *Organized Sound. roč. 8, vyd. 3., 2003. s. 321-330.* Dostupné také z: http://users.sussex.ac.uk/~nc81/research/livecoding.pdf

[58] NILSON, Click. *Live Coding Practice.* University of Sussex, Falmer, Brighton, 2007. Dostupné také z: http://users.sussex.ac.uk/~nc81/research/livecodingpractice.pdf

# 5. Textová programovací prostředí

## 5.1. SuperCollider

Jedním z nejstarších textových programovacích editorů využívaným také pro umění live codingu je jazyk SuperCollider. Byl vyvinut v roce 1996 Jamesem McCartneym jako nástroj pro algoritmickou kompozici a syntézu zvuku v reálném čase.

Velkou popularitu u hudebníků i vědců zabývajících se zpracováním počítačové hudby si získal především díky celé řadě úkolů, které s ním uživatelé mohou provádět. Z těch můžu zmínit: interaktivní spolupráci mezi hudebníky v reálném čase, zvukové instalace, generování materiálu pro komponování hudby z magnetických pásků (následně zpracovaných na digitální audio stanici), vytváření generativní hudební programy nebo na zpracovávat na míru šitou sadu nástrojů.

O oblíbenost se však zasloužil i fakt, že program může běžet na několika operačních systémech - Windows, Mac OS X i těch na platformě UNIX.

V informatice se řadí mezi *dynamické programovací jazyky,* což znamená, že patří do tzv. vysokoúrovňových jazyků. Ty jsou charakteristické schopností vykonávat několik programových reakcí zaráz, a navíc ve stejném čase zobrazovat jejich výsledky, čímž se liší od tzv. statických programovacích jazyků.[59] Pod zmíněným pojmem „reakce" si lze představit rozšiřování programu pomocí přidávání nového kódu, extenzí objektů a definic nebo modifikováním typového systému.[60]

Prostředí SuperCollideru je od roku 2012 rozděleno na dvě jednotlivé části - na server zvaný *scsynth* a klient *sclang.*[61] Jejich vzájemné komunikace je zajištěna pomocí protokolu pro spolupráci multimediálních zařízení *Open Sound Control.*

V první zmíněné části funguje aplikace *SC Server,* jenž podporuje plugin API, opět z jazyka C, díky němuž je snadné tvořit efektivní hudební algoritmy nazývané jako *generátory jednotek.* Ty jsou považovány za základní stavební jednotky programů na počítačovou hudbu již řadu let. Ve svých programech řady *MUSIC-n* je používal i Max Mathews. *SC Server* se stará o generování zvuku, jež je ve většině případů ovládáno skrz programovací jazyk ale může být použito i nezávisle na něm. Dále dokáže podporovat jakékoli množství vstupních i

---

[59] High-level programming language. *Wikipedia.org* [online] cit. 2. 5. 2016 Dostupné z: https://en.wikipedia.org/wiki/High-level_programming_language

[60] MEIJER, Erik; DRAYTON, Peter. *Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages.* Microsoft Corporation. s. 1. Dostupné také z: https://pdfs.semanticscholar.org/c803/59566e9feb4d8e2f39a3883148082c57a191.pdf

[61] WILSON, Scott a kol. *The SuperCollider Book.* MIT Press. 2011. s. 43 ISBN: 978-0262232692

výstupních kanálů, zpřístupňuje uspořádanou stromovou strukturu uzlů syntézy nebo provádí průběžné kalkulace různých kmitočtů a rychlostí.[62]

Ve druhé částí se pracuje s jazykem *SC Language.* Jde o dynamicky psaný, objektově orientovaný jazyk, kombinující strukturu reflektivního jazyku *SmallTalk* s prvky funkčních programovacích jazyků využívajících skladbu jedné z nejstarších jazykových rodin *C*. Jeho architektura pomáhá vytvářet rovnováhu mezi potřebou výpočtů v reálném čase a flexibilitou a jednoduchostí abstraktního jazyka. Stejně jako spousta dalších funkčních jazyků implementuje funkce jako objekty první třídy[63], které mohou být následně skládány dohromady. Funkce a metody mohou mít hodnoty standardních argumentů a dobu trvání proměnných argumentů, a mohou být vyvolány jakýmkoliv příkazem z argumentů klíčových slov.[64] *Uzavření* jsou zde lexikální a *rozsah* je jak lexikální, tak dynamický. Mezi další vlastnosti, typické právě pro funkční jazyky

