

Guilherme Martins Lunhane

***Study in Keith: um framework preliminar para  
um live coding de Andrew Sorensen.***

4 de novembro de 2015



Guilherme Martins Lunhane

***Study in Keith: um framework preliminar para um live coding de Andrew Sorensen.***

Prévia da dissertação para banca de qualificação no Programa Mestrado em Artes, Cultura e Linguagens do IAD-UFJF, frente em Artes Visuais, Música e Tecnologia.

Universidade Federal de Juiz De Fora – UFJF

Instituto de Artes e Design – IAD

Programa de Pós-Graduação em Artes Visuais, Música e Tecnologia

Orientador: Prof. Dr. Luiz Eduardo Castelões

4 de novembro de 2015

*À Via que gerou o um. Ao um que gerou o dois. Ao dois que gerou o três. E ao três que gerou as dez mil coisas.*

# Agradecimentos

Ao inominável.

À minha família, Jair e Júlia, por todo tipo de apoio, amo vocês!

Aos amigxs que estão (ou moraram em Juiz de Fora) e que foram fundamentais (de alguma forma) no caminho: Glerm, Anna Flávia, Tiago Rubini, Aline. Aos amigxs de Campinas e São Paulo que de alguma forma me ajudaram ultrapassar as distâncias, nostalgias e amizades: Doshi, Larissa, Dani, Tati, Fábio, Rebecchi, Felício).

Aos vizinhos que riem apenas de ver a lua, Dhiego e Luisa. Aos colegas de mestrado que também batalharam durante o período vigente, Diego, Nayse, Analu (e Paula). Aos velhos amigos que não vejo a muito

Aos Professores Dr. Luiz Eduardo Castelões, Dr. Alexandre Fenerich e Dr. Flávio Luiz Schiavonni por serem fundamentais, no apoio institucional; na sugestão de leituras; na cobrança de prazos; nas críticas; nas conversas sobre Música, Universidade e Tecnologia; ou até mesmo pelo sanduíche de queijo quando a bolsa não caiu. À FAPEMIG por suprir esta lacuna, em um momento delicado nas finanças da Universidade Brasileira.

À Professora Rosane Preciosa. Seus brilhos deram novos significados ao *Momentum* de Stockhausen. Ao Professor Dr. Elpp Aravena e Tonil Braz, Engenheiros da Cumbia.

Ao Professor Hans Joachim Koellreutter, para você, redes de silêncios e gestos sonoros. Tocar sua peça foi algo revelador.

Àqueles que passaram pelo caminho, que ajudaram ou atrapalharam. Suas potências permitiram a concepção de um novo universo de possibilidades.

Alguém poderia imaginar uma interface musical na qual um músico especifica o som resultante desejado em uma linguagem descritiva, na qual poderia ser então traduzido em parâmetros de partículas e renderizados em som. Uma alternativa poderia especificar um exemplo: "Faça um som assim (arquivo de som), mas com pouco vibrato"<sup>a</sup>

---

<sup>a</sup> Tradução de: *One can imagine a musical interface in which a musician specifies the desired sonic result in a musically descriptive language which would then be translated into particle parameters and rendered into sound. An alternative would be to specify an example: "Make me a sound like this (soundfile), but with less vibrato"*

Curtis [Roads](#) (2001)

# Resumo

Esta pesquisa foi construída a partir da seguinte pergunta: ao discutir uma prática conhecida como *live coding*, quais categorizações sonoras (ou gêneros musicais) são contextualizadas? No decorrer da pesquisa encontramos os mais variados exemplos, desde o *noise*, música eletrônica de pista e *jazz*.

A variedade dos diferentes exemplos da pergunta forçou a redução de, nas palavras de Wiggins (2006) e Alex McLean (2006), nosso *espaço conceitual*, inerente às categorizações sonoras pesquisadas (Capítulo 1).

Um método assimétrico, inspirado no sociólogo Boaventura de Souza Santos (2008), Santos (2007) (e nos espaços conceituais do parágrafo anterior), propõe um tratamento histórico de conceitos do *livecoding*, para poder lidar com os conceitos específicos de um exemplo (Capítulo 2)

O tratamento histórico será apresentado no Capítulo 3.

O Estudo de caso específico, a sessão de *livecoding Study in Keith*, do compositor e pesquisador Australiano Andrew Sorensen (2012) será apresentado no Capítulo 4. Seguiremos com ferramentas analíticas da peça baseada nos espaços conceituais apresentados no Capítulo 1. Este caso foi escolhido por caracterizar uma *replicação do estilo de improvisação* do compositor e pianista Keith Jarret, durante os concertos *Sun Bear*, gravados em 1976 e lançados em 1979 pela ECM.

**Palavras-chaves:** *Live coding. Study in Keith.* Sistemas criativos. Jazz.





# Sumário

	<b>Introdução</b>	<b>9</b>
<b>1.1</b>	<b>A questão da tradução</b>	<b>12</b>
<b>1.2</b>	<b>Sistemas Criativos, e o Universo de Conceitos</b>	<b>15</b>
1.2.1	Sistemas Criativos	16
1.2.2	Ferramenta de Estruturação de Sistemas Criativos	17
1.2.3	O modelo de improvisação	17
<b>1.3</b>	<b>Exemplo concreto da multiplicidade em um universo de possibilidades</b>	<b>19</b>
<b>1</b>	<b>PERGUNTA PARA O MÉTODO: <i>LIVE CODING</i> É MÚSICA?</b>	<b>11</b>
<b>2</b>	<b>METODOLOGIA</b>	<b>25</b>
<b>2.1</b>	<b>Método de pesquisa</b>	<b>27</b>
<b>3</b>	<b>TRABALHOS RELACIONADOS</b>	<b>29</b>
<b>3.1</b>	<b>Predecessores</b>	<b>30</b>
3.1.1	GROOVE	30
3.1.2	Pietro Grossi	32
3.1.3	Baía de São Francisco	33
3.1.4	Just In Time (JIT)	35
3.1.5	Live Algorithm Programming and Temporary Organization for its Promotion	36
3.1.6	<i>Show us your screens</i>	37
<b>3.2</b>	<b><i>Live coding</i> e Música de algoritmos</b>	<b>40</b>
3.2.1	Música de algoritmos simples	40
3.2.2	Música de algoritmos complexos	43
<b>3.3</b>	<b>Prática DJ</b>	<b>44</b>
<b>4</b>	<b>ESTUDOS DE CASOS</b>	<b>47</b>
<b>4.1</b>	<b>Study in Keith</b>	<b>47</b>
4.1.1	Concertos <i>Sun Bear</i>	48
4.1.2	Algorithms are Thoughts, Chainsaws are Tools	48
<b>5</b>	<b>CONCLUSÃO</b>	<b>51</b>
	<b>Referências</b>	<b>53</b>



# Introdução

A adoção do computador como instrumento musical remontou, neste trabalho, às investigações de Max Mathews (1963), Mathews et al. (1969), Mathews e Moore (1970). Enquanto os dois primeiros apresentam fundamentos do Processamento de Sinais Digitais (DSP) para o fazer musical em *tempo diferido*, o último apresenta a possibilidade de uma performance musical humana mediada pela máquina digital. Através de um sistema de retroalimentação, um ser humano era capaz de sintetizar sons, e manipulá-los em tempo de execução.

De certa forma, tais publicações permitiram a emergência de um *Programa de Investigação Científica* (Lakatos, 1970; Neto, 2008) envolvendo Música e Ciências da Computação. Resultados de pesquisas posteriores, foram materializadas na elaboração de diversos ambientes de programação musical, entre eles, o CSound<sup>1</sup>, Max/MSP<sup>2</sup>, PureData<sup>3</sup> e SuperCollider<sup>4</sup>.

Deste último *software*, foi possível extrair um *macro-objeto* de pesquisa. Este macro-objeto é uma atividade social, uma prática, e um paradigma científico, ou nas palavras de Wiggins (2006) ou Alex McLean (2006), um *Universo de possibilidades* para improvisações musicais. Esta atividade peculiar ficou conhecida como *livecoding*, ou:

*Programação imediata* (ou: programação de conversa, programação no fluxo, programação interativa) é um paradigma que inclui a atividade de programação ela mesma como uma operação do programa. Isto significa um programa que não é tomado como ferramenta que cria primeiro, e depois é produtivo, mas um processo de construção dinâmica de descrição e conversação - escrever o código e então se tornar parte da prática musical ou experimental. (SuperCollider.ORG, 2014, Verbete JITLib)<sup>5</sup>

A definição acima é apropriada para um verbete de um documento técnico. Porém reduz o *livecoding* ao modo de operação com o computador, e não descreve possíveis resultados musicais. Teoricamente, qualquer estética musical pode ser reproduzida durante uma sessão de improvisação (*livecoding session*). Dependendo de como este universo é configurado, diferentes estéticas musicais podem emergir. Este tema será discutido no [Capítulo 1](#).

<sup>1</sup> Disponível em <<https://csound.github.io/>>.

<sup>2</sup> Disponível em <<http://cycling74.com/products/max>>.

<sup>3</sup> Disponível em <<http://puredata.info/>>.

<sup>4</sup> Disponível em <<https://supercollider.github.io/>>.

<sup>5</sup> Tradução de *Just in time programming (or: conversational programming, live coding, on-the-fly programming, interactive programming)* is a paradigm that includes the programming activity itself in the program's operation. This means a program is not taken as a tool that is made first, then to be productive, but a dynamic construction process of description and conversation - writing code thus becoming a closer part of musical or experimental practice.

No [Capítulo 2](#) delimito os conceitos deste universo, e derivo o *espaço conceitual multidimensional* desta pesquisa, que envolve basicamente a correlação entre os *universos de possibilidades* próprios do *livecoding* e da sessão *Study in Keith*.

No [Capítulo 3](#), descrevo espaços conceituais históricos do *universo de conceitos do livecoding*. Mais especificamente, os predecessores da prática, e a emancipação de um Programa de Investigação Científica específico deste universo. Esta investigação permitiu observar sua ideologia e sua heurística.

No [Capítulo 4](#) investigo um caso específico do compositor e programador australiano Andrew Sorensen. Dada a multiplicidade do conceito de *livecoding*, foi necessário reduzir o escopo da pesquisa. Por outro lado, o exemplo de Sorensen é interessante por propor uma *replicação* não só de um gênero musical, mas de um estilo idiossincrático de improvisação *jazz*.

# 1 Pergunta para o método: *live coding* é música?

Giovanni Mori (2015a) perspetivou a importância da Música para os *live coders* de um ponto-de-vista etnográfico, ao apontar locais e costumes onde o *live coding* é praticado:

*Live coding* é uma técnica artística de improvisação. Pode ser empregada em muitos contextos performativos diferentes: dança, música, imagens em movimento e mesmo tecelagem. Concentrei minha atenção no lado musical, que parece ser o mais proeminente. (MORI, 2015a, p. 117)<sup>1</sup>

O fragmento acima sugere que a Música, no *live coding*, é usada de maneira permissiva. Não apenas tangente às linguagens audiovisual e do corpo, ou à mistura de categorizações musicais, mas também na capacidade de agregar pessoas.

Através de Victor Turner e Harris (1969), Carolina Prospero (2015) apresenta a comunidade de intérpretes e o público do *live coding* como semelhante às “comunidades abertas, que diferem das estruturas de comunidades fechadas”. Este tipo de comunidade é denominada *communita*. (TURNER; HARRIS, 1969 apud PROSPERO, 2015, p. 71).

Ao tratar desta configuração social, é aberta a discussão de uma *liminaridade* própria do *live coding*. Liminalidade é definida por Di Prospero como “modelos que são, em um nível, reclassificações da realidade e da relação do homem com a sociedade, natureza e cultura” (*Ibdem, idem*):

Liminalidade é apresentada de algumas formas no *live coding*: novos projetos e propostas, a procura por desmistificar a relação com a tecnologia, tornar o código um artesanato ou um produto artístico, mas, mais do que isso, na construção de uma comunidade participativa, uma comunidade coletiva imaginada. Liminalidade do espaço para se expressar e construir várias propostas que suscitam transformações não somente nos campos artísticos e culturais, mas também institucional, a cena do *live coding* envolve construir o mundo inteiro, um mundo da arte nos termos de Becker (1982). De acordo com o autor, aquele que colabora na produção de uma obra de arte não o faz a partir do nada, mas em acordos passados ou costumes, convenções, que geralmente cobrem as decisões tomadas, e isso torna as coisas mais simples (...), o caso do *live coding* contribui amplamente para a aceitação da

<sup>1</sup> Tradução de *Live coding is an improvisatory artistic technique. It can be employed in many different performative contexts: dance, music, moving images and even weaving. I have concentrated my attention on the music side, which seems to be the most prominent.*

mudança como uma constante, dentro de um quadro em que a expressão artística é um processo em vez de produto acabado. (*Ibidem, idem*).<sup>2</sup>

Di Prospero faz um apontamento (destacado em negrito na citação acima) que será trabalhado no [Capítulo 3](#). O *live coding* não é uma prática emergente sem respaldo em práticas musicais anteriores.

É interessante destacar o tratamento etnográfico e antropológico dados por Mori e Di Prospero. Esta abordagem é recente no Programa de Investigação do *livecoding*, e permite interpretá-lo como um ritual urbano, praticado por um grupo social. Como ritual, indivíduos buscam acessar um estado subjetivo onde limites conceituais estão em um espaço de transição (liminaridade). Isto é, limites concretos entre Música e Ciências da Computação, cânones musicais (erudito e *jazz*) às músicas-populares massivas (Sá, 2009), como também dos espaços universitários às Casas Noturnas, se tornam cada vez mais abstratos.

A forma deste ritual depende do espaço físico onde uma improvisação é realizada. Alguns casos acontecem salas de concerto, em salas de universidade, em uma casas noturna, ou em um ambiente informal. Por outro lado, estes espaços podem ser simplesmente virtuais, como o estudo de caso apresentado na [Capítulo 4](#).

Desta forma, o trabalho parte da concepção de um *Universo de Possibilidade* do que é *livecoding* (ou *live coding*); investiga um *Espaço Conceitual* do *livecoding* a partir de uma perspectiva histórica, e reduz o *Universo de possibilidades* para um espaço conceitual, que enfatize um ritual específico, a improvisação *Study in Keith* de Andrew ??).

## 1.1 A questão da tradução

Mori e Di Prospero, bem como diversos outros autores, denominam a prática com dois termos separados (*live coding*). Tomando esta separação, o prefixo *live* é traduzido como “ao vivo”, e o sufixo *coding* como “codificar”. Uma performance cujo ato é improvisar leis ou fórmulas dispersas.

A tradução falha em identificar questões-satélites (o que é codificado?). Para identificar algumas palavras-chaves, uma compilação dos anais do ICLC 2015 ([MCLEAN](#)

<sup>2</sup> Tradução nossa de: *Liminality is present in some ways in live coding: new projects and proposals, the search to demystify the relationship with technology, making the code a craft or artistic product, but, more than anything, in the construction of its "participatory community", a collectively imagined community. Liminality of space to express themselves and build various proposals raises transformations not only in the artistic or cultural field but also institutional, the live coding scene involves building an entire world, an art world in terms of Becker (Becker 1982). According to the author, who cooperates in producing a work of art do not do it from nothing but rest on past agreements or custom / conventions, which usually cover the decisions to be taken, and this makes things simpler(...), the case of live coding broadly contributes to the acceptance of change as a constant, within a framework in which artistic expression is a process rather than finished product.*



Tabela 1 – Tabela de classes qualitativas de termos utilizados nos anais do ICLC2015, agrupados por funções textuais.

