Guilherme Martins Lunhani

Live coding e o computador como instrumento: assimetria de gêneros musicais na rede social *Soundcloud*.

19 de outubro de 2015

Guilherme Martins Lunhani

Live coding e o computador como instrumento: assimetria de gêneros musicais na rede social *Soundcloud*.

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

19 de outubro de 2015

,

Criei todas as festas, todos os triunfos, todos os dramas. Experimentei inventar novas flores, novos astros, novas carnes, novas línguas. Acreditei adquirir poderes sobrenaturais. Ora bem! eis que devo enterrar minha imaginação e minhas lembranças! Que bela glória de artista e narrador arrebatada!

Arthur Rimbaud

Resumo

Apresento o seguinte tema: quais categorizações músicais estão incluídas em , improvisações feitas com computadores, mais especificamente em uma prática conhecida por *live coding*.

Descrevo como uma prática multidisciplinar e assimétrica em sua variedade. De fato, observo o *live coding* como um Programa de Investigação Científica emergente em Nortes políticos/econômicos. Este programa, multidisciplinar, enquadra Ciências da Computação e Artes (Música, Audiovisual, Dança). O método de pesquisa, mais observado no corpus de textos (e neste trabalho), até o momento, se apresentou demasiadamente cirúrgico.

Nesse sentido, realizo a atividade (preliminar) de abordar o *live coding* como uma *ecologia de saberes*, buscando na medida do possível, diminuir a assimetria epistemológica notificada. Direciono, na Parte I, uma discussão sobre assimetrias de definições do *live coding*, de significados, de precurssores, e de práticas recentes. Direciono, na Parte II, uma discussão sobre as assimetrias de gênero musical, em um ambiente restrito, uma mídia social conhecida como *Soundcloud*.

Palavras-chaves: *Livecoding*. Improvisação. Gêneros Musicais.

Sumário

Introdução

A adoção do computador como instrumento musical remonta às investigações de Max Mathews (1963), Mathews et al. (1969), Mathews e Moore (1970). A dificuldade desta investigação, pelo menos na época, toca a questão de um equilíbrio entre a performance humana, e a resolução dos timbres digitais. De fato, era uma tarefa computacionalmente complicada naquela época. Muitas negociações de capital simbólico (artigos, entrevistas, *softwares*) foram realizadas em mercados do conhecimento de música e de computadores. Estes mercados, círculos universitários ou corporativos, principalmente estadounidenses[1], e franceses[2], possibilitaram a materialização de programas como CSound, Max/MSP, PureData e SuperCollider, para citar os principais.

Colocar alguma referência concreta sobre alemães e italianos?

Este último *software*, *SuperCollider*, foi o ponta-pé para a escolha de um objeto de estudo. Uma parte da comunidade de músicos que contextualiza este ambiente de programação musical[3], se apropriou de um paradigma das ciências da computação – compilação *Just In Time*–, para valorizar outro, o objeto de pesquisa deste trabalho:

> *Programação imediata* (ou: programação de conversa, *livecoding* [<http://www.toplap.org>], programação no fluxo, programação interativa) é um paradigma que inclue 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)[4]

A definição é apropriada para um verbete de um documento técnico. Porém reduz o *livecoding* (ou também *live coding*) a um paradigma técnico, dos códigos programados, ou de um modo técnico de performatizar uma improvisação. Nesse trabalho, investiguei-o como um universo de conceitos. Esse universo de conceitos foi construído por um grupo social urbano, produtor de artefatos técnico-artísticos, sendo que exemplos musicais são numerosos. Embora com desenvolvimento recente, aproximadamente 15 anos, alguns predecessores são lembrados.

Um dos mais precoces, descrito por Giovanni Mori (2015b), é o compositor italiano Pietro Grossi. No final dos anos 70, trata o computador como um caricato do piano.

1 *Bell Labs*, *IBM*, *Princeton*, *MIT*.
2 IRCAM
3 Síntese sonora e composição algorítmica. Disponível em <https://supercollider.github.io>.
4 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.*

Trabalhando em um laboratório particular, desenvolve programas como TAU, TAU2 e TELETAU. Os dois primeiros abandonam a preocupação timbrística e salientam a questão performática, com apenas uma forma de onda quadrada. O último pode ser descrito como um dos mais antigos programas funcionais para performances musicais remotas.

Segundo Alex McLean e Wiggins (2009), no final dos anos 70 e dos anos 80, emergem compositores como John Bischoff, Tim Perkis, Chris Brown, Scot Gresham-Lancaster, Ron Kuivilla e Nicolas Collins. Antes tutoriados por compositores como Alvin Lucier, Darius Milhaud, John Chowning, Robert Ashley e Terry Riley, divulgam a *Live Computer Music* utilizando, principalmente experimentos com micro-controladores e sistemas de *feedback*.

No entanto, para McLean e Wiggins, o *live coding* não possui sua identidade cultural até a emergência da organização TOPLAP. Nesse sentido, o manifesto "*Live Algorithm Programming and Temporary Organization for its Promotion*" tem um papel fundamental. Este documento identifica não apenas um tipo de performance, mas uma personagem (o *live coder*), um núcleo/heurística para um novo Programa de Investigação Científica (PIC)[5], e uma rede de manifestações artísticas (delineadas neste trabalho pelos gêneros musicais). A partir disso realizo a seguinte pergunta: **como é a rede de gêneros músicais quando se fala em *live coding*? ela é simétrica ou assimétrica?**

O Capítulo 1 discute o *live coding* como *universo de conceitos*. No capítulo **??**, arituculo este último para almejar um *ecologia de saberes* sobre o *live coding*. Esse método contextualiza, no Capítulo 2, uma discussão sobre uma assimetria de conhecimentos no *live coding*. O **??**, é um prelúdio que contextualiza os resultados no Capítulo 3. Os resultados tem por fim confirmar a assimetria de conhecimentos, de maneira quantitativa, em uma rede social (*Soundcloud.com*).