Díky podpoře metod reflektivního, interaktivního a literárního programování je SuperCollider schopen relativně snadno nacházet algoritmy zcela nových zvuků a dle potřeb uživatele vyvíjet software a aplikační rámce.
Jazyk Super Collideru také uživateli dovoluje tvořit multiplatformní grafické uživatelské prostředí pro aplikace. Standardní knihovna tříd se složkami uživatelského prostředí tak může být rozšířena o počet dostupných aplikačních rámců (frameworků). V interaktivním programování navíc systém podporuje programový přístup ke kódovým souborům *Rich Text Formátu,* což může být použito k algoritmickému generování vektorové grafiky.

SuperCollider si jako jazyk pro programování hudby vytvořil oddanou uživatelskou základnu a je příkladem skvělého zařízení pro tvorbu generativní a interaktivní hudby v reálném čase. Mezi jeho věrné uživatele patří např. známý německý live coder Julian Rohrhuber, který tento program užívá již od jeho vzniku a spravuje také stránky poskytující podporu komunitě okolo SuperCollideru.

## 5.2. Impromptu

Na rozdíl od SuperCollideru je Impromptu programovacím prostředím určeným především pro umění live codingu. Vzhledem k tomu, že jeho otcem je právě jedna z nejdůležitějších postav live codingové scény - Andrew Sorensen, to však není nic překvapujícího. Rozhodně však není prostředím pouze pro live coding, jeho funkcí mohou využívat i další umělci pracující se zvukem, VJové nebo umělci pracující s grafikou a mající zájem o živé či interaktivní programování.

---

[62] WILSON, 2011. s. 48.

[63] Objekty první třídy jsou ty, jejichž identita je zcela nezávislá na jiných. pozn. aut.

[64] Jako argument se označují zadávané vstupní hodnoty podprogramů.

Termín *impromptu* původně označoval skladbu volné délky i formy, kombinovanou bez přípravy. Zajímavostí je, že poprvé tohoto termínu použil roku 1817 hudební časopis *Allgemeine musikalische Zeitung* pro označení díla českého skladatele Jana Václava Voříška.[65]

Za další podstatný rozdíl může být považován fakt, že je toto prostředí určeno pouze pro operační systém Mac OS X. Jeho základ tvoří multiparadigmatický funkční programovací jazyk Scheme, jenž patří do dialektu velmi starého jazyka Lisp.[66]

Prostředí programu přirozeně nabízí uživateli provádět změny v programu za jeho chodu, u proměnných a funkcí tak mohou být kdykoliv prováděny změny, které se ihned projeví. Pomyslným středobodem celého programu je čas. Impromptu totiž uživateli dovoluje zcela přesné časové rozvrhnutí všech událostí a celého kódu. Stává se tak velmi dynamickým a mocným prostředím pro tvorbu a manipulování s programy běžícími v živě probíhajícím představení.

Impromptu se může také stát hostitelem AudioUnits. Audio Units jsou zásuvné modely používané ke zpracování zvuku vytvořeného v hudebních aplikacích systému Mac OS X, jako jsou např. GarageBand, Logic Express nebo Soundtrack Pro. Jeho prostředí umožňuje tvorbu grafů zpracování signálu s libovolnou komplexností a poskytuje programátorovi zcela přesnou kontrolu nad jednotlivými uzly zásuvných modelů.
Navíc, pomocí těchto nástrojových uzlů může být do precizního časového rozvrhu naplánován jakýkoliv hudební materiál, a za chodu mohou být programově přetvořeny libovolné změny a přednastavení programu.