Número Qualitativo/Função	0	1	2	3	4	5	8	9
<b>Pessoas</b>	-	Collins, Blackwell, McLean, Grossi	-	-	-	-	-	-
<b>Aplicativos</b>	-	SuperCollider, Gibber, SonicPi	-	-	-	-	-	
<b>Verbos</b>	take, see, shared, networked, explore, made	make, provide, writing, solving, making	used	using, coding	performer	-	-	-
<b>Adjetivo ou numeral, ordinal</b>	less, open, potential, similar, important, cognitive, virtual	first, real, electronic, visual, ensemble, possible, free, livecoding, aspect	musical, many	new, one	-	-	live	-
<b>Substantivo</b>	Browser, point, approach, order, node, collaborative, number, source, present, community, server, framework, orchestra, digital, level, kind, type, memory, analysis, line, body, concept, technology, working, org, current, show, mean, end, processes, people, international	University, conference, proceedings, network, interface, environment, text, form, context, musician, space, paper, program, audience, function, change, control, human, laptop, interaction, structure, part, session, tool, result, create, object, case, algorithm, value, development, material, set, technique, parameter, idea, screen, video, application, support, composition, piece, knowledge, feature, cell, activity, art, action, information, method, web, rule, group, need, particular, project, allow, collaboration, programmer, member, play, output	use, coder, process, state, example, way, software, research, problem, experience, design, improvisation, different, machine, pattern, audio	work, instrument	system, computer, user, language, time, practice, sound	programming	performance, code	“live coding”, music



do *live coding*, não obtivemos resultados que retornassem, por exemplo, a palavra *hearing*. Isso é significativo, e no [Capítulo 3](#), exploro estas ações sob a ótica da Música de Processos de Steve Reich.

Já os adjetivos destacam características da prática, onde *live* é a palavra-chave. Como será observado no [Capítulo 3](#), a ação pode ocorrer em uma sala de concerto, um espaço público ou em uma casa noturna. Palavras como *electronic*, podem sugerir tanto uma música “eletroacústica”, quanto gêneros de música para dançar. *Visual* remete a uma característica tão fundamental quanto a Música. Um sem número de performances utilizam a projeção de telas de computadores como dispositivo de “transparência”; isto é, uma ideologia de justificação do ato de improvisação. *Ensemble* destaca uma a natureza de grupos. Poucas performances *solo* são realizadas se comparadas às performances de *duos*, *trios*.

Ainda no [Capítulo 3](#), o *live coding* se apropria da Música como Processo de Steve Reich (1968) e da Música Generativa de Brian Eno (1978) para justificar processos musicais com o virtuosismo de codificação<sup>5</sup>

Palavras como *university*, *research* e *technology*, e *laptop* acusam não apenas uma prática artística, mas um Programa de Investigação Científica próprio. A esfera de pesquisa acadêmica permitiu ramos de desenvolvimento com linguagens de programação, cognição, inteligência artificial, semiologia, performance musical (improvisação), e mais recentemente, antropologia, conferindo à produção de *live coding* espécie de autenticidade acadêmica.

## 1.2 Sistemas Criativos, e o Universo de Conceitos

Uma sessão de *live coding* é uma improvisação. Em seu artigo “*Music improvisation and creative systems*”, Alex McLean (2006), articula a improvisação musical como uma atividade passível de formalização lógica.

Este trabalho tem respaldo em outro, o artigo “*A preliminary framework for description, analysis and comparison of creative systems*”, de Wiggins (2006), e “*Improvisation: methods and models*” de Jeff Pressing (1987). O primeiro realiza definições formais do livro “*The creative mind*” de Margaret Boden (1990).

A relação entre os trabalhos é uma proposta de McLean para “traduzir o Modelo de Improvisação para funcionar em um modelo computacional” (McLean, 2006, p. 5)<sup>6</sup>. De fato, ao final de seu artigo, realiza uma adaptação entre a *Ferramenta de Estruturação de Sistemas Criativos*<sup>7</sup> de Wiggins (ver [subseção 1.2.1](#)), e o *Modelo de Improvisação* de

<sup>5</sup> Neste trabalho criticamos esta visão. Ao que nos parece, ocorre uma confusão entre *comportamentos criativos históricos* e, no caso do *livecoding* *emph*comportamentos criativos psicológicos.

<sup>6</sup> Tradução parcial de *The above analysis highlights a number of areas for consideration while translating the IM to a working computer model.*

<sup>7</sup> *Creative System Framework.*

Pressing (ver [subseção 1.2.3](#)).

Para esclarecer alguns pontos de ambos, discutimos o *Universo de Possibilidades* de Wiggins (ou *Universo de Conceitos para McLean*) na [subseção 1.2.2](#), e o *Modelo de Improvisação* de Pressing na [subseção 1.2.3](#).

### 1.2.1 Sistemas Criativos

O preceito para o CSF de McLean parte da relação entre os *Processos criativos/Espaço conceituais* de Boden, os *Sistemas Criativos* de Wiggins, “Boden concebe o processo de criatividade como uma identificação e/ou localização de novos objetos conceituais em um espaço conceitual” (WIGGINS, 2006, p. 450)<sup>8</sup>.

Boden realiza duas taxonomias da criatividade. A primeira separa os comportamentos criativo em tipos H (históricos) e P (psicológicos). A segunda, separa o comportamento criativo em exploradores, e transformacionais.

Comportamentos criativos do tipo H são aqueles tipos de conceitos historicamente novos, nunca inventados antes. Processos criativos do tipo P são os novos conceitos elaborados por uma pessoa. Por exemplo, os processos inventados em *It’s gonna rain* de Steve Reich, *I’m sitting in a room* de Alvin Lucier ou as técnicas similares publicadas por Pauline Oliveros (1969) são comportamentos P-criativos que se tornaram H-criativos. Por outro lado consideramos, neste trabalho, os comportamentos criativos do *livecoding* ainda como P-criativos. Carecem de tempo suficiente para serem considerados H-criativos.

Porém Wiggins poderia criticar ambas as visões (de Boden e a realizada no parágrafo anterior) da seguinte forma: “pode ser possível, por exemplo, um comportamento ser P-criativo em uma sociedade, mas H-criativo em outra” (*Ibdem, idem*).<sup>9</sup>

A segunda taxonomia, que separa em comportamentos exploradores e transformacionais, lidam diretamente com o espaço conceitual. Se no primeiro caso este espaço é construído progressivamente com a elaboração de P-conceitos (o que pode ser observado em crianças), o transformacional é um comportamento que modifica o espaço conceitual existente, levando a filtrar quais são H-conceitos.

Para lidar formalmente com os Espaços Conceituais, Wiggins define “criatividade”, “computação criativa”, “sistemas criativos” e “comportamentos criativos”. São apresentado na [Tabela 2](#)

<sup>8</sup> Tradução de *Boden conceives the process of creativity as the identification and/or location of new conceptual objects in a conceptual space.*

<sup>9</sup> Tradução de *it would be possible, for example, for a creative behaviour to be only P-creative in one society, but H-creative in another.*

Tabela 2 – Definições formais de criatividade por Wiggins (2006, p. 451)

Criatividade	“A performance de tarefas que, quando executados por um humano, são consideradas criativas” <sup>10</sup>
Computação criativa	“O estudo e suporte, através de meios e métodos computacionais, do comportamento exibido por sistemas naturais e artificiais, que são considerados criativos”. <sup>11</sup>
Sistemas criativos	“Uma coleção de processos, naturais ou automáticos, que são capazes de alcançarem ou simularem comportamentos que em humanos seria considerado criativo”
Comportamento Criativo	“Um ou mais dos comportamentos exibidos por um sistema criativo” <sup>12</sup>

### 1.2.2 Ferramenta de Estruturação de Sistemas Criativos

O universo,  $\mathcal{U}$ , é um espaço multidimensional, no qual dimensões são capazes de representar qualquer coisa, e todos os possíveis conceitos distintos correspondentes àqueles pontos em  $\mathcal{U}$  (...) Para tornar a proposta um espaço-tipo possível, permitirei que  $\mathcal{U}$  contenha todos os conceitos abstratos, bem como os concretos, e que é possível representar os artefatos tanto completos e incompletos (WIGGINS, 2006, p. 451).<sup>13</sup>

Para diferenciar o Universo  $\mathcal{U}$ , do Espaço conceitual  $\mathcal{C}$ , Wiggins esclarece que Boden não reconhece de forma explícita  $\mathcal{U}$ , sendo que “ela borra a distinção entre as regras que determinam a adesão do espaço (...) e outras disposições que possam permitir a construção e/ou detecção de um conceito representado por um ponto no espaço” (WIGGINS, 2006, p. 451). Portanto, diferentes espaços conceituais estão contidos em  $\mathcal{U}$ ;

Nas palavras de McLean (2006), regras que validam a concepção do conceito no universo, são representadas como um conjunto  $\mathcal{R}$ . Regra que determinam estratégia de detecção (transversais) de um conceito no universo, são representados como o conjunto  $\mathcal{T}$ . Para lidar com estas regras, é necessária uma *linguagem*  $\mathcal{L}$  para expressá-las. Nas palavras de Wiggins: “um conjunto de sequências compostas por algum alfabeto  $\mathcal{A}$ ”<sup>14</sup>

A Tabela 4 ilustra os termos até agora trabalhados por McLean:

### 1.2.3 O modelo de improvisação

McLean realiza uma comparação, admitindo que ela não é imediata, do *Universo de possibilidades* de Wiggins com o *Modelo de Improvisação* de Jeff Pressing (1987). No entanto, McLean argumenta que

Pressing discute comportamento criativo no contexto do Modelo de Improvisação, e de fato é parte da Ferramenta de Estruturação de Sistemas Criativos. (...) Durante a transferência de notação do Modelo de Improvisação para a Ferramenta de Sistemas Criativos, nós consideramos

<sup>13</sup> Tradução de *The universe,  $\mathcal{U}$ , is a multidimensional space, whose dimensions are capable of representing anything, and all possible distinct concepts correspond with distinct points in  $\mathcal{U}$ . (...) To make the proposal as state-spacelike as possible, I allow that  $\mathcal{U}$  contains all abstract concepts as well as all concrete ones, and that it is therefore possible to represent both complete and incomplete artefacts*

<sup>14</sup> Tradução de *it as the set of all sequences composed of some alphabet,  $\mathcal{A}$ .*

Tabela 3 – Definições formais do Universo de possibilidades de Wiggins (2006), ou Universo de Conceitos por McLean (2006).

Representação	Nome	Significado
$c$	Conceito	Uma instância de um conceito, abstrato ou concreto (WIGGINS, 2006).
$\mathcal{U}$	Universo de Conceitos	Superconjunto não restrito de conceitos. (WIGGINS, 2006). “Um universo de todos conceitos possíveis” (McLean, 2006) <sup>15</sup>
$\mathcal{L}$	Linguagem	Linguagem utilizada para expressar regras.
$\mathcal{A}$	Alfabeto	Alfabeto da linguagem que contém caracteres apropriados
$\mathcal{R}$	Regras de validação	Validam os conceitos em um universo, se apropriados ou não para o espaço trabalhado.
$[[\cdot]]$	Função de interpretação	“Uma função parcial de $\mathcal{L}$ para funções que resultam em números reais entre $[0, 1]$ (...) 0.5 [ou maior] significa uma verdade booleana e menos que 0.5 significa uma falsidade booleana; a necessidade disso para valores reais se tornará clara abaixo” (WIGGINS, 2006, p. 452) <sup>16</sup>
$[[\mathcal{R}]]$	Regras de validação	“Uma função que interpreta $\mathcal{R}$ , resultando em uma função indicando aderência ao conceito em $\mathcal{R}$ ” <sup>17</sup>
$\mathcal{C} = [[\mathcal{R}]](\mathcal{U})$	Espaço Conceitual	“Todos espaços conceituais são um subconjunto não-estrito de $\mathcal{U}$ ” <sup>18</sup> . Um subconjunto contido em $\mathcal{U}$ (WIGGINS, 2006). Uma função que interpreta $\mathcal{R}$ , resultando em uma função que indica aderência ao conceito em $\mathcal{R}$ <sup>19</sup>
$\mathcal{T}$	Regras de detecção	“Regras definidas dentro de $\mathcal{L}$ para definir estratégias transversais para localizar conceitos dentro de $\mathcal{U}$ ” (McLean, 2006) <sup>20</sup>
$\mathcal{E}$	Regras de qualidade	“(...) conjunto de regras que permitem-nos avaliar qualquer conceito que nós encontramos em $\mathcal{C}$ e determinar sua qualidade, de acordo com critérios que nós consideramos apropriados” (WIGGINS, 2006, p.453) <sup>21</sup> “Regras definidas dentro de $\mathcal{L}$ para avaliar a qualidade ou a desejabilidade do conceito $c$ ” (McLean, 2006) <sup>22</sup>
$\langle\langle\langle \mathcal{R}, \mathcal{T}, \mathcal{E} \rangle\rangle\rangle$	Função de interpretação	Uma regra necessária para definir o espaço conceitual, “independentemente da ordem, mas também, ficcionalmente, enumerá-los em uma ordem particular, sob o controle de $\mathcal{T}$ – isto é crucial para a simulação de um comportamento criativo de um $\mathcal{T}$ particular (WIGGINS, 2006) <sup>23</sup> . “Uma função que interpreta a estratégia transversal $\mathcal{T}$ , informada por $\mathcal{R}$ e $\mathcal{E}$ . Opera sobre um subconjunto ordenado de $\mathcal{mathcal{U}}$ (do qual tem acesso randômico) e resulta em outro subconjunto ordenado de $\mathcal{U}$ .” <sup>24</sup>

improvisação musical de uma maneira clara e temos uma language comum na qual comparar com outros modelos<sup>25</sup>.

<sup>25</sup> Tradução de *However Pressing does discuss creative behaviour in the context of the IM, and indeed the CSF is in part. (...) In transferring the IM to the notation of the CSF we may consider music improvisation in a clearer manner and have a common language in which to compare it with other models.*

Um sumário sobre o modelo de improvisação é apresentado na ??.

Tabela 4 – Definições formais do Modelo de improvisação de Jeff Pressing (1987), segundo McLean (2006, p. 2).

Representação	Significado
$E'$	A cluster of sound events
$K'$	A sequence of E event clusters, where event cluster onsets do not overlap with those of a following one
$I'$	An improvisation, partitioned by interrupts into a number of K sequences
$R'$	An optional referent, such as a score or stylistic norm
$G'$	A set of current goals
$M'$	Long term memory
$O'$	An array of objects
$F'$	An array of objects Features
$P'$	An array of Process

### 1.3 Exemplo concreto da multiplicidade em um universo de possibilidades

Durante a pesquisa, investigamos um campo específico, mas que não será utilizado como trabalho final. Por outro lado, permitiu visualizar os conceitos  $c$  de universo de conceitos  $U$  em uma mídia social (*Soundcloud*).

As ??, ??, ilustram a multiplicidade de conceitos  $c$ . Agrupados por data, país, cidade e *hashtags* que “delimitam” um gênero musical, possibilitaram reduzir um pouco o estudo, mas não completamente.