[5] Cf. Lakatos; Neto, 1970, 2008.

Parte I

Ecologia de saberes no *livecoding*

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 etnongráfico, ao apontar locais e costumes do *live coding*. Também discute a emancipação de um gênero musical com o mesmo nove de um tipo de evento, *algorave*. Este gênero/evento ocorre principalmente em congressos universitários e em casas noturnas. Ali recorrências à Música são constantes, mas não únicas (é possível incluir aqui também vídeo e dança).

Ao transitar na universidade, é possível pensar o *live coding* como um *Programa de Investigação Científica*, realizado principalmente em países de primeiro mundo, Inglaterra, Estados Unidos, Alemanha e Austrália. Pesquisas latino-americanas, como por exemplo México e Brasil, são significativas, porém tendem a seguir as epistemes do Norte político e econômico. Por exemplo, em duas recentes pesquisas. O primeiro caso é o *SuperCopair* de Junior, Lee e Essl (2015), e o segundo é o *Vivace* de Vieira et al. (2015).

O *SuperCopair* é um *software* que permite realizar uma performance musical remotamente com o *SuperCollider*[1]. Em testes realizados entre EUA e Brasil, sínteses sonoras e algoritmos musicais foram codificados e o áudio transmitido (*streaming*), com latência de 166 a 230 milissegundos (Junior; Lee; Essl, 2015, p. 153).

O *Vivace* é um *software* que possibilita realizar uma performance musical, presencialmente e/ou remotamente (ilustro na Figura 1 uma performance presencial). Utiliza navegadores de *internet* para performances audiovisuais, resultantes da re-definição dinâmica de conjuntos numéricos, por um humano. É curioso notar a menção de um termo derivado de *live coding*, como *freakcoding*.

Ao meu ver, conhecimentos latino-americanos do *live coding* são parcialmente reflexos dos conhecimentos produzidos em nível internacional. Essa reflexão seria (isso é uma hipótese), de uma assimetria no *universo de conceitos* musicais do *live coding*. Esta assimetria pode ser verificada percepto-auditivamente, mas também o foi quantitativamente. Descrevo o *algorave* como uma miscigenação de gêneros musicais Jr. (2003), Sá (2006), ou entre aquilo que Sá (2009) chama de *músicas-populares massivas* e um corpo de estudiosos universitários de *ComputerMusic*.

Neste capítulo, lido com dois desafios específicos, para melhor introduzir o que é *live coding*: *i)* o desafio de discutir a miscigenação de de conceitos em um universo cognitivo (seção 1.2); *ii)* o desafio de discutir a tradução dos conhecimentos de um domínio para português (seção 1.2);

[1] Disponível em <https://github.com/supercollider/supercollider>.

Figura 1 – Exemplo de performance de *livecoding*, realizado durante o .AUDIOVISUALAOVIVO (AVAV) em 18/12/2012 em Higienopolis – SP (<https://www.youtube.com/watch?v=i9Lj3S1bSH4>). Caleb Mascarenhas, Gera Rocha, Renato Fabbri e Vilson Vieira. O software utilizado é o *Vivace* (VIEIRA et al., 2015). **Fonte**: <http://i.ytimg.com/vi/i9Lj3S1bSH4/hqdefault.jpg>.

1.1 O universo de conceitos durante uma improvisação

Uma sessão de *live coding* é uma improvisação. McLean (2006), um dos principais pesquisadores ingleses, no campo musical, articula a improvisação musical como algo computável, organizado por representações lógicas. O *conceito* se torna uma instância, uma variável. É representado pela letra c. O *universo de conceitos* se torna um conjunto, representado pela letra U, como um agrupamento de instâncias de c. Por outro lado, c é definido por conjuntos de "objetos" (O'), características (F'), e processos (P').

O primeiro contem a premissa para uma improvisação. O segundo é um conjunto de várias premissas, no sentido de uma rede de conceitos que definem o conceito principal. Os três últimos são classes de informações que definem como o conceito deve ser tocado para caracterizar uma linguagem musical específica. Nesse sentido, sugiro que diferentes performances de *live coding* passaram por um cruzamento de configurações destas classes lógicas

Por outro lado, este pensamento pode ser demasiadamente disciplinar, acarretando em uma assimetria epistemológica. Para evitar este caminho, utilizei um dispositivo de contraste. Recorri à *Ecologia de saberes* do sociólogo Boaventura de Souza Santos (2007), Santos (2008, p. 20;p. 17) para apontar quais são algumas destas assimetrias.

A ecologia de saberes será discutida como possível método de pesquisa, na seção 1.4. Possibilitou contextualizar, de maneira parcialmente do ponto de vista social no Capítulo 2. Possibilitou verificar uma assimetria de conhecimentos, por "ano", "país" e "cidade", e "gênero musical" (*hashtags*) na mídia social *Soundcloud*[2] no Capítulo 3.