Sorensenovo prostředí poskytuje tvůrcům možnost psát, upravovat, sestavovat a za chodu měnit personalizovaný kód zpracování signálu. Posledně jmenovaná schopnost umožňuje programátorovi psát kód zvukové syntézy nízké úrovně během živého vystoupení. Tato funkce je stvořena ke spolupráci s grafem signálu AudioUnit modulů, dovolujíce umělci libovolně mixovat jeho oblíbené moduly s jeho vlastním kódem syntézy.

Se stejnou časovou přesností, s jakou je používán zvukový materiál, mohou být používány i grafické podprogramy. Díky tomu je tvůrce schopen audio a vizuální stránku svého díla dokonale sjednotit. [67] [68]

---

[65] Jan Václav Hugo Voříšek. *Skladatelé - Český rozhlas.* [online] cit. 11. 5. 2016 Dostupné z: http://www.rozhlas.cz/klasika/skladatele/_zprava/jan-vaclav-hugo-vorisek--953373

[66] DYBVIG, R. Kent. Chapter 1: Introduction. *The Scheme Programming Language.* MIT Press, 4. edice. [online] Dostupné z: http://www.scheme.com/tspl4/

[67] SORENSEN, Andrew. Impromptu: An Interactive programming environment for composition and performance. *impromptu.* [online] cit. 3. 5. 2015 Dostupné z: http://impromptu.moso.com.au/extras/sorensen_acmc_05.pdf

[68] SORENSEN, Andrew; BROWN, Andrew R. aa-cell In Practice: An Approach To Musical Live Coding. *Proceedings of the International Computer Music Conference,* Kodaň, Dánsko, srpen 2007. Dostupné z: http://impromptu.moso.com.au/extras/aa-cell-icmc07.pdf

Jednou z nejslavnějších skladeb vytvořených pomocí Impromptu je ve světě live codingu bezesporu *A Study in Keith,* již má na svědomí právě otec programu Andrew Sorensen. Vytvořil ji v roce 2007 a její název odkazuje na legendu americké jazzové scény Ketiha Jarretta. Sorensen totiž ve skladbě napodobuje Jarrettův styl; pokládá vedle sebe dvě propojené vrstvy představení, v tomto případě reprezentované prací s počítačovou querty klávesnící a klasickou klaviaturou.

# 6. Ostatní programovací prostředí

## 6.1. Grafické prostředí Pure Data

Hudebníci, performeři, vizuální umělci, hudební i grafičtí výzkumníci i vývojáři. Ti všichni mohou ve vizuálním programovacím prostředí Pure Data tvořit programy bez jakéhokoliv psaní dlouhých sekvencí kódu. Program vytvořil společně se svým týmem profesor hudby na University of California v San Diegu Miller Puckette. Ten je také autorem dřívějšího programu Max, s nímž si je Pure Data v jistých ohledech - rozsahu nebo vzhledu - velmi podobný. Sám autor však i vysvětlil, v čem se tyto programy naopak liší:

*„Systém Pure Data od počátku vznikal s představou grafického editování datových struktur, zatímco základním stavebním prvkem Maxu byl „objekt", který se propojoval s dalšími objekty. V Pure Data lze takové objekty také vytvářet, máte ovšem přístup i „do větší hloubky" datových struktur."*[69]

Program může být využit na zpracování a vytváření zvuku, videa nebo grafik, a jako rozhraní pro připojení nejrůznějších senzorů a vstupních zařízení (např. MIDI). Dále je schopný snadno pracovat přes místní i vzdálené sítě a je vhodný na učení se základního zpracování multimédií metod vizuálního programování, stejně jako pro vytváření složitých systémů pro projekty velkého rozsahu.

Tento programovací jazyk si může každý stáhnout zdarma, poněvadž Puckette se rozhodl jej zpřístupnit jako *open-source.* K tomu se uchýlil proto, že pro něj prý volně přístupný software znamená jistotu, že bude moci svou práci šířit navždy.[70] Navíc tak k jeho práci může přispívat spousta dalších programátorů, což se zrovna u tohoto programu děje ve velkém.