Os seguintes os termos delimitados foram: *a) livecoding/live-coding*: dados coerentes com a bibliografia pesquisada<sup>26</sup>; *b) algorave/algopop*: parte considerável da produção do *livecoding* realizada em ambientes noturnos, informais ou de entretenimento (possue relação com o elemento dança); *c) bytebeat*: parte considerável de uma técnica de programação musical descrita pela primeira vez por Heikkilä (2011) e aplicada no *livecoding*, isto é, apenas um fragmento dessa produção pode se encaixar como *livecoding* (um desses programas é o *Wavepot*); *d) algorithmic music*: música algorítmica, seguindo a subdivisão Música Gerada por Computador, ou *Computer Generated Music* (COPE, 2008).

Os dados levantados são de janeiro e fevereiro de 2015; não realizamos mais levantamentos. Os motivos foram: *i)* a dificuldade de análise das assimetrias observadas, que requerem maior experiência com técnicas estatísticas. *ii)* os dados ilustram as variedades

<sup>26</sup> A exclusão do termo *live code/live coding* foi feita pois a separação criava uma ambiguidade de busca no *Soundcloud*, isto é, *live* poderia não se referir ao que pesquisamos por *livecoding*.

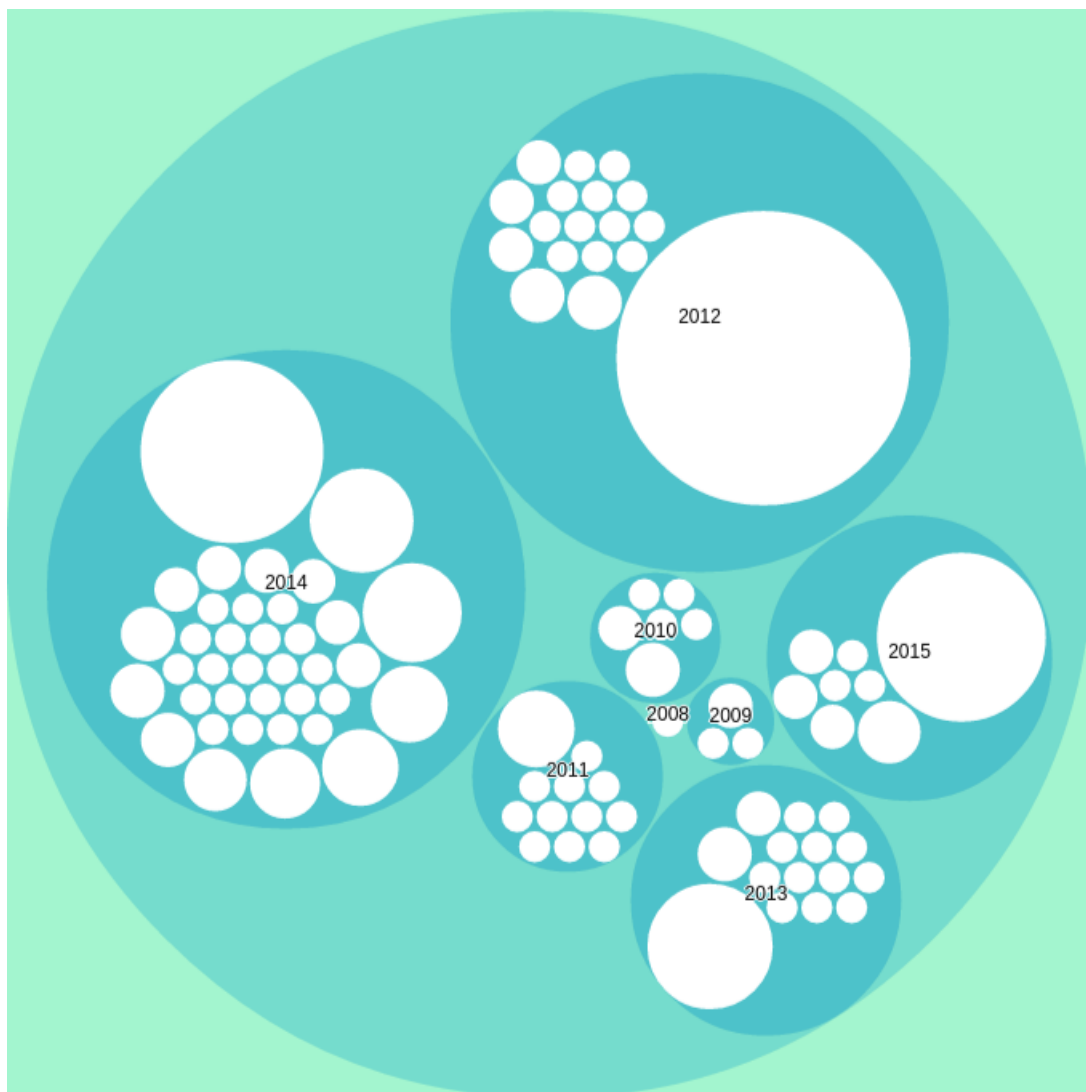


Figura 2 – Empacotamento de informações a respeito de gênero musical a partir de anos de produção

de categorizações musicais no *live coding*. Pode ser interessante observar se tais variedades se aplicam em um único *live coder*.

Antes de discutir três casos específicos de um *live coder* (Sorensen), contextualizamos os *live coding* no Capítulo 3. Alguns precedentes históricos do *live coding*, como *GROOVE*, o compositor *Pietro Grossi*, a *Live Computer Music* da Baía de São Francisco entre os anos 70 e 80 nos Estados Unidos, e a emancipação de uma organização, *TOPLAP*<sup>27</sup>, e o destaque de *live coders* ingleses, foram fatores para descrever, neste trabalho, o *live coding* como técnica de improvisação artística original de um Norte político e econômico; delimitar o *live coding* como um Programa de Investigação Científica em universidades; ou descrever o *live coding* como dispositivo de improvisação para replicação de categorizações musicais.

<sup>27</sup> Disponível em <<http://www.toplap.org>>.

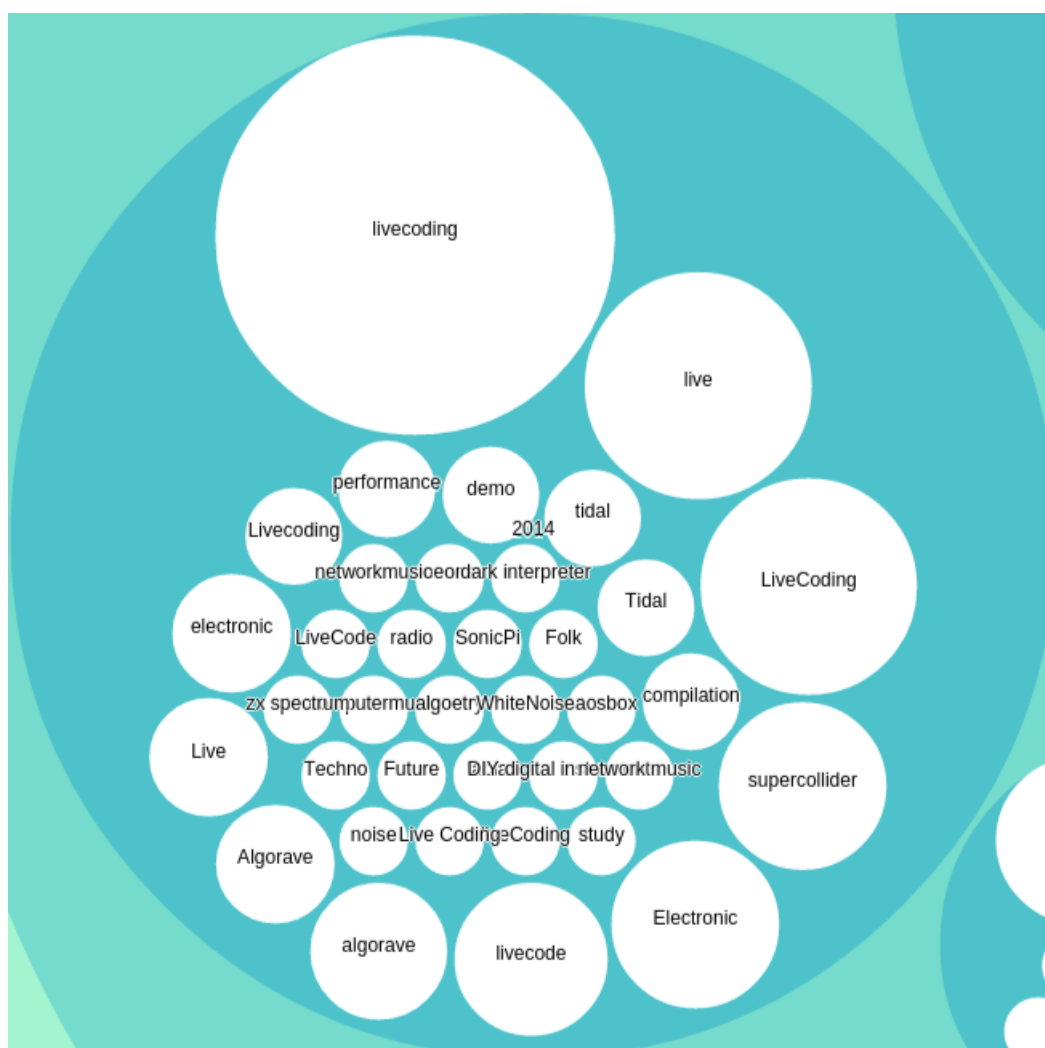


Figura 3 – Detalhamento de informações a respeito de gênero musical a partir de anos de produção

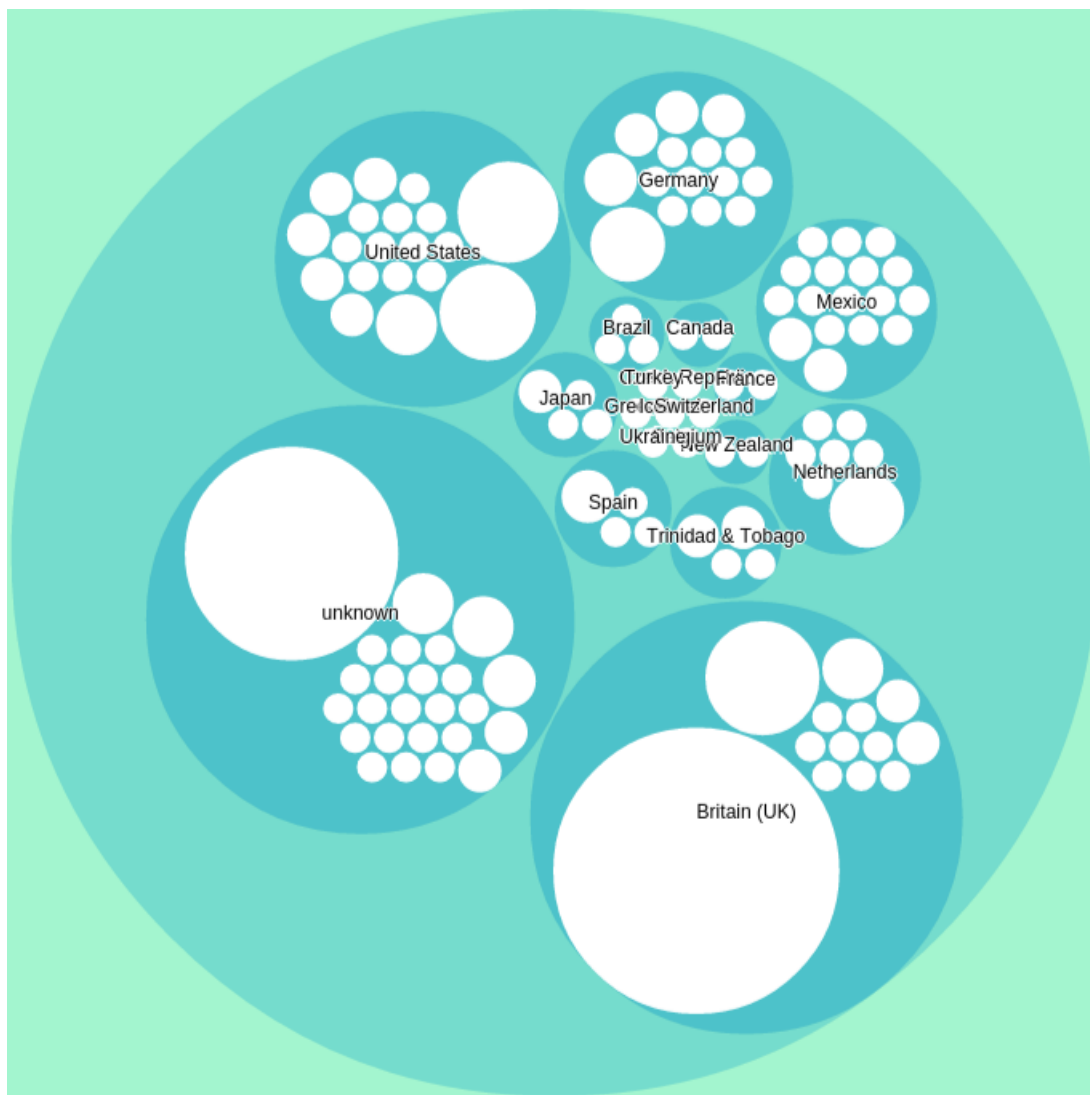


Figura 4 – Empacotamento de informações a respeito de gênero musical a partir de países onde ocorreram as produções



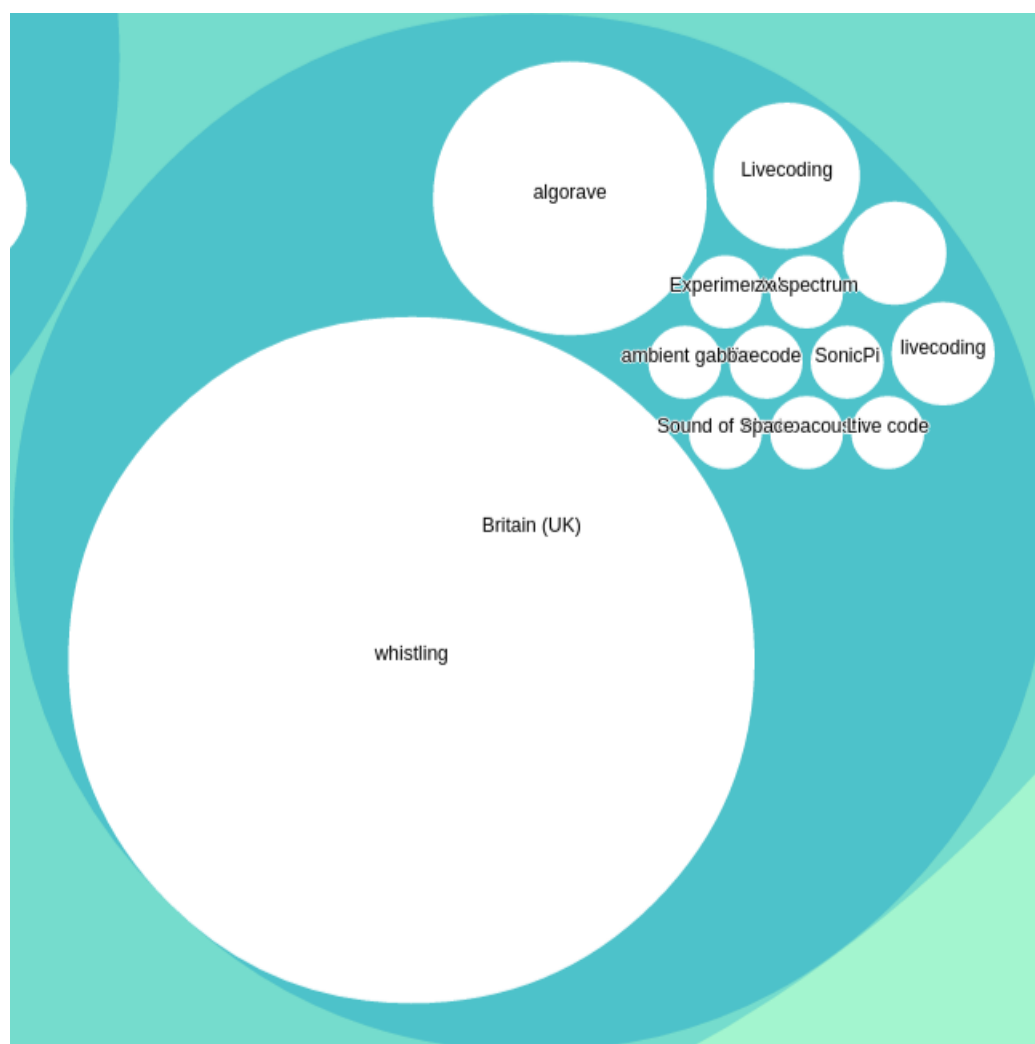


Figura 5 – Detalhamento de informações a respeito de gênero musical a partir de países onde ocorreram as produções.