1.2 A questão da tradução

Ao traduzir literalmente *live*, como "ao vivo", e o sufixo *coding* como "codificar", contextualizo uma performance cujo ato de codificação é o ato produtivo para a realização do fazer musical.

Porém a tradução falha em identificar questões-satélites, como por exemplo, a própria Música. Para identificar algumas questões-palavras, fiz uma "compilação" dos anais do ICLC 2015 (MCLEAN et al., 2015) .A Figura 2 é uma *nuvem de palavras* [3], uma representação visual de dados textuais. Longe de ser uma análise linguística, utilizei esta ferramenta para me auto-induzir a encontrar as questões-satélites do *live coding*.

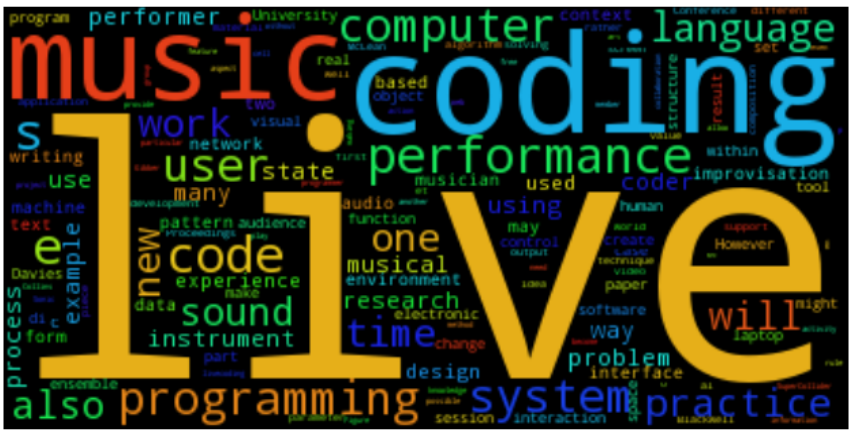

Figura 2 – Nuvem de palavras dos anais ICLC2015 **Fonte**: autor.

Uma breve análise da nuvem de palavras pode elucidar parte das questões-satélites. Na Tabela 1 filtrei parte dos resultados na nuvem de palavras por conjuntos de funções textuais – sujeitos-humanos, sujeitos-ferramentas, verbos, adjetivos e substantivos – e quantas vezes foram utilizados, em categorias qualitativas (0, menos usado e 9 o mais usado, sendo que 6 e 7 não apresentaram resultados). [4].

[2] Disponível em <https://www.soundcloud.com/>.
[3] Disponível em <https://github.com/amueller/word_cloud>.
[4] O método de extração será explicado em um apêndice oportuno.

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**
Pessoas	-	Collins, Blackwell, McLean, Grossi	-	-	-	-	-	-
Aplicativos	-	SuperCollider, Gibber, SonicPi	-	-	-	-	-	
Verbos	take, see, shared, networked, explore, made	make, provide, writing, solving, making	used	using, coding	performer	-	-	-
Adjetivo ou numeral, ordinal	less, open, potential, similar, important, cognitive, virtual	first, real, electronic, visual, ensemble, possible, free, livecoding, aspect	musical, many	new, one	-	-	live	-
Substantivo	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

No caso dos sujeitos-humanos, podemos ver nomes de Nick Collins e Alex McLean, praticantes responsáveis pela criação de um manifesto, cujo um fragmento será discutido no capítulo 2. Pietro Grossi, é um personagem recentemente estudado por Mori (2015b) como um caso prematuro de *live coding*, a partir do final da década de sessenta.

No caso dos sujeitos-ferramentas, destacamos o papel do *SuperCollider*, já citado anteiormente, e do *Gibber*[5]. Ambos são ambiente de programação para de síntese sonora e composição algorítmica. Uma semelhança paradigmática para estes ambientes, é o procedimento de compilação de códigos conhecido como *Just In Time* (AYCOCK, 2003). Enquanto no primeiro *software* a questão está posta em uma máquina – *laptop* – local, o *Gibber* representa a viabilidade do navegador de *internet* como plataforma musical Roberts e Kuchera-Morin (2012), Wyse e Subramanian (2014).

Os verbos fornecem informação sobre o comportamento dos improvisadores de códigos. Além da atividades como *performatizar* e *codificar*, é notável atividades sociais ligadas à visão, à escrita, à técnica, à lógica. Embora a Música seja a atividade proeminente do *live coding*, não obtivemos resultados que retornassem, por exemplo, a palavra *hearing*. Isso é significativo, e no Capítulo 2, 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 2, 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*.

Substantivos relacionam a atividade como processo (*process*). Como será discutido no Capítulo 2, o *live coding* se apropria de conceitos da Música como Processo de Steve Reich e da Música Generativa de Brian Eno para justificar um processo de codificação incessante. Por outro lado, palavras como *university*, *research* e *technology*, e *laptop* acusam não apenas uma prática artística, mas um programa de pesquisa tipo de performance, que utiliza um computador para resultados audiovisuais, realizados por universitários. A esfera de pesquisa acadêmica permitiu ramos de desenvolvimento com linguagens de programação, cognição, inteligência artificial, semiologia, performance musical (improvisação), e mais recentemente, etnologia, conferindo à produção de *live coding* uma aura de legitimidade escolar.