S tím souvisí dvě verze programu, jež je možné si volně pořídit. První je označována jako *Pd vanilla* nebo jen jednoduše *Pd,* a obsahuje jádro napsané převážně Millerem Puckettem, zaměřující se na audio signál a zpracování MIDI vstupu.

Druhá má název *Pd extended,* což je vlastně *Pd vanilla* rozšířená o spoustu knihoven napsaných členy komunity. Tato verze může být použita např. pro grafické renderování, zpracování binárních souborů, streamování videa i zvuku, tvorbu fyzikálních modelů nebo dokonce představení na bázi senzorů.[71]

Obé verze jsou k dispozici pro Windows, Mac OS X, Linux i další nezávislé operační systémy. PureData je najvíc multiplatformní; může běžet takřka na čemkoliv, ať už jde o starý notebook, moderní tablet i chytrý telefon.

---

[69] KRATOCHVÍL, Matěj. Budoucnost patří algoritmům. *His Voice.* [online] 2. 1. 2006 Dostupné z: http://www.hisvoice.cz/cz/articles/detail/196

[70] Ibid.

[71] Pure Data. *Pure Data - Pd Community Site.* [online] cit. 2. 4. 2016 Dostupné z: https://puredata.info

PureData je příkladem programovacích jazyků datových toků (dataflow). Dataflow programování je programovacím paradigmatem, jež vytváří program jako orientovaný graf
toku dat mezi operacemi, a tak implementuje principy a architekturu datového toku.[72]

Základem tvorby algoritmů v tomto prostředí je tvorba víceúrovňových vazeb mezi grafickými objekty, v tomto případě malými obdélníčky. Do těch uživatel napíše dané instrukce, na jejichž základě pak změní své chovaní, popřípadě i vzhled. Díky takovému postupu je programování relativně uspořádané a dobře orientační a dá se ho tedy v případě přítomnosti alespoň základní snahy snadno naučit. Výsledným programům vzniknuvším z těchto chytrých obdélníků se říká *patche*.

Ač se takový přístup může zdát novátorský, je vlastně značně podobný práci se starými analogovými syntezátory, u nichž se signál také spojoval dohromady, samozřejmě však pomocí drátů. Avšak na rozdíl od těchto syntezátorů, jež se během svého života postupně proměnily v ne tak složité aparáty *„využívající především předdefinovaný počet harwarových oscilátorů, filtrů a efektů* [...]*, u Pd je stále zachována komplexita a modularita. Hlavní charakteristikou Pd je* [díky tomu] *jeho všestrannost s téměř neomezenými možnostmi použití."* [73]

S všestranností samozřejmě narůstá i složitost pracovního prostředí, které tak může některé uživatele zprvu vyděsit. Při detailnějším pohledu se však ukáže, že jde o šikovně propracovanou strukturu, jejíž domnělá zmatečnost a složitost se dá navíc zredukovat pomocí nabízeného vytvoření osobního grafického rozhraní.

Takováto intuitivní datová struktura dovoluje uživatelům modelovat hudební data do statických i dynamických nebo animovaných grafických podob libovolné komplexnosti.

Jak říká Miller Puckette, Pure Data navrhl tak, aby nabízelo extrémně nestrukturované prostředí pro vytváření datových struktur a jejich grafického vzhledu.[74]

Hlavní myšlenkou Pure Data ostatně je umožnit tvůrcům zobrazit jakákoliv data, která si přejí. K dosažení toho Pd nabízí grafickou datovou strukturu, v jistých ohledech podobnou struktuře jazyka C, obohacenou však o možnost připojovat k datům rozličné tvary a barvy, díky nimž je může programátor snadno vizualizovat a upravovat.

---

[72] Dataflow programming. *Wikipedia.org* [online] cit. 3. 5. 2016 Dostupné z: https://en.wikipedia.org/wiki/Dataflow_programming#Languages

[73] FLOREŠ, Ivan. *Pure Data: Nástroj pro algoritmické umění.* Masarykova Univerzita, Brno, 2011. Bakalářská diplomová práce. s. 20-21.