## 2 Metodologia

É interessante notar que o o PIC do *livecoding* realiza um acesso constante às suas liminaridades.

Este acesso levou, nas primeiras críticas a este trabalho, aos seguintes questionamentos: o *livecoding* apenas se configura como tal se projetarmos um código textual para um público, e improvisarmos neste código? A utilização de linguagens visuais, como *PD* ou *Max/Msp*, ao vivo (*live patching*) são um caso à parte ou devem ser incluídas no objeto de pesquisa? É pertinente chamar de *live coding* uma performance que utiliza *DAWs*<sup>1</sup> *Live!Ableton*? Utilizar dispositivos diferentes do *laptop*, como corpos dançantes, em associação com dispositivos digitais, é *livecoding*? Se tais perguntas apenas expõem as multiplicidades do *livecoding*, dificultam uma análise de caso.

Uma pergunta mais específica irá delinear o método de pesquisa, tomando por base, um *universo de conceitos do livecoding*  $\mathcal{U}_{livecoding}$ . Este, por sua vez, contém infinitos espaços conceituais  $\mathcal{C}$ , com instâncias de conceitos  $c$ . Para efeito de estudo selecionamos dois espaços conceituais:  $\mathcal{C}_{livecoding}$  e  $\mathcal{C}_{Study\ in\ Keith}$ , como definido na [Equação 2](#).

$$\mathcal{U}_{pesquisa} = [\dots, \mathcal{C}_{livecoding} \cup \mathcal{C}_{Study\ in\ Keith}, \dots] \quad (2.1)$$

Mais especificamente, **O que pode ser pressuposto musicalmente em uma sessão de *livecoding* nomeada *Study in Keith*?** Esta pergunta leva a especificar o espaço conceitual da pesquisa, como definido nas [Equação 4](#).

$$\mathcal{C}_{pesquisa} = \mathcal{C}_{livecoding} \cup \mathcal{C}_{Study\ in\ Keith} \quad (2.2)$$

Por sua vez, os espaços conceituais do *livecoding* e de *Study in Keith* são definidos como uma união entre as regras de validação, de gosto e de localização de conceitos no espaço dado:

$$\mathcal{C}_{livecoding} = \langle \langle \langle \mathcal{R}_{livecoding}, \mathcal{T}_{livecoding}, \mathcal{E}_{livecoding} \rangle \rangle \rangle \quad (2.3)$$

$$\mathcal{C}_{Study\ in\ Keith} = \langle \langle \langle \mathcal{R}_{Study\ in\ Keith}, \mathcal{T}_{Study\ in\ Keith}, \mathcal{E}_{Study\ in\ Keith} \rangle \rangle \rangle \quad (2.4)$$

---

<sup>1</sup> *Digital Audio Workstation.*



O mesmo processo pode ser aplicado à *Study in Keith*, como na [Equação 2.1](#).

$$\mathcal{C}_{Study\ in\ keith} = [\mathcal{S} \sqcap |_{Study\ in\ Keith}, \mathcal{A}_{\sqrt{\sqrt{Study\ in\ Keith}}}, \mathcal{V} \sqcap \nabla \lfloor_{Study\ in\ Keith}, \mathcal{A} \lceil |_{Study\ in\ Keith}, \mathcal{S} \sqcap \lfloor_{Study\ in\ Keith}, \dots] \mathcal{S} \sqcap |_{Study\ in\ Keith} \quad (2.6)$$

Por sua vez, *Study in Keith* é um outro espaço conceitual que inclui o pianista e compositor Keith Jarret como sujeito. Por outro lado, se a necessidade de estudo histórico foi levantada, realizaremos uma pequena contextualização dos seus concertos *Sun Bear* (1976-1979).

### 3 Trabalhos Relacionados

## 3.1 Predecessores

Na [subseção 3.1.1](#) descrevo um trabalho de [Mathews e Moore \(1970\)](#), GROOVE, ainda pouco observado por *live coders*. Seu paradigma composicional é diverso do MUSIC N, e o primeiro de Mathews com reflexões nos aspectos performáticos, semelhantes aos pesquisados neste trabalho.

[Mori \(2015b\)](#) descreve um caso prematuro de *live coding* na Itália, com o compositor Pietro Grossi (1917-2002). Divergente em algumas das propostas de Max Mathews, sacrificou a questão timbrística para trabalhar na questão performática. Esta abordagem será trabalhada na [seção 3.1.2](#).

[McLean e Wiggins \(2009\)](#) descrevem, no final dos anos 70 e dos anos 80, atividades dos grupos *The Hub* e *The League of Automatic Composers* como fundamentais para o entendimento histórico do *live coding*. Será explicado na [seção 3.1.3](#).

Como um breve parêntese, sugiro falar sobre a compilação JIT ([AYCOCK, 2003](#)). Este é um personagem sócio-técnico fundamental para que o *live coding* fosse possível. Será descrito na [subseção 3.1.4](#).

Para [McLean e Wiggins](#), o *live coding* não possui sua identidade cultural até a emergência da organização TOPLAP. Na [subseção 3.1.5](#) proponho a revisão de um trecho da publicação “*Live Algorithm Programming and Temporary Organization for its Promotion*”.

Além deste primeiro manifesto, existe outro de grande importância, “Show us your screens”, que define alguns cânones do *live coding*. Na [subseção 3.1.6](#) tratarei do caso específico.

### 3.1.1 GROOVE

Conectando as práticas da imediaticidade, foi possível concluir que o GROOVE ([MATHEWS; MOORE, 1970; NUNZIO, 2010](#)) é um precedente mais antigo do *live coding* (até que outro programa seja descoberto). Seu desenvolvimento iniciou em 1968 na *Bell Labs*. Segundo o próprio Mathews, o funcionamento do sistema oferece algumas possibilidades a partir de três conceitos: criação, *retroalimentação* e *ciberficação*.

O primeiro conceito foi implementado com um sistema de arquivos, onde as funções criadas no processo criativo são memorizadas, e podem ser editadas.

O segundo conceito se relaciona com o terceiro. Tange a imediatidade do fazer musical, e potencialmente, de uma necessidade de improvisação. Esta abordagem é contemporânea às técnicas de composição baseadas por *retroalimentação*, como as descritas por Pauline [Oliveros \(1969\)](#). Não sabemos se a primeira foi inspirada na segunda, mas o paralelo não deixa de ser significativo e importante para a concepção do que é o *live*



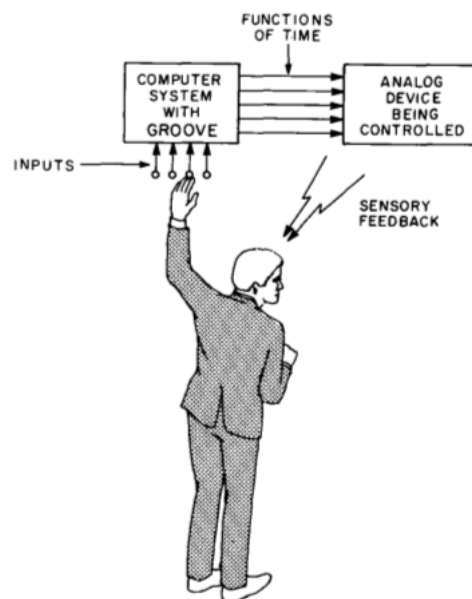


Fig. 1. Feedback loop for composing functions of time

Figura 6 – Esquema de concepção do projeto GROOVE descrito no artigo homônimo por Max Mathews **Fonte:** (MATHEWS; MOORE, 1970).

*coding:*

O GROOVE provê oportunidades para uma retroalimentação imediata de observações dos efeitos das funções temporais para as entradas do computador, que compõem a função. No modo de composição do sistema GROOVE, um ser humano está em um ciclo de retroalimentação, como mostrado na figura 1 [Figura 6]. Assim ele é capaz de modificar as funções instantaneamente como um resultado de suas observações daqueles efeitos. (MATHEWS; MOORE, 1970, p. 715) <sup>1</sup>

O terceiro conceito observa a existência de uma relação entre um humano e uma máquina. De certa forma, o sistema passa por um processo de ciberficação, considerado por Mathews como um conceito nebuloso, e chamado de *engenharia humana*. Consistiu na observação de um tempo diferencial entre o que o(a) musicista cria e o que edita. De certa forma, é o conceito central que permite que os dois anteriores fossem possíveis, tange a corporificação de um regente:

O conceito final é mais nebuloso. Desde que o GROOVE é um sistema homem-máquina, a engenharia humana do sistema foi a mais importante. Por exemplo, nós descobrimos que o controle do programa de tempo necessita ser bastante diferente para a composição do que para a edição, e o programa foi modificado de acordo. A engenharia humana adetou toda

<sup>1</sup> Tradução nossa de *GROOVE provides opportunity for immediate feedback from observations of the effects of time functions to computer inputs which compose the function. In the compose mode of the GROOVE system, a human being is in the feedback loop (...) Thus he is able to modify the functions instantaneously as a result of his observations of their effects.*

estrutura do GROOVE de forma que pontuaremos subsequentemente. (...) O intérprete de computador não deve tentar definir todo o som em tempo real. Ao invés, o computador deve ter uma partitura e o intérprete deve influenciar a forma como a partitura é tocada. Seus modos de influência pode ser mais variados do que aqueles que um regente convencional, que pode principalmente controlar o tempo, *loudness* e estilo. (MATHEWS; MOORE, 1970, p. 715-716) <sup>2</sup>

### 3.1.2 Pietro Grossi

Embora pouco conhecido no contexto geral da música européia, o compositor Pietro Grossi foi um dos pioneiros da *Computer Music* Italiana. Diverge do *paradigma* do MUSIC III e suas preocupações timbrísticas. Mas concorda, parcialmente, com a abordagem do GROOVE. Sacrificou o desenvolvimento do timbre desde o início, e para concentrar na performance, utilizou apenas uma forma de onda (pulsos). O primeiro *software* desenvolvido foi o DCMP (*Digital Computer Music Program*) e, segundo (MORI, 2015b), ao usar este programa,

(...) o intérprete era capaz de produzir e reproduzir música em tempo real, digitando alguns comandos específicos e os parâmetros composicionais desejados. O som resultante vinha imediatamente depois da operação de decisão, sem qualquer atraso causado por cálculos. Havia muitas escolhas de reprodução no programa: era possível salvar na memória do computador peças de músicas pré-existent, para elaborar qualquer material sonoro no disco rígido, para administrar arquivos musicais e iniciar um processo de composição automático, baseado em algoritmos que trabalham com procedimentos “pseudo-casuais”. Existia também uma abundância de escolhas para mudanças na estrutura da peça. Um dos mais importantes aspectos do trabalho de Grossi foi que todas as intervenções eram instantâneas: o operado não tinha que esperar pelo computador terminar todas as operações requisitadas, e depois ouvir os resultados. Cálculos de dados e reprodução sonora eram simultâneos. Esta simultaneidade não era comum no campo da *Computer Music* da-quele tempo, e Grossi deliberadamente escolheu trabalhar desta forma, perdendo muito no lado da qualidade sonora. Seu desejo era poder escutar os sons resultantes imediatamente. (MORI, 2015b, p. 126) <sup>3</sup>

<sup>2</sup> Tradução nossa de *The final concept is more nebulous. Since GROOVE is a man-computer system, the human engineering of the system is most important. For example, we discovered that the control of the program time needs to be quite different for composing than for editing, and the program was modified accordingly. Human engineering has affected the entire structure of GROOVE in ways which will be pointed out subsequently. (...) The computer performer should not attempt to define the entire sound in real-time. Instead, the computer should have a score and the performer should influence the way in which the score is played. His modes of influence can be much more varied than that of a conventional conductor who primarily controls tempo, loudness, and style..*

<sup>3</sup> Tradução nossa de *(...) the performer was able to produce and reproduce music in real time by typing some specific commands and the desired composition's parameters. The sound result came out immediately after the operator's decision, without any delay caused by calculations. There were many reproduction choices inscribed in this software: it was possible to save on the computer memory pieces of pre-existing music, to elaborate any sound material in the hard disk, to manage the music archive and to start an automated music composition process based on algorithms that worked with “pseudo-casual” procedures. There were also plenty of choices for piece structure modifications. One of the most important aspects of Grossi's work was that all the interventions were instantaneous: the*

Esta abordagem parte de uma abordagem “preguiçosa” (*lazy*). Grossi dizia sobre si mesmo, como “uma pessoa que está consciente de que o seu tempo é limitado e não quer perder tempo em fazer coisas inúteis ou na espera de alguma coisa quando não é necessário.”<sup>4</sup>. Neste sentido, defendia que o desenvolvimento de novos timbres deveria esperar por melhores implementações. .

O sacrifício do timbre reflete a utilização o computador como uma paródia do piano, ou até mesmo de um violino. Grava em 1967 “Mixed Paganini”<sup>5</sup>, uma execução ultra-virtuosística dos *Capricci op.1* de Niccolò Paganini. Aparentemente pode soar como um pastiche musical. Porém uma escuta mais atenta permite perceber que, a utilização de operações canônicas (inversão, aceleração e retrogradação), executadas no computador Olivetti GE-115 Mori (2015b, p. 126), lembra dicotomias entre o tempo e o ritmo já discutidas por ??).

### 3.1.3 Baía de São Francisco

No final dos anos 70, na cena musical da Baía de São Francisco, uma das atividades de John Bischoff (1949-, aluno de composição de James Tenney and Robert Ashley.) e Tim Perkis era passar horas ajustando uma rede de microcontroladores programáveis KIM-1<sup>6</sup>. Aquele era um momento onde os *happenings* já eram manifestações artísticas consolidadas. Não demorou muito para que o público participasse da atividade:

qual sua  
data de  
nasci-  
mento?