[5] Disponível em <http://gibber.mat.ucsb.edu>

1.3 Pergunta para o Método

Na Figura 1 apresentei um caso brasileiro. Por um número de documentos reduzidos em língua portuguesa, irei chamar de *live coding* o que é produzido utilizando moajoritariamente nesta língua.

McLean et al. (2010) corrobora com definição de Blackwell e Collins (2005). Enquanto primeiro afirma que o *live coding* é uma "improvisação de vídeo e/ou música usando **linguagens de computador** que tem se desenvolvido em um campo ativo de pequisa e prática artística ao longo da última década"[6], o segundo descreve como um processo social:

> Nos anos recentes houve uma expansão da atividade do *live coding*, e a formação de um corpo internacional para suportar *live coders* - TOPLAP. O sítio <toplap.org> e sua lista de email é a mais ativa casa para práticas artístcas, e o TOPLAP tem sido listado para tais festivais de música eletrônica como Ultrasound (2004), transmediale 2005 (Berlin) e Sonar 2005 (Barcelona) (BLACKWELL; COLLINS, 2005, p. 3-4).[7]

Atualmente um outro sítio, *Live Code Research Network*[8], é outra referência para o programa de pesquisa. Conforme alternativas organizacionais, técnicas e estéticas do *live coding* proliferam, a tarefa de definir o que é ou não é *live coding* se torna bastante nebulosa. Por exemplo: a utilização de linguagens visuais (*live patching*) são um caso a parte ou devem ser incluídas no objeto de pesquisa? É pertinente chamar de *live coding* uma performance que utiliza o *Live*!*Ableton*? Utilizar dispositivos diferentes do *laptpop*, como corpos dançantes, é *live coding*?

Sugiro um percursso reformulando a pergunta feita na introdução deste trabalho: **considerando o *live coding* como uma prática interdisciplinar, o que pode ser pressuposto musicalmente entre os praticantes e o público?**

1.4 Método de pesquisa

Ao surgir como programa de pesquisa em meados dos anos 2000, *live coders* tendem a praticar uma forma de construção do conhecimento emprestando assuntos de outras áreas. Notei as seguintes referências aos assuntos: (1) Música; (2) Cinema; (3) Instalações Artísticas; (4) Educação; (5) Ciências da Computação; (6) Semiologia (7) Lógica; (8) Etnografia.

[6] Tradução parcial nossa de: *Live coding , the improvisation of video and/or music using computer language, has developed into an active field of research and arts pratice over the last decade.* Grifo nosso

[7] Tradução parcial de: *Recent years have seen further expansion of live coding activity, and the formation of an international body to support live coders- TOPLAP (Ward et al. 2004). The toplap.org site and mailing list is the most active current home for this artistic practice, and TOPLAP have been booked for such electronic music festivals as Ultrasound 2004 (Huddersfield), transmediale 2005 (Berlin) and sonar 2005 (Barcelona).*

[8] Disponível em <http://www.livecodenetwork.org/>.

Representar esta multiplicidade de assuntos, e todas suas assimetrias epistemológicas seria uma tarefa cirúrgica. Empresto o termo do sociólogo português Boaventura de Souza Santos (2007), Santos (2008), sugerindo o *live coding* como uma feira de idéias, onde se compra e vende conceitos conforme a necessidade, nesse caso, a própria tese de mestrado.

Reconheço aqui um ponto de vista que observa o objeto de pesquisa como algo assimétrico. Para evitar um corte demasiadamente cirúrgico, Santos sugere um método consciente de pluralidades e desproporções daquilo que se quer saber. O pesquisador será forçado na academia, então, a criar uma abordagem limitada do que se quer falar:

> O saber só existe como pluralidade de saberes, tal como a ignorância só existe como pluralidade de ignorâncias. As possibilidades e os limites de compreensão e de acção de cada saber só podem ser conhecidas na medida em que cada saber se propuser uma comparação com outros saberes. Essa comparação é sempre uma versão contraída da diversidade epistemológica do mundo, já que esta é infinita. É, pois, uma comparação limitada, mas é também o modo de pressionar ao extremo os limites e, de algum modo, de os ultrapassar ou deslocar. Nessa comparação consiste o que designo por ecologia de saberes. (. . .) Sendo sempre limitado o conjunto de saberes que integra a ecologia dos saberes há que definir como se consituem esses conjuntos. **À partida, é possível um número ilimitado de ecologia de saberes, tão ilimitado quanto o da diversidade epistemológica do mundo. Cada exercício de ecologia de saberes implica uma selecção de saberes e um campo de interacção onde o exercício tem lugar**. (SANTOS, 2008, p. 28-30).

Para diferenciar o termo *ecologia* de sua aplicação nas ciências ecológicas, adaptei para este trabalho como **estudo do conjunto limitado e desproprocional de categorizações musicais do *live coding***.

Tarefas

Tarefas sugeridas para realizar no Capítulo 2: (1) Investigar de onde surgem o núcleo e a heurística do *live coding* (2) Exposição de uma ideologia musical que mistura: (a) *Música Algorítmica* (que será abreviado como MA) (b) *Música Processual* (MP) (c) *Música Generativa* (MG) (d) *Música de Pista* ou práticas DJ (DJ). .

Tarefas sugeridas para realizar no Capítulo 3: (1) Contextualização do gênero musical (4) Exposição dos dados.

2 Trabalhos Relacionados

Na seção 2.1 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 seção 2.2.

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 seção 2.3

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 seção 2.4.