[74] PUCKETTE, Miller. Pd Documentation. *Miller Puckette University of California San Diego.* [online] cit. 4. 5. 2016 Dostupné z: http://msp.ucsd.edu/Pd_documentation/x2.htm#s9

## 6.2. Personalizovaná prostředí vytvořená v hnutí TOPLAP

### Fluxus

Prostředí *fluxus* si vytvořil čistě pro své potřeby britský live coder Dave Griffiths na základě standardu multiplatformního rozhraní Open Graphics Library (OpenGL). Jde o renderovací nástroj, který dokáže zpracovávat audio vstup a vytvářet z něj v reálném čase animace. Ve druhém okně se ve stejném čase se také vytváří kód detailně popisující promítané výjevy. K tomu sám autor říká: *„It resembles the Maya* [skriptovací jazyk], *in that you write code and highlight it to execute it in fragments. That way you can build up a palette of function calls and bits of code and execute them with the mouse and a keypress."*[75]
K tomu ještě dodává, že je fluxus vytvořen primárně pro živé kódování a měl by k němu tak být používán.

### The Thingee

Program vytvořený novomediální a audiovizuální umělkyní Amy Alexander, která o něm říká, že je dvěma nástroji zároveň - live codovým editorem a nástrojem pro VJing. Kód, který do něj uživatel napíše, je okamžitě zpracováván a zároveň z něj vytváří vizuály. Jde tedy o systém principem velmi podobný předchozímu *fluxusu.*
Na obrazovce programu je během psaní zobrazeno šest různých editorů, a dvě 'pojistky', jež v případě potřeby dokáží uložit obraz a údaje aktuální situace na obrazovce (aplikovatelné pro zpětnou vazbu).[76]

---

[75] WARD, Adrian; ROHRHUBER Julian; a kol. Live Algorithm Programming And A Temporary Organisation For Its Promotion. *Runme.org* [online] cit. 1. 5. 2016 s. 254-256 Dostupné z: http://art.runme.org/1107861145-2780-0/livecoding.pdf
[76] Ibid. s. 252-254

# Závěr

Mezi Mozartovou *Musikalisches Würfelspiel* a kupříkladu Sorensenovou *Study in Keith* zeje vskutku propastný rozdíl. Jednu věc, nepočítaje přitom algoritmický princip, však mají společnou. Stvořili je hudebníci - experimentátoři, kteří chtěli vytvořit něco zcela nového, co nikdo před nimi neudělal, a posunout tak hudební teorii i praxi zase o kousek dál. A kdo ví, kdyby byl salzburský génius žil v naší době, možná by dnes, krom jiného, byl jedním z aktivních live coderů na univerzitě v Karlsruhe.

Ať už by tato fantazie došla naplnění či ne, jedno je jisté; byla to lidská touha něco dokázat, šikovnost a kreativita, která posunula dál hranice počítačové hudby a audio-vizuálního umění i umění jako celku. Nikdo z těch umělců, které ve své práci zmiňuji, se touto tvorbou nezabývají ze zištných důvodů, ba naopak. Velké množství open-source softwarů je toho důkazem.

Live coding je krásná umělecká forma, jež od svých tvůrců vyžaduje skutečný zápal pro věc, snahu a spoustu šikovnosti i osobního času. Dokonale naplňuje definici multimediálního umění a svou náročností přesahuje spoustu jiných forem počítačového umění a umění vůbec. Divákům a posluchačům zase nabízí tak detailní nahlédnutí do samého nitra umělcovy práce, jakého se jim u jiných forem umění dostane jen stěží.