Na primavera de 1979, montamos uma série quinzenal regular de apresentações informais sob os auspícios da *Bay Center for the Performing Arts*. Todos outros domingos à tarde passávamos algumas horas configurando nossa rede de KIMs na sala *Finnish Hall*, na Berkeley, e deixávamos a rede tocando, com retoques aqui e ali, por uma ou duas horas. Os membros da audiência poderiam ir e vir como quisessem, fazer perguntas, ou simplesmente sentar e ouvir. Este foi um evento comunitário de tipos como outros compositores aparecendo, tocando ou compartilhando circuitos eletrônicos que tinham projetado e construído. Um interesse na construção de instrumentos eletrônicos de todos os tipos parecia estar "no ar". Os eventos da sala *Finn Hall* foram feitos para uma cena com paisagens sonoras geradas por computador misturado com os sons de grupos de dança folclórica ensaiando no andar de cima e as reuniões ocasionais do Partido Comunista na sala de trás do edifício velho venerável. A série durou cerca de 5 meses que eu me lembre. (BROWN; BISCHOF, 2013, online)<sup>7</sup>

*operator had not to wait for the computer to finish all the requested operations and then hear the results. Data calculation and sound reproduction were simultaneous. This simultaneity was not common in the computer music field of that time and Grossi deliberately chose to work in this way, losing much on the sound quality's side. His will was to listen to the sound result immediately.*

<sup>4</sup> Tradução nossa de *a person who is aware that his or her time is limited and do not want to waste time in doing useless things or in waiting for something when it is not necessary.*

<sup>5</sup> Disponível em <[https://www.youtube.com/watch?v=ZQSP\\_wF7wSY](https://www.youtube.com/watch?v=ZQSP_wF7wSY)>.

<sup>6</sup> Disponível em <<http://www.6502.org/trainers/buildkim/kim.htm>>.

<sup>7</sup> Tradução nossa de: *In the spring of 1979, we set up a regular biweekly series of informal presentations under the auspices of the East Bay Center for the Performing Arts. Every other Sunday afternoon*

ajustar  
informa-  
ções

Esta experiência foi levada a cabo por Bischof, Perkis, Chris Brown (1953-), Scot Gresham-Lancaster (1954-)<sup>8</sup>, Mark Trayle (1955-)<sup>9</sup> e Phil Stone, que formaram nos anos 80 o grupo *The Hub*, com um primeiro disco lançado em 1989 intitulado *The Hub: Computer Network Music*.

Outro artista, Ron Kuivila, segundo McLean e Wiggins (2009) realiza as primeiras performances de *live coding* em 1985. Um pequeno panorama de suas produções pode ajudar a entender um contexto sonoro. Em 1982 é lançada “Going out with slow smoking”, co-produzida com Nicolas Collins (2002)<sup>10</sup>. Em 1985 são lançadas duas faixas em um disco peças pela *TELLUS #9 – The Audio Casset Magazine*, “Cannon Y for C.N.” e “Parodicals”. Em 1988 Kuivila grava “Linear Predictive Zoo”<sup>11</sup>, parte da *TELLUS #22*<sup>12</sup>.

A primeira performance conhecida de *live coding* foi em 1985, por Ron Kuivila na STEIM em Amsterdã (BLACKWELL; COLLINS, 2005). O *live coding* não possui sua própria identidade cultural até o TOPLAP, a Organização Temporária para a Promoção da Programação Ao Vivo de Algoritmos, formada na *Changing Grammars Workshop* em 2004 (WARD et al., 2004). Mesmo sendo possível pensar em fazer *live coding* sem computadores, através da auto-modificação de composições baseadas em regras, não existe evidência que isto foi feito antes da invenção dos computadores. Parece que foi necessária a invenção da interpretação dinâmica de códigos para o *live coding* aparecer como possível ou talvez desejável (McLean; WIGGINS, 2009, p. 1-2)<sup>13</sup>

inserir  
mais  
coisas

Embora uma breve menção de Kuivila seja realizada, é enfatizado o fato de que os construtos sócio-técnicos dos anos 80 ainda não poderiam articular o *live coding* como conhecemos. Isso só foi possível com alguns desenvolvimentos técnicos e organizações de sujeitos. Como a compilação JIT (AYCOCK, 2003) em *softwares* musicais; e a emancipação

*we spent a few hours setting up our network of KIMs at the Finnish Hall in Berkeley and let the network play, with tinkering here and there, for an hour or two. Audience members could come and go as they wished, ask questions, or just sit and listen. This was a community event of sorts as other composers would show up and play or share electronic circuits they had designed and built. An interest in electronic instrument building of all kinds seemed to be "in the air." The Finn Hall events made for quite a scene as computer-generated sonic landscapes mixed with the sounds of folk dancing troupes rehearsing upstairs and the occasional Communist Party meeting in the back room of the venerable old building. The series lasted about 5 months as I remember.*

<sup>8</sup> Aluno de Darius Milhaud, John Chowning, Robert Ashley e Terry Riley.

<sup>9</sup> Aluno de Robert Ashley.

<sup>10</sup> Áudio disponível em <<http://www.nicolascollins.com/slowsmoketracks.htm>>.

<sup>11</sup> Disponível em <<https://www.youtube.com/watch?v=DZ5pUUXqkMc>>

<sup>12</sup> Disponível em <<http://www.discogs.com/Various-False-Phonemes/release/785540>>.

<sup>13</sup> Tradução nossa de: *The earliest known live coding performance was in 1985, by Ron Kuivila at STEIM in Amsterdam (Blackwell and Collins, 2005). Live coding did not get its own cultural identity until TOPLAP, the Temporary Organisation for the Promotion of Live Algorithm Programming was formed at the Changing Grammars workshop in Hamburg in 2004 (Ward et al., 2004). Although it is possible to do live coding without computers, through self-modifying rule-based composition, there is no evidence that this was done before the invention of computers. It would seem that it required the invention of dynamic code interpretation for live coding to appear possible or perhaps desirable.*

da organização TOPLAP. Antes de descrever o JIT e o TOPLAP, sugiro pensar em outras proto-histórias.

### 3.1.4 Just In Time (JIT)

Passageiro para o motorista: leve-me ao número 37. Eu te digo o nome da rua quando chegarmos lá.<sup>14</sup>

A sentença acima é uma “piada de um professor austríaco” (*idem, ibidem*), e descreve como este paradigma de programação em tempo-real funciona. Segundo Aycock (2003), o primeiros programas JIT foram Genesis (com base no LISP, 1960), LC<sup>2</sup> (*Language for Conversational Computing*, 1968) e APL (1970). Este último tinha duas novidades técnicas, a partir dos termos *drag-along* e *beating*; estes são hoje chamados de *lazy evaluation* (avaliação preguiçosa).

O *SuperCollider* foi o primeiro dos *softwares* descritos na introdução deste trabalho que implementou a avaliação preguiçosa. A documentação oferece uma descrição de como isso pode funcionar durante uma performance de *live coding*:

Para programação interativa, pode ser útil ser capaz de usar algo antes de estar ali – isso faz o pedido de avaliação ser mais flexível e permite postergar decisões para um outro momento. Algumas preparações geralmente tem que ser feitas (...) Em outras situações este tipo de preparação não é suficiente, por exemplo se alguém quer aplicar operações matemáticas em sinais de processos sendo executados no servidor [do *SuperCollider*]<sup>15</sup>

Atualmente, esta técnica têm sido largamente implementada para navegadores de internet (ROBERTS; WAKEFIELD; WRIGHT, 2013). Programas como Gibber<sup>16</sup> (ROBERTS; KUCHERA-MORIN, 2012) e *wavepot*<sup>17</sup> são exemplares. Durante a pesquisa, foi desenvolvido em parceria com o pesquisador Flávio Schiavonni um ambiente JIT, inspirado no GROOVE. Enquanto não foi publicado um artigo explicativo, anexamos ele no ??.

<sup>14</sup> Tradução nossa de *Passenger to taxidriver: take me to number 37. I'll give you the street name when we are there..* Disponível em <<http://doc.sccode.org/Overviews/JITLib.html>>.

<sup>15</sup> Tradução nossa de *For interactive programming it can be useful to be able to use something before it is there - it makes evaluation order more flexible and allows to postpone decisions to a later moment. Some preparations have to be done usually - like above, a reference has to be created. In other situations this sort of preparation is not enough, for example if one wants to apply mathematical operations on signals of running processes on the server..* Disponível em <[http://doc.sccode.org/Tutorials/JITLib/jitlib\\_basic\\_concepts\\_01.html](http://doc.sccode.org/Tutorials/JITLib/jitlib_basic_concepts_01.html)>

<sup>16</sup> Disponível em <<http://gibber.mat.ucsb.edu/>>.

<sup>17</sup> Disponível em <<https://www.wavepot.com>>.

### 3.1.5 Live Algorithm Programming and Temporary Organization for its Promotion

Na ??, Blackwell e Collins (2005) comenta a emancipação de um grupo conhecido como TOPLAP. Este acrônimo é de difícil definição. Deriva do manifesto “*Live Algorithm Programming and Temporary Organization for its Promotion*” Ward et al. (2004). No Wiki do site oficial<sup>18</sup>, a cada visita, cada letra é substituída por palavras randômicas, criando diferentes significados<sup>19</sup>.

Este manifesto expõe os ambientes de performance, bem como alguns ritos técnicos do improvisador. Espaços de Música Eletrônica de Pista se misturam com a Música algorítmica. e Música de processos. Em outras palavras, um fenômeno onde produtores e DJs se misturam aos universitários.

O *Livecoding* permite a exploração de espaços algorítmicos abstratos como uma improvisação intelectual. Como uma atividade intelectual, pode ser colaborativa. Codificação e teorização podem ser atos sociais. Se existe um público, revelar, provocar e desafiar eles com uma matemática complexa se faz com a esperança de que sigam, ou até mesmo participem da expedição. Estas questões são, de certa forma, independentes do computador, quando a valorização e exploração de algoritmo é que importa. Outro experimento mental pode ser encarado com um DJ ao vivo codificando e escrevendo uma lista de instruções para o seu *set* (realizada com o iTunes, mas aparelhos reais funcionam igualmente bem). Eles passam ao HDJ [ *Headphone Disk Jockey* ] de acordo com este conjunto de instruções, mas no meio do caminho modificam a lista. A lista está em um retroprojeto para que o público possa acompanhar a tomada de decisão e tentar obter um melhor acesso ao processo de pensamento do compositor. (WARD et al., 2004, p. 245)<sup>20</sup>

inserir  
mais  
informa-  
ções ou  
discus-  
sões

O manifesto “*Live Algorithm Programmin and Temporary Organization for its Promotion*” (WARD et al., 2004) fornece informações a respeito de comportamentos sociais e gostos musicais:

<sup>18</sup> Disponível em <[http://toplap.org/wiki/Main\\_Page](http://toplap.org/wiki/Main_Page)>.

<sup>19</sup> Deparamo-nos, por exemplo como *Transdimensional Organisation for the Pragmatics of Live Algorithm Programming*, *Terrestrial Organisation for the Proliferation of Live Artistic Programming*, *Temporal Organisation for the Proliferation of Live AudioVisual Programming* e outros.

<sup>20</sup> Tradução nossa de: *Live coding allows the exploration of abstract algorithm spaces as an intellectual improvisation. As an intellectual activity it may be collaborative. Coding and theorising may be a social act. If there is an audience, revealing, provoking and challenging them with the bare bone mathematics can hopefully make them follow along or even take part in the expedition. These issues are in some ways independent of the computer, when it is the appreciation and exploration of algorithm that matters. Another thought experiment can be envisaged in which a live coding DJ writes down an instruction list for their set (performed with iTunes, but real decks would do equally well). They proceed to HDJ according to this instruction set, but halfway through they modify the list. The list is on an overhead projector so the audience can follow the decision making and try to get better access to the composer's thought process.*



Contudo, alguns músicos exploram suas idéias como processos de *software*, muitas vezes ao ponto que o *software* se torna a essência da música. Neste ponto, os músicos podem ser pensados como programadores explorando seu código manifestado como som. Isso não reduz seu papel principal como um músico, mas complementa, com a perspectiva única na composição de sua música. **Termos como “música generativa” e “música de processos” tem sido inventados e apropriados para descrever esta nova perspectiva de composição.** Muita coisa é feita das supostas propriedades da chamada “música generativa” que separa o compositor do resultado do seu trabalho. Brian Eno compara o fazer da música generativa com o semear de sementes que são deixadas para crescer, e sugere abrir mão do controle dos nossos processos, deixando eles “brincarem ao vento”.<sup>21</sup>

Isso sumariza a prática musical do *live coding* como 1) Música de Processos (subseção 3.2.1), ou Música de algoritmos simples, 2) Música Generativa (subseção 3.2.2), e 3) práticas de Disk Jockey (seção 3.3).

### 3.1.6 *Show us your screens*

Além das performances inaugurais nos festivais Europeus, e do manifesto “Live Algorithm Programming and Temporary Organization for its Promotion”, um texto possui uma importância fundamental para *live coding*. Premissas de comportamentos e técnicas são delineadas no texto “*Show Us Your Screens*”(GRIFFITHS, 2008; McCallum; SMITH, 2011, p. 22; online):

Exigimos:

- Acesso à mente do intérprete, para todo o instrumento humano.
- Obscurantismo é perigoso. Mostre-nos suas telas.
- Programas são instrumentos que podem modificar eles mesmos.
- O programa será transcendido - Língua Artificial é o caminho.
- O código deve ser visto assim como ouvido, códigos subjacentes visualizados bem como seu resultado visual.

• Codificação ao vivo não é sobre ferramentas. Algoritmos são pensamentos. Motosserras são ferramentas. É por isso que às vezes algoritmos são mais difíceis de perceber do que motosserras.

Reconhecemos contínuos de interação e profundidade, mas preferimos:

- Introspecção dos algoritmos.
- A externalização hábil de algoritmo como exibição expressiva/impressiva de destreza mental.

<sup>21</sup> WARD et al., op. cit., p. 245-246. Tradução nossa de *Indeed, some musicians explore their ideas as software processes, often to the point that a software becomes the essence of the music. At this point, the musicians may also be thought of as programmers exploring their code manifested as sound. This does not reduce their primary role as a musician, but complements it, with unique perspective on the composition of their music. Terms such as “generative music” and “processor music” have been invented and appropriated to describe this new perspective on composition. Much is made of the alleged properties of so called “generative music” that separate the composer from the resulting work. Brian Eno likens making generative music to sowing seeds that are left to grow, and suggests we give up control to our processes, leaving them to “play in the wind”.*

- Sem *backup* (minidisc, DVD, safety net computer).

Nós reconhecemos que:

- Não é necessário para uma audiência leiga compreender o código para apreciar, tal como não é necessário saber como tocar guitarra para apreciar uma performance de guitarra.
- Codificação ao vivo pode ser acompanhada por uma impressionante exibição de destreza manual e a glorificação da interface de digitação.
- Performance envolve contínuos de interação, cobrindo talvez o âmbito dos controles, no que diz respeito ao parâmetro espaço da obra de arte, ou conteúdo gestual, particularmente direcionado para o detalhe expressivo. Enquanto desvios na tradicional taxa de reflexos táteis da expressividade, na música instrumental, não são aproximadas no código, por que repetir o passado? Sem dúvida, a escrita de código e expressão do pensamento irá desenvolver suas próprias nuances e costumes. <sup>22</sup>

“Dar acesso à mente do intérprete” e “obscurantismo é perigoso” descrevem um meio de evitar qualquer código mal intencionado; isto é uma hipótese: *i*) programas de *live coding* geralmente são programas em fase de desenvolvimento; *ii*) programas em desenvolvimento possuem, inevitavelmente, *bugs*<sup>23</sup> *iii*) *bugs* podem ser explorados e levar à corrupção do sistema. Se esta hipótese estiver correta, justificaria a atitude de exposição da *imagem-texto*. No entanto não encontrei algum estudo crítico descrevendo se isso é verdade a partir do ponto de vista do público, isto é, será que o público pode estar realmente interessado na exposição da *imagem-texto*? Será que essa exposição não pode ser perigosa para o processo artístico e para a experiência do público? Embora sejam questões que fogem do escopo do trabalho, são importantes, necessitando verificar algumas performances para averiguar.