Para McLean e Wiggins, o *live coding* não possui sua identidade cultural até a emergência da organização TOPLAP. Na seção 2.5 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 seção 2.10 tratarei do caso específico.

2.1 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 coding*:

> O GROOVE provê oportunidades para uma retroalimentação imediata de observações dos efeitos das funções temporais para as entradas do

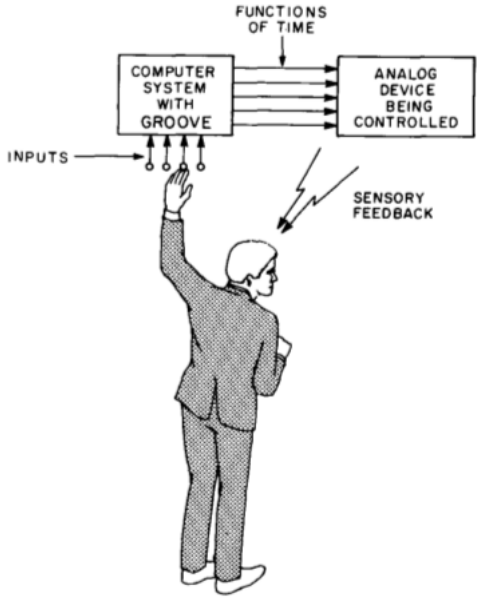

FIG. 1. Feedback loop for composing functions of time

Figura 3 – Esquema de concepção do projeto GROOVE descrito no artigo homônimo por Max Mathews **Fonte**: (MATHEWS; MOORE, 1970).

> 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 3]. Assim ele é capas de modificar as funções instantâneamente como um resultado de suas observações daqueles efeitos.(MATHEWS; MOORE, 1970, p. 715) [1]

O terceiro conceito observa a existência de uma relação entre um humano e uma máquina. De certa forma, o sistema passa pr 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 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 ua partitura e o intérprete deve influenciar a forma como a partitura é tocada. Seus modoes de influência pode ser mais variados do que aqueles que um regente

[1] 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 beign is in the feedback loop (...) Thus he is able to modify the functions instanteneously as a result of his observations of their effects.*

> convencional, que pode principalmente controloar o tempo, *loudness* e estilo.(MATHEWS; MOORE, 1970, p. 715-716) [2]

2.2 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 intéprete 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. Haviam muitas escolhas de reprodução no programa: era possível salvar na memória do computador peças de músicas pré-existentes, 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". Exsitia também uma abundância de escolhas para mudanças na estrutura da peça. Um dos mais importantes aspectos do trabalho de Grossi foi que todas intervenções eram instantâneas: o operado não tinha que esperar pelo computador terminar todas operações requisitadas, e depois ouvir os resultados. Cálculos de dados e reprodução sonoras eram simultâneos. Esta simultaneidade não era comum no campo da *Computer Music* daquele 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) [3]

[2] Tradução nossa de *The final concept is more nebulous. Since GROOVE is a man-computer system, the human engeneering 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 modiffied 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 a conventional conductor who primarily controls tempo, loudness, and style..*

[3] 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 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.*

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."[4]. 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"[5], 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 Stockhausen (1959).

2.3 Baía de São Franscisco

No final dos anos 70, na cena musical da Baía de São Fransisco, 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[6]. 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 nascimento?

> 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)[7]

[4] 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.*

[5] Disponível em <https://www.youtube.com/watch?v=ZQSP_wF7wSY>.

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

[7] 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 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*

ajustar informa-ções

Esta experiência foi levada a cabo por Bischof, Perkis, Chris Brown (1953-), Scot Gresham-Lancaster (1954-)[8], Mark Trayle (1955-)[9] e Phil Stone, que formaram nos anos 80 o grupo *The Hub*, com um primeiro disco lançado em 1989 entitulado *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)[10]. 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"[11], parte da *TELLUS #22*[12].

> A primeira performance conhecida de *live coding* foi em 1985, por Run 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)[13]

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

---

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.

8 Aluno de Darius Milhaud, John Chowning, Robert Ashley e Terry Riley.

9 Aluno de Robert Ashley.

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

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

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

13 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.*

2.4 Just In Time (JIT)

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

A sentença acima é uma "piada de um professor austríaco" (*idem, ibdem*), 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^2 (*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 destre trabalho que implementou a avaliação preguiçosa. A docuentação 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 geralmenet 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*][15]

Atualmente, esta técnica têm sido largamente implementada para navegadores de internet (ROBERTS; WAKEFIELD; WRIGHT, 2013). Programas como Gibber[16] (ROBERTS; KUCHERA-MORIN, 2012) e *wavepot*[17] 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 **??**.

2.5 LAPTOP ou TOPLAP?

Na seção 1.3, 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).

[14] Tradução nossa de *Passenger to taxtidriver: 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>.

[15] 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>

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

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

No *Wiki* do site oficial[18], a cada visita, cada letra é substituída por palavras randômicas, criando diferentes significados[19].

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 retroprojetor 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) [20]

inserir mais informações ou discussões

2.6 Live Algorithm Programming and Temporary Organization for its Promotion

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:

> Contudo, alguns músicos exploram suas idéias como processos de *software*, muitas vezes ao ponto que o *software* se torna a essência da música.

[18] Disponível em <http://toplap.org/wiki/Main_Page>.

[19] 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.

[20] 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.*

> 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". [21]

Isso sumariza a prática musical do *live coding* como *1)* Música de Processos (seção 2.7), ou Música de algoritmos simples, *2)* Música Generatva (seção 2.8), e *3)* práticas de Disk Jockey (seção 2.9).