Díky umění živého kódování vzniklo několik zajímavých programovacích prostředí a nových jazyků. Programátoři se nabídla další a velmi významná možnost, jak ukázat své nejen programátorské ale i umělecké vlohy, praktický i teoretický svět hudby byl obohacen o celé nové odvětví, a klubová scéna se rozšířila o *algorave.*

Je velká škoda, že se live codingu nedostává více prostoru a pozornosti, které si dle mého zaslouží. Vzhledem k tomu, že velká část populace ani neví, že takové umění existuje, možná by jen stačila lepší propagace a informovanost. Jsem si totiž jist, že by umění živého kódování mohlo zaujmout více lidí, než v současné chvíli. A byl bych tak velmi rád, kdyby má práce, byť i jen malou měrou, k této propagaci a osvětě přispěla.

# Resumé

Práce se zabývá uměním živého kódování, jinak taky zvaného jako live coding. V úvodu je zmíněn kulturní kontext a přiblížen vliv softwaru na dnešní společnost. Část práce je věnována historii a vývoji počítačem generované hudby a zvukové syntézy, zmíněna jsou velká jména počítačové hudby jako Lejaren Hiller, Iannis Xenakis nebo John Cage. Velký prostor dostane historie a popis procesu algoritmické hudební kompozice, tvořené jak bez počítače, tak i s jeho pomocí. Detailně jsou rozebrány dějiny a původ live codingu, jeho význačné osobnosti jako Andrew Sorensen, Nick Collins nebo Julian Rohrhuber, uskupení a samotný proces live codingového představení.
Neopomenuta je technická stránka věci, tedy různá programovací prostředí a jejich jazyky používané k tvorbě živého kódování, a role live codingu ve světě umění.

# Summary

This work aims to present the art of live coding, sometimes reffered to as 'on-the-fly' programming. In the beggining its cultural context is mentioned, as well as influence of software on current society. One part of the work is dedicated to history and evolution of computer generated music and digital sound synthesis. Important names of computer music like Lejaren Hiller, Iannis Xenakis or John Cage are not omitted. A sufficient space is also given to algorithmic composition, its history and process of making it itself.
The history, the origin and the process of live coding is described in detail, great names of live coding like Andrew Sorensen, Nick Collins or Julian Rohrhuber are mentioned as well.
In the end are analysed various programming enviroment, programming languages and the role of live coding in the world of art.

# Seznam zdrojů a použité literatury

## Tištěné zdroje:

1. HILLER, Lejaren; ISAACSON Leonard M. *Experimental Music: Composition With an Electronic Computer.* McGraw-Hill, New York, 1959, 2. vyd. s. 5-7. ISBN: 978-0-313-22158-3.

2. FERNÁNDEZ, José David; VICO Francisco. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research.* č. 48, 2013. s. 528-532.

3. ROADS, Curtis. *The Computer Music Tutorial.* MIT Press, 1996. s. 89. ISBN: 978-0262680820

4. XENAKIS, Iannis. *Formalized Music: Thought and Mathematics in Composition.* Pendragon Press, 1994. s. 144. ISBN: 2234015103

5. WEISS, Piero; TARUSKIN RIchard. *Music in the Western World: a history in documents.* Cengage Learning, 2008. op. cit. s. 3. ISBN: 978-0-534-58599-0.

6. WILSON, Scott a kol. *The SuperCollider Book.* MIT Press. 2011. s. 43 ISBN: 978-0262232692

### Elektronické zdroje:

1. ALPERN, Adam. *Techniques for algoritmic composition of music.* Hampshire College, 1995. s. 1 cit. 12. 5. 2015 Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9364&rep=rep1&type=pdf

2. ANDREWS, Robert. Real DJs Code Live. *Wired.* [online] 3. 7. 2006 Dostupné z: http://www.wired.com/2006/07/real-djs-code-live/

3. Andrew Sorensen. *Pioneers.* [online] cit. 1. 5. 2016 Dostupné z: https://pioneers.io/blog/people/andrew-sorensen/

4. ARMITAGE, Tom. Slub: Making music with live computer code. *Wired.co.uk* [online] cit. 24. 9. 2009 Dostupné z: http://www.wired.co.uk/news/archive/2009-09/25/making-music-with-live-computer-code-.aspx