<sup>22</sup> Tradução nossa de: *We demand:* • *Give us access to the performer's mind, to the whole human instrument.* • *Obscurantism is dangerous. Show us your screens.* • *Programs are instruments that can change themselves.* • *The program is to be transcended - Artificial language is the way.* • *Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome.* • *Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws.* . *We recognise continuums of interaction and profundity, but prefer:* • *Insight into algorithms* • *The skillful extemporisation of algorithm as an expressive/impressive display of mental dexterity* • *No backup (minidisc, DVD, safety net computer) . We acknowledge that:* • *It is not necessary for a lay audience to understand the code to appreciate it, much as it is not necessary to know how to play guitar in order to appreciate watching a guitar performance.* • *Live coding may be accompanied by an impressive display of manual dexterity and the glorification of the typing interface.* • *Performance involves continuums of interaction, covering perhaps the scope of controls with respect to the parameter space of the artwork, or gestural content, particularly directness of expressive detail. Whilst the traditional haptic rate timing deviations of expressivity in instrumental music are not approximated in code, why repeat the past? No doubt the writing of code and expression of thought will develop its own nuances and customs.*

<sup>23</sup> Segundo James S. Huggins, historicamente “O termo *bug* é usado de forma limitada para designar qualquer falha ou problema em conexões ou no trabalho com aparatos elétricos” Tradução nossa de *The term "bug" is used to a limited extent to designate any fault or trouble in the connections or working of electric apparatus.* (ver <[http://www.jamesshuggins.com/h/tek1/first\\_computer\\_bug.htm](http://www.jamesshuggins.com/h/tek1/first_computer_bug.htm)>). Nesse sentido, um *bug* em um programa é uma falha de operação, geralmente causada por algum erro de lógica, por parte do programador.



“Programas são instrumentos” e “O programa será transcendido - Língua Artificial é o caminho”, são frases que fazem menção direta à experiência de usuário (*live coder*), isto é, um sistema que é programável de maneira facilitada. A seguinte hipótese pode ser feita: quanto mais simplificada a linguagem de programação, mais expressão visual ou musical um espetáculo poderá ter (o que pode não ser verdade, e sim que a expressão musical estaria no nível sensível). Por “Língua Artificial” entendo que o *live coder* pode criar *mini-linguagens* ou Linguagens de Domínio Específico (DSL)<sup>24</sup> que possibilitam criar programas para criar um espetáculo audiovisual ou musical. Segundo Collins e McLean (2014), tais DSLs estariam no formato de “mini-linguagens” bem desenvolvidas para a tarefa específica de codificar música ao vivo, operando técnicas composicionais como a transformação de um padrão musical (como por exemplo, técnicas barrocas como inversão e retrogradação ou técnicas aleatórias, como embaralhamento de um conjunto de eventos sonoros), facilitando a espontaneidade no processo criativo:

Existe um número crescente de sistemas de *livecoding* com “mini-linguagens” amigáveis, que facilitam o *loop* e construções de camadas centrais típicas para dançar música. *Ixilang* é um exemplo primário, e possui um editor de código estruturado que, enquanto baseado em texto, suporta correspondências visuais. *Tidals* é outro, e, embora com foco na rapidez de utilização ao invés da facilidade de aprendizagem, está começando a ter mais ampla aceitação. Ambos *ixilang* e *Tidal* promovem padrões em termos de funções transformadoras como embaralhamento, inversão e extrapolação de formas diferentes. (COLLINS; McLean, 2014, p. 357)<sup>25</sup>

“O código deve ser visto assim como ouvido” entraria em um problema próprio de programas de pesquisa em notação musical, sendo que o processo de correlação entre o que está escrito e o que está sendo ouvido leva um tempo ou pode mesmo nem existir. Mesmo com o convite expressado pelo manifesto “*Live Algorithm Programming and Temporary Organization for its Promotion*” no início do capítulo, a questão não está nem no uso do computador nem em alguma abordagem musical, e conforme a performance avança, a imagem-texto vai se tornando tão poluída que poderia causar um desinteresse.

<sup>24</sup> Sobre esse tema recomendo o texto “Minilanguages, finding a notation that sings” de Raymond (2003): “Historicamente, linguagens de dominio especifico sao do tipo que sao chamadas de ‘pequenas linguagens’ ou ‘minilinguagens’ no mundo do Unix, porque os primeiros exemplos eram pequenos e de pouca complexidade, em relação às linguagens de propósito geral (...) Nós manteremos o termo tradicional ‘minilinguagem’ para engatizar que no decorrer do curso é geralmente utilizado para projetar e mantê-las o menor e simples possível” (RAYMOND, 2003, 3º parágrafo). Tradução nossa de *Historically, domain-specific languages of this kind have been called ‘little languages’ or ‘minilanguages’ in the Unix world, because early examples were small and low in complexity relative to general-purpose languages (...) We’ll keep the traditional term ‘minilanguage’ to emphasize that the wise course is usually to keep these designs as small and simple as possible.*

<sup>25</sup> Tradução nossa de: *There are increasingly user friendly “mini-language” livecoding systems which facilitate loop and layer-centric constructions typical to dance music. ixilang is a primary example, and features a structured code editor which while text-based, supports visual correspondences. Tidal is another, and although its focus is on speed of use rather than ease of learning, is beginning to see wider take-up. Both ixilang and Tidal promote pattern in terms of transformative functions as scrambling, reversal and extrapolation in different ways.*

Codificação e teorização podem ser atos sociais. Se existe um público, revelar, provocar e desafiar eles com uma matemática complexa se faz com a esperança de que sigam, ou até mesmo participem da expedição. Estas questões são, de certa forma, independentes do computador, quando a valorização e exploração de algoritmo é que importa. (WARD et al., 2004, p. 204)

“Algoritmos são pensamentos. Motosserras são ferramentas.”, “Introspecção dos algoritmos.” e “A externalização hábil de algoritmo” descrevem uma atividade constante de formalizações lógicas, do processo febril de explorar uma complexidade própria do que se criou, de ficar digitando sem parar um teclado de computador. Alguns colegas e amigos não programadores, músicos e não músicos, utilizam a expressão “gostar de apertar botão” para se referir à caricatura do programador em um espaço reservado, no qual controla dispositivos diversos.

“Sem *backup*” indica o comportamento do *live coder* após uma improvisação, que não memoriza em discos rígidos, cd’s ou *pendrives* o documento criado (isto é, o código textual, em alguma extensão apropriada para a linguagem utilizada, por exemplo, *.pl*, Perl, *.scheme*, Scheme, *.js*, JavaScript), da mesma forma que um músico de improvisação dificilmente transcreveria o que tocou em uma partitura, no máximo gravando o áudio da performance.

A respeito de “Não é necessário para uma audiência leiga compreender” e “A codificação ao vivo pode ser acompanhada por uma impressionante exibição de destreza” pode indicar uma proximidade com aquele modelo de prática musical virtuosística (como um espetáculo de habilidades técnicas); mais especificamente, este modelo poderia partir daquilo que Magnusson (2014) chama de “adoção de um método pré-romântico de compor através da performance em tempo real, onde tudo fica aberto a mudar – o processo composicional o design do instrumento e a inteligência do sistema tocando a peça.” (MAGNUSSON, 2014, p. 4)<sup>26</sup>

## 3.2 *Live coding* e Música de algoritmos

### 3.2.1 Música de algoritmos simples

Segundo Wooler et al., a música de processos é uma “Música resultante de um conjunto de processos colocados em movimento pelo compositor, tais como ‘In C’ de Terry Riley e ‘It’s gonna rain’ de Steve Reich” (WOOLER et al., 2005 apud ENO, 1996, p. 1)<sup>27</sup>; o que é chamado de “procedural” por Wooler et al., é chamado por Joshua Mailman

<sup>26</sup> Tradução nossa de: *live coding adopts a pre-Romantic method of composing through performance in real time, where everything remains open to change – the compositional process, the instrument design, and the intelligence of the system performing the piece.*

<sup>27</sup> Tradução nossa de *Music resulting from processes set in motion by the composer “In C” by Terry Riley and “It’s gonna rain” by Steve Reich*

(2013) em seu artigo “*Agency, Determinism, Focal Time Frames and Processive Minimalist Music*”, música de processos mínimos, ou música minimalista de processos determinísticos. O autor faz menção aqui ao compositor Alvin Lucier (1931-) e sua peça *Crossings* (1984). Segundo o autor,

A longa forma de *Crossings* (1984) de Alvin Lucier é especialmente clara; ela surge processivamente, neste caso de um glissando de som senoidal puro que lentamente (mais de 16 minutos) ascende de infra-sons para a ultra-sons. Um processo discreto combina com este glissando, criando uma série de processos contínuos de curto alcance. Uma orquestra dividida alterna no jogo de alturas consecutivas, em uma escala cromática ascendente ascendente que interage com o glissando. Um grupo inicia e sustenta um intervalo acima daquela do glissando de tal modo que a uma dissonância mútua cria batimentos; conforme o glissando aumenta, a velocidade do batimento diminui; quando se alcança um uníssono com os instrumentos, os batimentos momentaneamente cessam; como o glissando ascende acima da nota dos instrumentos, batimentos gradualmente aceleram. Eventualmente, o outro conjunto entra e sustenta um intervalo de um semitom acima, começando esse processo de curto alcance novamente. (MAILMAN, 2013, p. 128)<sup>28</sup>

Essa descrição é um breve comentário da peça, mas descreve um simples processo de ação musical, bem como relata de forma clara os resultados sonoros. É possível presumir que existe, na execução de *Crossings*, um simples algoritmo que permeia toda a peça: uma alternância de um ritmo lento de dois grupos instrumentais defasados temporalmente em relação ao som emitido pelo alto-falante<sup>29</sup>.

Uma vez que na MP também é possível notar a influência de um algoritmo, é necessário evitar confusão com o uso original da MP, tal qual em compositores como Reich, Lucier e Riley; já o seu uso apropriado é realizado por Adrian Ward, Julian Rohrer, Frederick Olofsson, Alex McLean, Dave Griffiths, Nick Collins e Amy Alexander. Sugiro portanto comparar o fragmento do manifesto citado na página 53 com um fragmento do manifesto *Music as a Gradual Process* do compositor Steve Reich (1968):

O que é distintivo sobre processos musicais é que eles determinam todos detalhes de nota-a-nota (som-a-som) e toda a forma simultaneamente (pense em um *Round* ou um *Canon*). **Eu estou interessado em processos perceptivos. Eu quero ser capaz de ouvir o processo**

<sup>28</sup> Tradução nossa de *The long-range form of Alvin Lucier's Crossings (1984) is especially clear; it arises processively, in this case from a glissando of a pure sine tone that slowly (over sixteen minutes) ascends from infrasonic to ultrasonic. A discrete process combines with this glissando, creating a series of short-range continuous processes. A divided orchestra alternates in playing the consecutive pitches of a rising chromatic scale that interacts with the glissando. One ensemble initiates and sustains a pitch just higher than that of the glissando such that their mutual dissonance creates beats; as the glissando rises, the speed of the beats slows; when it reaches a unison with the instruments, the beats momentarily cease; as the glissando ascends above the pitch of the instruments, the beats gradually accelerate. Eventually, the other ensemble enters and sustains a pitch a semitone higher, starting this short-range process again.*

<sup>29</sup> Esse tipo de algoritmo pode ser facilmente implementado em uma linguagem de programação musical, como o SuperCollider (ver ??).

**acontecendo através da música que soa.** Para facilitar a descrição detalhada de um processo musical, a escuta deve acontecer muito gradualmente. (REICH, 1968, p. 1).<sup>30</sup>

Quanto ao manifesto de Adrian Ward, Julian Rohruber, Frederick Olofsson, Alex McLean, Dave Griffiths, Nick Collins e Amy Alexande, tenho uma primeira impressão que essa transformação gradual do som não está presente da mesma forma que em Reich, Lucier ou Riley; vejo que o uso a reminiscência do termo “processo” a partir de “procedural” e “processador” se dá mais pela atividade de reprogramação em uma CGS ou CGC, mais especificamente, para fins de entretenimento. Aqui o *processo* é formalizado como algoritmo, colocado na situação de constante modificação.

Já no manifesto de Reich o som não é programado, mas se prevê um comportamento musical em constante modificação a partir de uma ação mínima sobre aquilo que produz o som. Um exemplo interessante é seu *Pendulum Music* (1968), onde deixa oscilar microfones posicionados acima de alto-falantes, com os cabos presos no teto; através de um único tipo de ação, restrita apenas pelas distâncias tomadas por intérpretes antes de colocar os microfones em movimento e pela comprimento do cabo preso ao teto<sup>31</sup>, ocorrerá um *processo físico* (desaceleração das oscilações que influenciam quando o microfone passará pelo alto-falante) que desencadeará um *processo perceptivo* (escuta de diferentes momentos de quando estes microfones passam pelo alto-falante até um momento estacionário).

Tal técnica faz uso deliberado de um fenômeno conhecido como microfonia, um *processo recursivo* onde os sons projetados pelo alto-falante são retro-alimentados pelo microfone. A retroalimentação já era usada por Pauline Oliveros (1969), como por exemplo em *Beautiful Soup* (1967), peça para fita de dois canais, elaborada através de um circuito em retroalimentação (a compositora indica uma atividade musical econômica que ao longo do tempo causa "interessantes mudanças timbrísticas em sons sustentados"<sup>32</sup>). Uma proposta posterior também foi realizada por Alvin Lucier (1931-) em *I'm sitting in a room* (1969) quando retroalimenta seu próprio discurso em uma sala, incrementando sinais ao gravador, criando um *drone*, ou um som de duração indefinida<sup>33</sup>. Esta técnica de retroalimentação também pode ser utilizada no *live coding*, por exemplo, no SuperCollider (ver ??).

<sup>30</sup> Tradução nossa de: *The distinctive thing about musical processes is that they determine all the note-to-note (sound-to-sound) details and the over all form simultaneously. (Think of a round or infinite canon.) I am interested in perceptible processes. I want to be able to hear the process happening throughout the sounding music. To facilitate closely detailed listening a musical process should happen extremely gradually.*

<sup>31</sup> Seria possível incluir aqui também a resistência do ar ao movimento do objeto.

<sup>32</sup> Cf. OLIVEROS, 1969, p. 43 Tradução nossa de: *interesting timbre changes on sustained sounds*

<sup>33</sup> É interessante comentar, nesse contexto, o uso deliberado de *drones* no *livecoding* através de uma apropriação da *Drone Music*.