2.7 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 Rilley e 'It's gonna rain' de Steve Reich" (WOOLER et al., 2005 apud ENO, 1996, p. 1)[22]; o que é chamado de "procedural" por Wooler et al., é chamado por Joshua Mailman (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

[21] 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".*

[22] Tradução nossa de *Music resulting from processes set in motion by the composer "In C" by Terry Riley and "Its gonna rain" by Steve Reich*

> 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)[23]

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[24].

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 Rohruber, 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 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).[25]

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 Rilley; 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.

[23] 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.*

[24] Esse tipo de algoritmo pode ser facilmente implementado em uma linguagem de programação musical, como o SuperCollider (ver **??**).

[25] 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.*

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[26], 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-alimentaados pelo microfone. A retroalimentação já era usada por Pauline Oliveros (1969), como por exemplo em *Beaultifoul 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"[27]). 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[28]. Esta técnica de retroalimentação também pode ser utilizada no *live coding*, por exemplo, no SuperCollider (ver **??**).

2.8 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 subseção 2.8.1.

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 as 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

[26] Seria possível incluir aqui também a resistência do ar ao movimento do objeto.

[27] Cf. OLIVEROS, 1969, p. 43 Tradução nossa de: *interesting timbre changes on sustained sounds*

[28] É interessante comentar, nesse contexto, o uso deliberado de *drones* no *livecoding* através de uma apropriação da *Drone Music*.

> idiossincrasias 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) [29]

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 [30]; 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 4). 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 mantêm como um plano sonoro que se mantêm, 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 atenção no resultado sonoro obtido, ao invés de ficar constantemente re-programando o código-fonte.

2.8.1 Abordagens estruturais

Músicas realizadas a partir do **??**, na **??**, página 61, podem ser relacionadas com pesquisas de gramática generativa, inaugurada com o artigo “Three Models for the description of language” de Noam Chomsky (1956). Nesta disciplina linguística/estrutural prevalecem as implementações computacionais analíticas que possibilitam usos criativos a partir de um *corpus teórico* já estabelecido[31].

---

[29] *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.

[30] Nesse sentido indicamos as peças de Steve Reich e Alvin Lucier e Pauline Oliveros já citadas.

[31] Sobre este tema sugerimos a leitura da tese "Luteria de Algoritmos Pós-Tonais"de Soares (2015-03-13), defendida recentemente nesta mesma instituição onde esta pesquisa está sendo realizada. Também elaborei uma peça eletroacústica nomeada *Impulsos* (<http://soundcloud.com/opusd/musica-gerativa-9-impulsos>), onde a partir do primeiro modelo de Chomsky (cadeia Markoviana de ordem 0) e de técnicas de retroalimentação de sinais descritas por Oliveros (1969) e Alvin Lucier, elaborei uma *árvore-drone* a partir de pulsos-finitos, feitos no aplicativo *Wavepot* (<http://www.wavepot.com>), semelhantes ao som de um metrônomo.

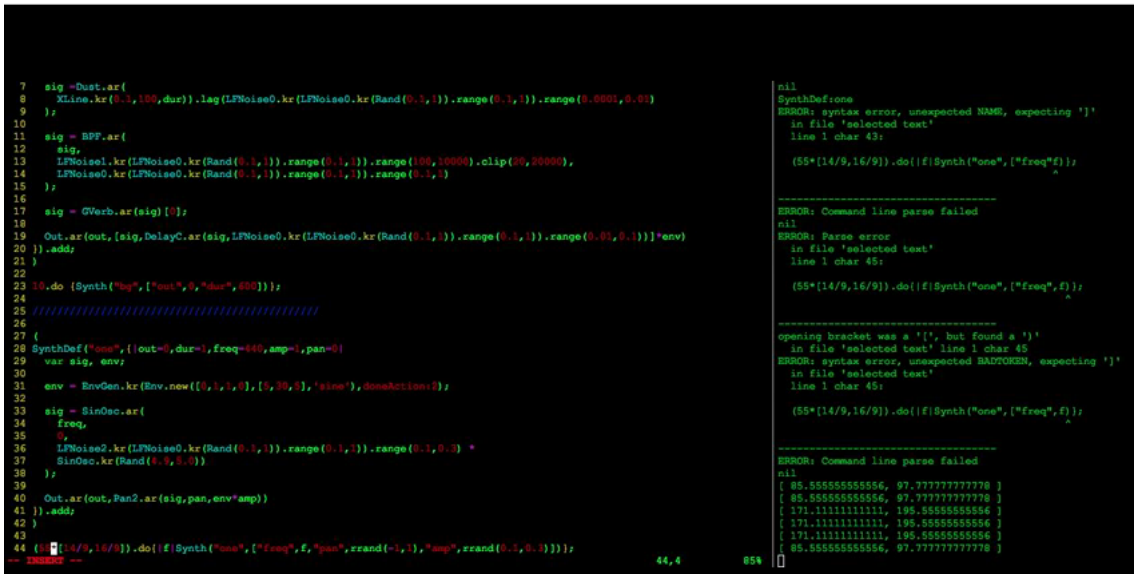

Figura 4 – Improviso com *drones* no SuperCollider **Fonte**: <https://www.youtube.com/watch?v=b8j4umQ2lIE>.