5. Batch processing. *PC Mag.* [online] cit. 30. 4. 2016 Dostupné z: http://www.pcmag.com/encyclopedia/term/38460/batch-processing

6. BROWN, Chris. The first Hub. *Crossfade - Walker Art Center.* [online] cit. 1. 5. 2016 Dostupné z: http://crossfade.walkerart.org/brownbischoff/

7. COLLINS, Nick; McLEAN Alex, a kol. Live coding in laptop performance. *Organized Sound. roč. 8, vyd. 3., 2003. s. 321-330.* Dostupné také z: http://users.sussex.ac.uk/~nc81/research/livecoding.pdf

8. Dataflow programming. *Wikipedia.org* [online] cit. 3. 5. 2016 Dostupné z: https://en.wikipedia.org/wiki/Dataflow_programming#Languages

9. DI NUNZIO, Alex. Illiac Suite. *Musica informatica.* [online] cit. 29. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/illiac-suite.php

10. DI NUNZIO, Alex. Groove. *Musica informatica.* [online] cit. 29. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/groove.php

11. DI NUNZIO, Alex. Musicomp. *Musica informatica.* [online] cit. 30. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/musicomp.php

12. DI NUNZIO, Alex. HPSCHD. *Musica informatica.* [online] cit. 30. 4. 2016 Dostupné z: http://www.musicainformatica.org/topics/hpschd.php

13. DYBVIG, R. Kent. Chapter 1: Introduction. *The Scheme Programming Language.* MIT Press, 4. edice. [online] Dostupné z: http://www.scheme.com/tspl4/

14. FILDES, Jonathan. 'Oldest' computer music unveiled. *BBC News.* [online] cit. 29. 4. 2016 Dostupné z: http://news.bbc.co.uk/2/hi/technology/7458479.stm

15. First Recorded Computer Music. *YouTube.* video [online] Dostupné z: https://www.youtube.com/watch?v=xLwjz0UR5_A

16. FLAŠAR, Martin. *Elektroakustická hudba.* Masarykova Univerza, Brno, 2015. op. cit. s. 81. Dostupné z: http://is.muni.cz/do/rect/el/estud/ff/ps15/eah/web/media/Elektroakusticka_hudba.pdf

17. FLOREŠ, Ivan. *Pure Data: Nástroj pro algoritmické umění.* Bak. dip. práce, Masarykova Univerzita, Brno, 2011. s. 17. Dostupné také z: http://is.muni.cz/th/256351/ff_b/pure_data_ivan_flores_2011.pdf

18. High-level programming language. *Wikipedia.org* [online] cit. 2. 5. 2016 Dostupné z: https://en.wikipedia.org/wiki/High-level_programming_language

19. HistoricalPerformances. *TOPLAP.* [online] cit. 1. 5. 2016 Dostupné z: http://toplap.org/wiki/HistoricalPerformances

20. HMSL - Heirarchical Music Specification Language. *SoftSynth.* [online] cit. 1. 5. 2016 Dostupné z: http://www.softsynth.com/hmsl/

21. CHESHIRE, Tom. Hacking meets clubbing with the 'algorave'. *Wired.co.uk* [online] 23. 8. 2013 Dostupné z: http://www.wired.co.uk/magazine/archive/2013/09/play/algorave

22. CHUANG, John. Mozart's Musikalisches Würfelspiel. *SunSITE Austria at the University of Vienna.* [online] cit. 29. 4. 2016 Dostupné z: http://sunsite.univie.ac.at/Mozart/dice

23. Jan Václav Hugo Voříšek. *Skladatelé - Český rozhlas.* [online] cit. 11. 5. 2016 Dostupné z: http://www.rozhlas.cz/klasika/skladatele/_zprava/jan-vaclav-hugo-vorisek--953373

24. KRATOCHVÍL, Matěj. Budoucnost patří algoritmům. *His Voice.* [online] 2. 1. 2006 Dostupné z: http://www.hisvoice.cz/cz/articles/detail/196