### 3.2.2 Música de algoritmos complexos

Esta seção discute o discurso da MG a partir de dois pontos: da perspectiva de Brian Eno, próximo à MP, e da perspectiva estrutural, próxima das teorias de gramáticas generativas. A primeira será abordada na ?? e a segunda na ??.

de Brian Eno foi mencionada por Ward et al. (2004), farei um comentário a respeito, embora a inclusão pareça ser polisêmica a partir de um fragmento de texto de Eno (1978):

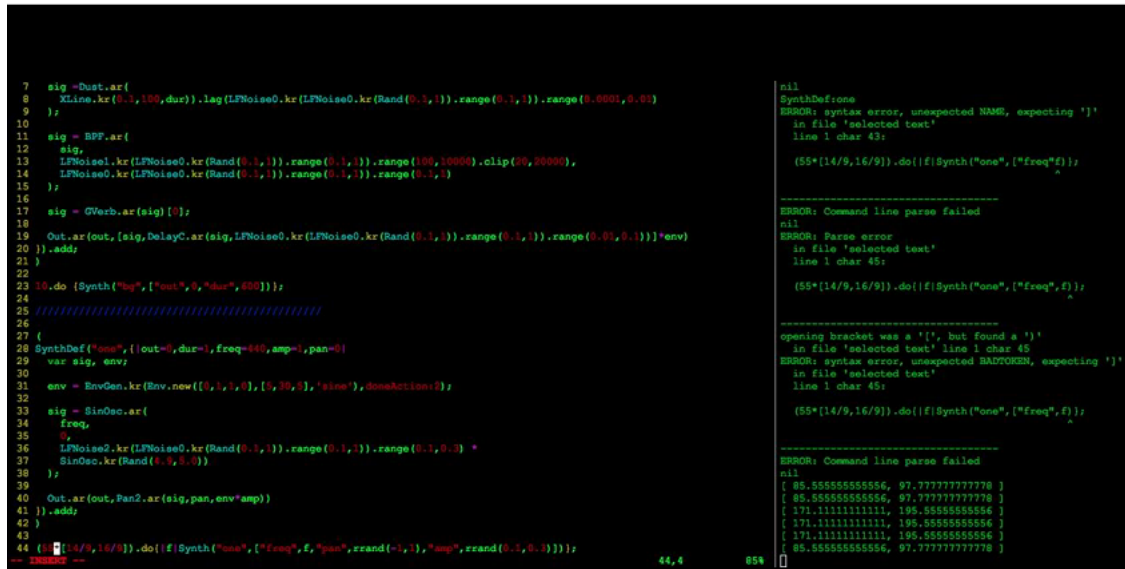
O conceito de música projetada especificamente como pano-de-fundo ambiental foi um pioneirismo da indústria Muzak nos anos cinquenta, e tem sido conhecido genericamente como Muzak. As conotações que este termo carrega são particularmente associadas com um tipo de material que a indústria Muzak produz - melodias familiares arranjadas e orquestradas de maneira leve e derivativa. Compreensivelmente, isso leva ouvintes atentos (e compositores) a dispensar inteiramente o conceito de música ambiental como digna de atenção (...). Se as empresas existentes de música enlatada se baseiam em regularizar ambientes cobrindo as idiosincrasias acústicas e atmosféricas, a Música AMbiental intenciona em melhorá-las. **Se o pano-de-fundo convencional é produzido removendo todo o sentido de dúvida e incerteza (e portanto o interesse genuíno) da música, a Música Ambiental retém essas qualidades** (ENO, 1978, online) <sup>34</sup>

É interessante que neste contexto podemos fazer referência à *Discreet Music* (1975) e *Ambient 1: Music for Airports* (1978); tais peças fazem uso deliberado de laços de fitas que criam sistemas de atrasos (*delays*). Se tais procedimento criativos não eram novos na época <sup>35</sup>; geralmente essas abordagens podem criar *drones* que ficam variando em seu espectro.

Um exemplo interessante de *live coding* com *drones*, que utiliza pequenos impulsos sonoros no SuperCollider (ver Figura 7). Aqui o processo de escuta é contínuo, semelhante à práticas da música eletroacústica, onde algoritmos são pré-concebidos e modificados durante a performance. Cole Ingraham utiliza um objeto que gera impulsos sonoros aperiódicos, e o mantém como um plano sonoro que se mantém, e vai sendo. Paralelamente, sons contínuos vai permeando este plano dos impulsos, a partir de senóides com um vibrato. É interessante notar que em alguns momentos, o *live coder* deixa de programar para prestar

<sup>34</sup> The concept of music designed specifically as a background feature in the environment was pioneered by Muzak Inc. in the fifties, and has since come to be known generically by the term Muzak. The connotations that this term carries are those particularly associated with the kind of material that Muzak Inc. produces - familiar tunes arranged and orchestrated in a lightweight and derivative manner. Understandably, this has led most discerning listeners (and most composers) to dismiss entirely the concept of environmental music as an idea worthy of attention. (...) Whereas the extant canned music companies proceed from the basis of regularizing environments by blanketing their acoustic and atmospheric idiosyncracies, Ambient Music is intended to enhance these. Whereas conventional background music is produced by stripping away all sense of doubt and uncertainty (and thus all genuine interest) from the music, Ambient Music retains these qualities. Grifo nosso.