Aparentemente, tais implementações podem se encaixar naquilo que Curtis Roads (2001) define por *Design da macroforma feita de cima para baixo*, próximo dos esquemas diretores da MA, em contraste ao *Design da macroforma feita de baixo para cima*, esta última mais próxima da MP. Sobre o termo "de cima para baixo" me refiro a uma abstração de alto-nível que influencia na elaboração de detalhes de uma composição; por "de baixo para cima" me refiro a processos elementares de uma composição, que influencia na sua macro-forma.

No entanto temos que considerar que tal dicotomia pode colocar a criatividade em campos demasiadamente rígidos, no qual outras duas abordagens podem ser tomadas para entender o aspecto *linguístico*, uma negociadora entre a dicotomia e outra que elimina a dicotomia, a partir da noção de *restrições*. Esta última se encaixa mais em alguns dos modelos propostos por Chomsky. Roads problematiza esta questão da seguinte maneira:

> Uma abordagem estritamente de cima-para-baixo considera a macroestrutura como um plano global pré-concebido ou um modelo cujos detalhes são preenchidos em fases posteriores da composição. Isto corresponde à noção tradicional de forma na música clássica, onde certos esquemas formais tem sido usados por compositores como moldes (Apel 1972). Muitos compositores predeterminam a macroestrutura de suas peças de acordo um um esquema mais-ou-menos formal antes de um único som ser composto. Por contraste, uma abordagem estritamente de baixo-para-cima concebe a forma como resultado de um processo de desenvolvimento interno provocado por interações nos níveis mais baixos da estrutura (. . .) Para alguns, composição envolve uma mediação entre as abordagens de cima-para-baixo e de baixo-para-cima, entre uma concepção abstrata de alto-nível e os materiais concretos sendo desenvolvidos nos níveis mais baixos da estrutura. Isso implica uma negociação entre o desejo de ordenar a macroestrutura e imperativos que emergem da fonte material

> (...) Ultimamente, a dicotomia entre forma e processo é uma ilusão, uma falha da linguagem para cegar dois aspectos do mesmo conceito em uma unidade. Nas ciências da computação, o conceito de *restrição* elimina esta dicotomia (Sussman and Steele 1981). Uma forma é construída de acordo com um conjunto de relações. Um conjunto de relações implica um processo de avaliação que resulta em uma forma. (ROADS, 2001, p. 12-14) [32]

Restrições são pequenas regras, que se assemelham a gramáticas; tais gramáticas, podem depender métodos e técnicas de composição que restringem possibilidades criativas, afim de gerar novas restrições e possibilidades. Esta categoria de composições dependem de um corpo teórico bem estabelecido, procedendo primeiramente através da análise; tais análises podem partir como a Teoria de níveis estruturais de Heinrich Schenker, da Teoria Generativa da Música Tonal de Lerdhal e Jackendoff (GTTM) e a Teoria de Implicação-Realização de Eugene Narmour, para depois agir criativamente; tais análises indicariam comportamentos direcionais, melódicos, harmônicos e rítmicos de uma composição, colocando este aspecto como relativo a algumas produções da música tonal européia [33].

No *live coding*, essa abordagem pode ser realizada com o *Tidal* de McLean e Wiggins (2010)[34]., como na Figura 5.

2.9 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

[32] Tradução nossa de:*A strict top-down approach considers macrostructure as a preconceived global plan or template whose details are filled in by lates stages of composition. This corresponds to the traditional notion of form in classical music, wherein certain formal schemes have been used by composers as molds (Apel 1972). (...) Many composers predetermine the macrostructure of their pieces according to a more-or-less formal scheme before a single sound is composed. By contrast, a strict bottom-up approach conceives of form as the result of a process of internal development provoked by interactions on lower levels of musical structure (...) For some, composition involves a mediation between the top-down and bottom-up approaches, between an abstract high-level conception and the concrete materials beign developed on lower levels of musical structure. This implies negotiation between a desire for ordely macrostructure and imperatves that emerge from source material (...) Ultimately, the dichotomy between form and process is an illusion, a failure of language to bind two aspects of the same concept into a unit. In computer science, the concept of constraints does away with this dichotomy (Sussman and Steele 1981). A form is constructed according to a set of relationships. A set of relationships implies a process of evaluation that results in form*

[33] Cf. ROWE, 1991, p. 29–43

[34] "*Tidal* é uma lingagem de padrões embebida na linguagem de programação Haskell, consistindo de representações de padrões, uma biblioteca de geradores de padrões" (McLean; WIGGINS, 2010, p. 2). Tradução nossa de *Tidal is a pattern language embedded in the Haskell programming language, consisting of pattern representation,a library of pattern generators and combinators, an event scheduler and programmer's live coding interface.*