25. ManifestoDraft. *TOPLAP.* [online] cit. 30. 4. 2016 Dostupné z: http://toplap.org/wiki/ManifestoDraft

26.MANOVICH, Lev. *Cultural Software.* [online] 2011. Dostupné z: http://manovich.net/content/04-projects/070-cultural-software/67-article-2011.pdf

27. McLEAN, Alex. Algoravers. *TOPLAP.* [online] 2. 11. 2012 Dostupné z: http://toplap.org/algoravers/

28. MEIJER, Erik; DRAYTON, Peter. *Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages.* Microsoft Corporation. s. 1. Dostupné také z: https://pdfs.semanticscholar.org/c803/59566e9feb4d8e2f39a3883148082c57a191.pdf

29. NIERHAUS, Gerhard. *Algorithmic Composition: Paradigms of Automated Music Generation.* SpringerWienNewYork, 2009. [verze Kindle] s.4. Získáno z Amazon.com

30. NILSON, Click. *Live Coding Practice.* University of Sussex, Falmer, Brighton, 2007. Dostupné také z: http://users.sussex.ac.uk/~nc81/research/livecodingpractice.pdf

31. Odst. 1b. § 63. Powers to remove persons attending or preparing for a rave. Část V. Powers in relations to raves. Criminal Justice and Public Order Act 1994. *Legislation.gov.uk.* Dostupné také z: http://www.legislation.gov.uk/ukpga/1994/33/part/V/crossheading/powers-in-relation-to-raves#term-music

32. OWEN, James. Bone Flute Is Oldest Instrument, Study Says. *National Geographic News.* [online] cit. 29. 4.. 2016 Dostupné z: http://news.nationalgeographic.com/news/2009/06/090624-bone-flute-oldest-instrument.html

33. Prof. Julian Rohrhuber. *Institute For Music And Media.* [online] cit. 3. 5. 2016 http://musikundmedien.net/en/studies/faculty/julian-rohrhuber/

34. Powebooks Unplugged. ZXZW independent culture. [online] cit. 3. 5. 2016 Dostupné z: http://past.incubate.org/2007/acts.php?id=119

35. PUCKETTE, Miller. Pd Documentation. *Miller Puckette University of California San Diego.* [online] cit. 4. 5. 2016 Dostupné z: http://msp.ucsd.edu/Pd_documentation/x2.htm#s9

36. Pure Data. *Pure Data - Pd Community Site.* [online] cit. 2. 4. 2016 Dostupné z: https://puredata.info

37. Remembering Max - A tribute. *cSound.* [online] cit. 29. 4. 2016 Dostupné z: http://www.csounds.com/mathews/max_ideas.html

38. SHANNON, Claude. A Mathematical Theory of Communication. *The Bell System Technical Journal,* 1948.s. 379–423. Dostupné z: http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf

39. SCHUSTER, Werner. Andrew Sorensen on Real Time Programming, Live Coding Music, Memory Management. *InfoQ.* [online] 13. 12. 2014 Dostupné z: http://www.infoq.com/interviews/sorensen-extempore-realtime

40. SORENSEN, Andrew; BROWN, Andrew R. aa-cell In Practice: An Approach To Musical Live Coding. *Proceedings of the International Computer Music Conference,* Kodaň, Dánsko, srpen 2007. Dostupné z: http://impromptu.moso.com.au/extras/aa-cell-icmc07.pdf

41. SORENSEN, Andrew. Impromptu: An Interactive programming environment for composition and performance. *impromptu.* [online] cit. 3. 5. 2015 Dostupné z: http://impromptu.moso.com.au/extras/sorensen_acmc_05.pdf

42. Stochastic. *Your Dictionary.* [online] cit. 30. 4. 2016 Dostupné z: *http://www.yourdictionary.com/stochastic*

43. WARD, Adrian; ROHRHUBER Julian; a kol. A Historical Perspective. Live Algorithm Programming And A Temporary Organisation For Its Promotion. *Runme.org* [online] cit. 1. 5. 2016 Dostupné z: http://art.runme.org/1107861145-2780-0/livecoding.pdf