<sup>35</sup> Nesse sentido indicamos as peças de Steve Reich e Alvin Lucier e Pauline Oliveros já citadas.



```

7  sig = Dust.ar(
8    XLine.kr(0.1, 100, dur).lag(LFNoise0.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.0001, 0.01)
9  );
10
11  sig = BPF.ar(
12    sig,
13    LFNoise1.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(100, 10000).clip(10, 20000),
14    LFNoise0.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.1, 1)
15  );
16
17  sig ~ QVerb.ar(sig) [0];
18
19  Out.ar(out, [sig, DelayC.ar(sig, LFNoise0.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.01, 0.1))] * env)
20  ].add;
21
22
23  10.do { Synth("bg", ["out", 0, "dur", 600]);
24
25  ///////////////////////////////////////////////////
26
27  {
28    SynthDef("one", { |out=0, dur=1, freq=640, amp=1, pan=0|
29      var sig, env;
30
31      env ~ EnvGen.kr(Env.new([0, 1, 0], [0, 30, 0], 'sine'), 0.000001);
32
33      sig = SinOsc.ar(
34        freq,
35        LFNoise2.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.1, 0.3) *
36        SinOsc.kr(Rand(0.1, 0.3))
37      );
38    };
39
40    Out.ar(out, Pan2.ar(sig, pan, env * amp))
41  ].add;
42
43
44  { [14/9, 16/9].do { |f| Synth("one", ["freq", f, "pan", 0, "dur", 0.1, "amp", 0.1]); } }
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figura 7 – Improviso com *drones* no SuperCollider **Fonte:** <<https://www.youtube.com/watch?v=b8j4umQ2IIE>>.

atenção no resultado sonoro obtido, ao invés de ficar constantemente re-programando o código-fonte.

### 3.3 Prática DJ

Somado a todos esses elementos, existe a imagem já descrita de um DJ que se utiliza destas técnicas e estéticas para controlar seu *set* (embora seja possível assumir que esta personagem pode utilizar outros dispositivos, como sintetizadores e baterias eletrônicas, todos eles presentes em um computador) e promover uma música na qual a dança é elemento central. É importante deixar claro que uma cultura DJ possui suas peculiaridades dependentes de contexto social: por exemplo, um dj europeu/norte-americano está em uma configuração social diversa do dj latino-americano e africano. Se até o momento temos discutido práticas musicais, na sua maioria, a partir de pessoas que falam o inglês como sua primeira língua, é natural supor que esta cultura DJ que falamos no *livecoding* faz parte de uma cultura anglófona.

Esse fenômeno de apropriação das MA, MG, MP e DJ, na cultura musical dos países rotulados como desenvolvidos, pode ser entendido a partir daquilo que Fernando Iazzetta (2009) chamou de "sinergia de produções, que em sua diversidade compartilham dos mesmos elementos sociais e culturais" (IAZZETTA, 2009, p. 152). Mais especificamente, trata-se de um fenômeno em sentido mais amplo das produções musicais usando o computador; mas buscarei encarar a questão no *livecoding* da mesma maneira, como uma sinergia entre boates, casas noturnas e ambientes informais com ambientes de produção de conhecimento



(universidades, escolas de música, faculdades de engenharia).

Primeiro é necessário esclarecer se a apropriação foi intencional ou não. O discurso do manifesto de [Ward et al. \(2004\)](#) indica uma consciência parcial dos autores a respeito desta sinergia. No entanto é discutível se esse cruzamento de gêneros é mais um reflexo das transformações sociais e culturais que o computador trouxe consigo do que algo intencional: o uso deliberado de muitas ferramentas de um estúdio portátil, a expansão de possibilidades de produção musical através das redes de computadores, a troca de informações a respeito do uso de novos *softwares* entre músicos acadêmicos e não-acadêmicos, trouxe à tona diferentes comunidades daquela ou outra tecnologia. De forma semelhante, no *livecoding*, as novas possibilidades tecnológicas em contato com modos de fazer música já estabelecidos possibilitaram a emancipação de novas práticas, o que por sua vez, capacitou o cruzamento de estéticas (tudo isso, no entanto, visto apenas pela ótica das culturas de pessoas que falam o inglês como primeira língua).

Poderia ser questionado se o intercruzamento de gêneros musicais no *livecoding* depende dos *softwares*. [Iazzetta](#) responderia não, mas que dependem de uma articulação entre produções musicais e seus compositores, que carregam diferentes formações teóricas. Esse problema é colocada da seguinte forma:

Embora seja possível considerar uma "comunidade Max/MSP", ela está dentro de uma população contendo as comunidades "SuperCollider", "PureData", "ChucK", indicando apenas alguns.

Dentro dessa população, surgem as pequenas comunidades de *softwares* de *live coding*, pela utilização e invenção de mini-linguagens pouco usadas, se comparadas com as do parágrafo acima. No contexto anglófono, essas pequenas comunidades apropriam um termo, segundo [Collins e McLean \(2014\)](#), *algorave*.

Algorave é um tipo de música eletrônica de pista, que se utiliza de algoritmos, processos, e teorias generativas; não se configura como uma música de pista normal, em seu processo de criação, mas reproduz alguns regimes de escuta, como *dance*, *drum'n'bass*, *cyberpunk*, etc. Nesse processo de criação, é comum utilizar alguns *softwares* já comentados, como o *iXiLang* e o *Tidal*, o que leva a crer na emancipação de “comunidades iXiLang” e “comunidades Tidal”.

No entanto, *algorave* é um termo anterior ao advento do *live coding* e ao uso dos *softwares* supracitados:

*Algorave* não é sustentado exclusivamente por *live coders*, mas estes têm mantido uma forte presença em todos os eventos até agora. É assim talvez, porque a tradição do *live coding* de projetar telas motiva todo o esforço; onde algoritmos não estão visíveis por períodos de tempo durante uma *algorave*, se corre o risco das coisas parecerem muito como um evento de

música eletrônica padrão. (COLLINS; McLean, 2014, p. 356) <sup>36</sup>

Collins e McLean apresenta dados a respeito da história da *algorave*: em 1992 uma performance intitulada *Cybernetic Composer* de Charles Ames; passando pelo *Aphex Twin* (Richard David James), que reivindica em 1997 o termo (interessante do ponto de vista de gênero musical) *live club algorithm*; em 2000 o *Slub*, citado no início deste capítulo (na época Adrian Ward, Alex McLean), realizam performances, autodenominadas *generative techno*, com abordagem *gabba*; é interessante aqui o uso do termo *club live coding*. Em 2001 é identificado a utilização de redes neurais para composição de padrões semelhantes ao *drum'n'bass*. Em 2004 é fundado o TOPLAP, organização internacional de *live coding*, em uma casa noturna de Hamburgo. <sup>37</sup>

---

<sup>36</sup> Tradução nossa de *Algorave is not exclusively a preserve of live coders, but they have maintained a strong presence at every event thus far. This is perhaps because the live coding tradition of projecting screens help motivates the whole endeavour; where algorithms are not made visible for periods during an algorave, we run the risk of things feeling much like a standard electronic music event.*

<sup>37</sup> COLLINS; McLean, 2014, loc. cit..



## 4 Estudos de casos

Neste capítulo, apresento *Study in Keith* de Andrew (??), considerando o que foi exposto na [Equação 4](#), no [Capítulo 2](#) na pág. X.

Mais especificamente, este capítulo objetiva estudar como ocorre a união entre as regras de validação, localização transversal e de gosto específicos do *livecoding* e de *Keith Jarret* (ver [Equação 2](#)) no espaço conceitual de *Study in Keith*.

$$\mathcal{C}_{Study\ in\ Keith} = \mathcal{C}_{livecoding} \cup \mathcal{C}_{Sun\ Bear\ Concerts} \quad (4.1)$$

A variável  $\mathcal{C}_{Study\ in\ Keith}$  será discutida na [seção 4.1](#). A variável  $\mathcal{C}_{livecoding}$  foi discutida no capítulo anterior. A variável  $\mathcal{C}_{Sun\ Bear\ Concerts}$  será discutida na [subseção 4.1.1](#).

De forma análoga à [Equação 2](#), o espaço conceitual de  $\mathcal{C}_{Sun\ Bear\ Concerts}$  pode ser definido pela função de interpretação de regras:

$$\mathcal{C}_{Sun\ Bear\ Concerts} = \langle\langle\langle \mathcal{R}_{Sun\ Bear\ Concerts}, \mathcal{T}_{Sun\ Bear\ Concerts}, \mathcal{E}_{Sun\ Bear\ Concerts} \rangle\rangle\rangle \quad (4.2)$$

Neste trabalho, assumo que, para todos os conceitos, uma mesma linguagem  $L$  está sendo usada por Sorensen para expressar regras. De um lado técnico da Ciências da Computação, esta linguagem é o LISP. Do lado da “linguagem musical”, estamos falando de um estilo pessoal de tocar *Jazz* de *Keith Jarret*, ou em outras palavras, uma união entre uma linguagem de programação específica, e um espaço conceitual de Andrew Sorensen sobre *Keith Jarret*

$$\mathcal{L}_{Study\ in\ Keith} = \mathcal{L}_{LISP} \cup \mathcal{C}_{Andrew\ Sorensen \Rightarrow Keith\ Jarret} \quad (4.3)$$

### 4.1 Study in Keith

Segundo Andrew Sorensen, “*A Study In Keith* é um trabalho para piano solo (NI’s Akoustik Piano), inspirado nos concertos *Sun Bear* de Keith Jarrett’s” ([SORENSEN, 2012](#)).

O NI, é uma abreviação para *Native Instruments*, uma empresa de manufatura para de tecnologias para áudio <sup>1</sup>. O *Akoustic Piano* é uma extensão (*plugin*) VST<sup>2</sup>, que emula diferentes pianos acústicos, e suas reverberações, em 10 diferentes tipos de dinâmicas e tempos de sustentação. Entre os instrumentos utilizados, incluem o *Bösendorfer 290 Imperial*, *Steinway D*, e *Bechstein D 280*<sup>3</sup>

Os concertos *Sun Bear* são originalmente dez LPs de improvisações de Keith Jarret, gravados pela *ECM Records*<sup>4</sup> em 1976 no Japão, e lançados em 1978. Kyoto, 5 de novembro<sup>5</sup>. Osaka, 8 de novembro. Nagoya, 12 de novembro<sup>6</sup>. Tokyo, 14 de novembro. Sapporo, 18 de Novembro.

Uma nota pertinente sobre esta improvisação feita pelo próprio Sorensen: nos primeiros dois minutos do vídeo, existe um silêncio. Este silêncio é decorrente da construção das estruturas lógicas do programa. (Id., 2012). Este comportamento, do tempo de codificação, ao tempo de ação musical, é similar em outro vídeo analisado na ??.

#### 4.1.1 Concertos *Sun Bear*

TODO ...

#### 4.1.2 Algorithms are Thoughts, Chainsaws are Tools

“Algorithms are Thoughts, Chainsaws are Tools” é o nome dado ao vídeo de Stephen Ramsay (2010), contendo uma análise da performance de *Study in Keith*. Ramsay apresenta o vídeo como:

Um curta sobre *livecoding* apresentado como parte do Grupo de Estudos de Crítica de Códigos, em 2010, por Stephen Ramsay. Apresenta uma leitura ao vivo [*live reading*] de uma performance do compositor Andrew Sorensen. Também fala sobre J.D. Salinger, the Rockets, tocando instrumentos, Lisp, do clima em Brisbane e kettle drums. <sup>7</sup>

É interessante aqui notar que este vídeo contém comentários. Abaixo realizei uma compilação de fragmentos de alguns dos comentários que considere pertinentes.

<sup>1</sup> Tradução parcial de *Native Instruments is a leading manufacturer of software and hardware for computer-based audio production and DJing. The company's mission is to develop innovative, fully-integrated solutions for all musical styles and professions. The resulting products regularly push technological boundaries and open up new creative horizons for professionals and amateurs alike.* Disponível em <<http://www.native-instruments.com/en/company/>>.

<sup>2</sup> *Virtual Studio Technology*. Disponível em <<http://www.steinberg.net/>>.

<sup>3</sup> Disponível em <<http://www.kvraudio.com/product/akoustik-piano-by-native-instruments>>.

<sup>4</sup> <http://www.ecmrecords.com/>

<sup>5</sup> Disponível em <<https://www.youtube.com/watch?v=T2TfIQNxhjc>>.

<sup>6</sup> <<https://www.youtube.com/watch?v=3a7ezm3D1jA>>.

<sup>7</sup> Tradução de *A short film on livecoding presented as part of the Critical Code Studies Working Group, March 2010, by Stephen Ramsay. Presents a "live reading" of a performance by composer Andrew Sorensen. It also talks about J. D. Salinger, the Rockettes, playing musical instruments, Lisp, the weather in Brisbane, and kettle drums..*

O primeiro comentário destaca a função de uma partitura-programação (FENRICH; OBICI; SCHIAVONI, 2014, p. 5), escrita em linguagem LISP. Amanda French nega a utilização do termo *partitura* para explicitar diferenças no uso da programação-partitura, em uma performance de improvisação com o computador, para uma performance não-improvisada com partitura:

[Amanda French]: A noção de partitura não se aplica aqui, é como não fosse possível aplicá-lo ao músico de *jazz* ou tocador de *bluegrass*. (...). Levanta a questão, para mim, se, em uma sessão de *livecoding* \*feita\* constituído no ato de digitar em um programa existente, seria tão convincente – Eu acho que isso pode definitivamente ter pontos de interesse. Ou qual seria o análogo do *livecoding* para uma performance não-improvisada de música?<sup>8</sup>

Um segundo comentário, coloca a pergunta de Amanda em outra perspectiva:

(...) ao ver como o *livecoding* sugere o que associamos de forma típica à música improvisada, podemos imaginar uma forma na qual uma performance musical sugere o *livecoding*? O que torna o *livecoding* diferente, e pode a performance de música tradicional imitar isso? Para responder esta questão, parece importante notar que as formas nas quais a música improvisada muitas vezes apela para alguma noção de autenticidade ou gênio. Enquanto o *livecoding* ele mesmo à noção de virtuosismo de código, “autenticidade” parece fora de lugar aqui. Se música improvisada sugere expressão, o *livecoding* sugere um conjunto de restrições na expressão, descrevendo os parâmetros através dos quais a máquina (midi) ganha expressão<sup>9</sup>

<sup>8</sup> Tradução parcial de *The notion of "sheet music" doesn't apply here, as it wouldn't apply to a jazz musician or a bluegrass picker. Even the name of his environment, Impromptu, makes that point. Raises the question for me precisely of whether a livecoding session that \*did\* consist of simply typing in an existing program would be as compelling – I think it would definitely have its points of interest, actually. Or what would the livecoding analog be to a non-improvisational live performance of music?*

<sup>9</sup> Tradução nossa de (...) *having seen how livecoding suggests what we typically associate with improvised music, can we imagine a way in which a music performance suggests livecoding? What makes livecoding different, and can a traditional music performance mimic it? To answer this question, it seems important to note the ways in which improvised music often appeals to some notion of authenticity or genius. While livecoding might lend itself to some notion of coding virtuosity, "authenticity" seems out of place here. If improvised music is expression, livecoding suggests a setting of constraints on expression, describing the parameters through which the machine (midi) gets expressed.*



## 5 Conclusão



# Referências

- AYCOCK, J. A brief history of just-in-time. p. 97–113, 2003. Disponível em: <http://www.cs.tufts.edu/comp/150IPL/papers/aycock03jit.pdf>. Citado 4 vezes nas páginas 13, 30, 34 e 35.
- BECKER, H. *Art Worlds*. [S.l.: s.n.], 1982. Citado na página 11.
- BLACKWELL, A.; COLLINS, N. The programming language as a musical instrument. p. 120–130, 2005. Disponível em: [http://www.researchgate.net/publication/250419052\\_The\\_Programming\\_Language\\_as\\_a\\_Musical\\_Instrument](http://www.researchgate.net/publication/250419052_The_Programming_Language_as_a_Musical_Instrument). Citado 2 vezes nas páginas 34 e 36.
- BODEN, M. *The Creative Mind: myths and mechanisms*. 2. ed. Routledge, Taylor & Francis Group, 1990. ISBN 0-203-34008-6. Disponível em: <http://www.pauladaunt.com/books>. Citado na página 15.
- BROWN, C.; BISCHOF, J. *INDIGENOUS TO THE NET: Early Network Music Bands in the San Francisco Bay Area*. 2013. Disponível em: <http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html>. Citado na página 33.
- Collins, N. All this and brains too: Thirty years of howling round. p. 7, 2002. Disponível em: <http://www.nicolascollins.com/texts/allthisandbrains.pdf>. Citado na página 34.
- COLLINS, N.; McLean, A. Algorave: Live performance of algorithmic electronic dance music. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. [S.l.: s.n.], 2014. p. 355–358. Citado 3 vezes nas páginas 39, 45 e 46.
- COPE, E. D. Prefacio a OM composer's book vol. 2. In: *OM Composer's Book*. [S.l.]: Editions Delatour, 2008. v. 2, p. ix–xv. Citado na página 19.
- ENO, B. Base de dados, *Music for Airports liner notes*. 1978. Disponível em: [http://music.hyperreal.org/artists/brian\\_eno/MFA-txt.html](http://music.hyperreal.org/artists/brian_eno/MFA-txt.html). Citado 2 vezes nas páginas 15 e 43.
- ENO, B. *Generative Music: "Evolving metaphors, in my opinion, is what artists do. A talk delivered in San Francisco"*. 1996. Disponível em: <http://www.inmotionmagazine.com/eno1.html>. Citado na página 40.
- FENERICH, A.; OBICI, G.; SCHIAVONI, F. Marulho TransOceânico: performance musical entre dois continentes. p. 12, 2014. Disponível em: <https://www.academia.edu/t/M8Fvh/8178331>. Citado na página 49.
- GRIFFITHS, D. Fluxus: Scheme livecoding. 2008. Disponível em: <http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>. Citado na página 37.
- HEIKKILÄ, V.-M. Discovering novel computer music techniques by exploring the space of short computer programs. p. 8, 2011. Disponível em: <http://arxiv.org/abs/1112.1368>. Citado na página 19.

IAZZETTA, F. *Música e mediação tecnológica*. [S.l.]: Ed. Perspectiva-Fapesp, 2009. ISBN 9.788527308724E12. Citado 2 vezes nas páginas 44 e 45.

Lakatos, I. Falsification and the methodology of scientific research. In: *The Methodology of Scientific Researchs Programmes*. [s.n.], 1970. p. 8–93. Disponível em: <http://strangebeautiful.com/other-texts/lakatos-meth-sci-research-phil-papers-1.pdf>. Citado 2 vezes nas páginas 9 e 27.

MAGNUSSON, T. Herding cats: Observing live coding in the wild. v. 38, n. 1, p. 8–16, 2014. Citado na página 40.

MAILMAN, J. B. Agency, determinism, focal time frames, and processive minimalist music. In: *Music and Narrative Since 1900*. [s.n.], 2013. p. 125–144. Disponível em: [https://www.academia.edu/749803/Agency\\_Determinism\\_Focal\\_Time\\_Frames\\_and\\_Narrative\\_in\\_Processive\\_Minimalist\\_Music](https://www.academia.edu/749803/Agency_Determinism_Focal_Time_Frames_and_Narrative_in_Processive_Minimalist_Music). Citado na página 41.

MATHEWS, M. V. The digital computer as a musical instrument. v. 142, n. 3592, p. 553–557, 1963. Disponível em: <http://www.jstor.org/stable/1712380>. Citado na página 9.

MATHEWS, M. V. et al. *The technology of computer music*. 2a, 1974. ed. [S.l.]: MIT press, 1969. ISBN 0 26213050 5. Citado na página 9.

MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 4 vezes nas páginas 9, 30, 31 e 32.

McCallum, L.; SMITH, D. *Show Us Your Screens*. Vimeo, 2011. Disponível em: <https://vimeo.com/20241649>. Citado na página 37.

McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: [https://www.academia.edu/467101/Music\\_improvisation\\_and\\_creative\\_systems](https://www.academia.edu/467101/Music_improvisation_and_creative_systems). Citado 6 vezes nas páginas 5, 9, 15, 17, 18 e 19.

MCLEAN, A. et al. (Ed.). *Proceedings of the First International Conference on Live Coding*. [S.l.]: ICSRiM, University of Leeds, 2015. 300 p. ISBN 9780853163404. Citado na página 13.

McLean, A.; WIGGINS, G. *Patterns of movement in live languages*. 2009. Disponível em: [https://www.academia.edu/7249277/Patterns\\_of\\_movement\\_in\\_live\\_languages](https://www.academia.edu/7249277/Patterns_of_movement_in_live_languages). Citado 2 vezes nas páginas 30 e 34.

MORI, G. Analysing live coding with ethnographical approach. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 117–124. ISBN 978 0 85316 340 4. Citado na página 11.

MORI, G. Pietro grossi's live coding. an early case of computer music performance. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 125–132. ISBN 978 0 85316 340 4. Citado 4 vezes nas páginas 13, 30, 32 e 33.

Neto, J. B. Imre lakatos e a metodologia dos programas de investigação científica. (apresentação do pensamento do filósofo da ciência imre lakatos). p. 12, 2008. Disponível em: <http://people.ufpr.br/~borges/diversos/publicacoes.html>. Citado 2 vezes nas páginas 9 e 27.



NUNZIO, A. D. *Genesi, sviluppo e diffusione del software "MUSIC N" nella storia della composizione informatica*. phdthesis — Facoltà di Lettere e Filosofia - Università degli Studi di Bologna, 2010. Citado na página 30.

OLIVEROS, P. Tape delay techniques for eletronic composers. In: *Software for people: collected writings 1963-80*. 1. ed. Smith Publications, 1969. p. 11. Disponível em: <[http://monoskop.org/images/2/29/Oliveros\\_Pauline\\_Software\\_for\\_People\\_Collected\\_Writings\\_1963-80.pdf](http://monoskop.org/images/2/29/Oliveros_Pauline_Software_for_People_Collected_Writings_1963-80.pdf)>. Citado 3 vezes nas páginas 16, 30 e 42.

Pressing, J. Improvisation: Methods and models. In: *Generative processes in music*. (ed. J. Sloboda) Oxford University Press, 1987. p. 50. Disponível em: <<http://musicweb.ucsd.edu/~sdubnov/Mu206/improv-methods.pdf>>. Citado 3 vezes nas páginas 15, 17 e 19.

PROSPERO, C. D. Social participation. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 68–73. ISBN 978 0 85316 340 4. Citado na página 11.

RAMSAY, S. *Algorithms are Thoughts, Chainsaws are Tools*. Vimeo, 2010. Disponível em: <<https://vimeo.com/9790850>>. Citado na página 48.

RAYMOND, E. S. Minilanguages, finding a notation that sings. In: *The Art of Unix Programming*. Eric Steven Raymond, 2003, (<http://www.faqs.org/faqs/>). Disponível em: <<http://www.faqs.org/docs/artu/minilanguageschapter.html>>. Citado na página 39.

REICH, S. Music as a gradual process. In: *Writings about Music, 1965–2000*. Oxford University Press, 1968. ISBN 978-0-19-511171-2. Disponível em: <<http://ccnmtl.columbia.edu/draft/ben/feld/mod1/readings/reich.html>>. Citado 3 vezes nas páginas 15, 41 e 42.

ROADS, C. *Microsound*. [S.l.]: MIT press, 2001. ISBN 978-0-262-18215-7. Citado na página 4.

ROBERTS, C.; KUCHERA-MORIN, J. *Gibber: live-coding audio in the browser*. [S.l.]: University of California at Santa Barbara: Media Arts & Technology Program, 2012. Citado 2 vezes nas páginas 13 e 35.

ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado na página 35.

SANTOS, B. d. S. Para além do pensamento abissal: Das linhas globais a uma ecologia de saberes. n. 78, p. 3–46, 2007. Disponível em: <[http://www.ces.uc.pt/myces/UserFiles/livros/147\\_Para%20alem%20do%20pensamento%20abissal\\_RCCS78.pdf](http://www.ces.uc.pt/myces/UserFiles/livros/147_Para%20alem%20do%20pensamento%20abissal_RCCS78.pdf)>. Citado 2 vezes nas páginas 5 e 27.

SANTOS, B. d. S. A filosofia à venda, a douta ignorância e a aposta de pascal. n. 80, p. 11–43, 2008. Disponível em: <[http://www.ces.uc.pt/myces/UserFiles/livros/47\\_Douta%20Ignorancia.pdf](http://www.ces.uc.pt/myces/UserFiles/livros/47_Douta%20Ignorancia.pdf)>. Citado 2 vezes nas páginas 5 e 27.

SORENSEN, A. *A Study in Keith*. Youtube, 2012. Disponível em: <[https://www.youtube.com/watch?v=BqYBT\\_HoG4M](https://www.youtube.com/watch?v=BqYBT_HoG4M)>. Citado 3 vezes nas páginas 5, 47 e 48.

SuperCollider.ORG. *SuperCollider Overviews: JITLib - An overview of the Just In Time library*. 2014. Citado na página 9.

Sá, S. P. d. Se você gosta de madonna também vai gostar de britney! ou não? gêneros, gostos e disputa simbólica nos sistemas de recomendação musical. v. 12, n. 2, p. 1808–2599, 2009. Citado na página 12.

TURNER, V. W.; HARRIS, A. *"Liminality and Communitas". The Ritual Process: Structure and Anti-Structure*. [S.l.: s.n.], 1969. Citado na página 11.

WARD, A. et al. *Live algorithm programming and temporary organization for its promotion*. TOPLAP.ORG, 2004. Disponível em: <<http://art.runme.org/1107861145-2780-0/livecoding.pdf>>. Citado 7 vezes nas páginas 13, 34, 36, 37, 40, 43 e 45.

WIGGINS, G. A. A preliminary framework for description, analysis and comparison of creative systems. v. 19, n. 3592, p. 449–458, 2006. Disponível em: <<http://axon.cs.byu.edu/Dan/673/papers/wiggins.pdf>>. Citado 6 vezes nas páginas 5, 9, 15, 16, 17 e 18.

WOOLER, R. et al. A framework for comparasion of process in algorithmic music systems. *Generative Arts Practice*, p. 109–124, 2005. Disponível em: <<http://eprints.qut.edu.au/6544/1/6544.pdf>>. Citado na página 40.

WYSE, L.; SUBRAMANIAN, S. The viability of the web browser as a computer music platform. v. 37, n. 4, p. 10–23, 2014. Citado na página 13.