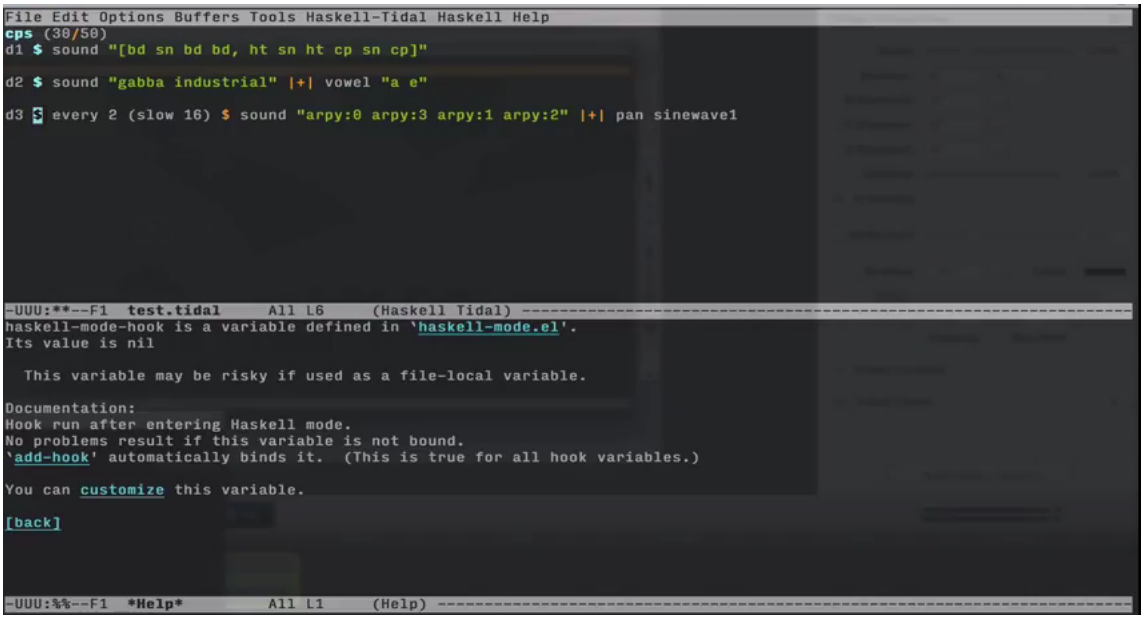

Figura 5 – *Live coding* com Tidal **Fonte**: <https://www.youtube.com/watch?v=ZQqg9Ghxruo>.

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 possiblitaram 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 eletronica 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) [35]

Collins e McLean apresenta dados a respeito da história da *algorave*: em 1992 uma performance entitulada *Cybernetic Composer* de Charles Ames; passando pelo *Aphex Twin* (Richard David James), que reinvindica em 1997 o termo (interessante do ponto de vista de gênero musical) *live club algorithm*; em 2000 o *Slub*, citado no inicio deste capitulo (na época Adrian Ward, Alex McLean), realizam performances, autodenominadas *generative*

[35] 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.*

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. [36]

2.10 *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):

Aqui as correções ainda não foram aplicadas, apenas sessões foram ajustadas, suas ordens

> 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.
>
> • 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. [37]

[36] COLLINS; McLean, 2014, loc. cit..

[37] 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*

"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*[38] *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.

"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 Artifical" entendo que o *live coder* pode criar *mini-linguagens* ou Linguagens de Domínio Específico (DSL)[39] que possibilitam criar programas para criar um espetáculo audiovisual ou musical. Segundo Collins e McLean

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.

[38] 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>). Nesse sentido, um *bug* em um programa é uma falha de operação, geralmente causada por algum erro de lógica, por parte do programador.

[39] 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 relaçao 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.*

(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 emabaralhamento 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 contruções 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)[40]

"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.

> 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

[40] Tradução nossa de: *There are increasingly user friendly "mini-language" livecoding systems which facilitate loop and layer-centric con-structions 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.*

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)[41]

[41] 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 inteligence of the system performing the piece.*

Parte II

Implementação computacional

3 Mapeamento dos gêneros musicais no *live coding*

Neste capítulo descrevo o *live coding* sob a ótica de categorizações do mercado musical, ou nas palavras de (JR., 2003; Sá, 2006; Sá, 2009), gêneros musicais. Esta sessão tem por função oferecer ferramentas de conhecimento, para lidar com os dados apresentados no Capítulo 4.

Na **??** realizo uma contextualização do termo *gênero musical*. Na subseção 3.1.1 contextualizo o gênero musical no *live coding*.

Ao contextualizar gênero como uma categorização do mercado, é pressuposta dinâmicas sonoras, sociais e corporais dentro um ambiente que gera um capital simbólico. Suponho que parte da produção do *live coding* absorveu sonoridades de diferentes culturas musicais. Nos lados mais extremos: *1)* aquilo que será definido como *música-popular massiva*; e *2)* aquelas tendências conhecidas como *noise* e *glitch*. Cada uma delas gera um capital simbólo diferente, de acordo com seus contextos sociais (o ambiente de performance, como é performatizada a música, que músicas lembram o que está sendo tocado, a luz utilizada durante a performance, a roupa que se veste na performance, etc.).

Por outro lado, absorver diferentes sonoridades não significa que todas caracterísicas sociais serão incluídas integralmente. Neste sentido, busco contextualizar na **??** o que é gênero musical, como uma articulação entre os sujeitos sociais e mercado regulador. Na subseção 3.1.1 busco contextualizar o *live coding*...

3.1 Categorização de sonoridades no *live coding*

O próprio conceito de gênero, segundo Judith Butler, é tão complexo que "exige um conjunto interdisciplinar e pós-disciplinar de discursos" (Problemas de gênero, p.12) . Embora essa conceitualização seja própria da crítica à divisão dos gêneros sexuais, ela não deixa de ser elucidar o tratamento dos gêneros musicais no *live coding*.

O gênero musical é um processo de categorização de sonoridades, construtos sociais e modos de performance com o corpo. Isto é, são reconhecidos fatores *intra-musicais* e *extra-musicais* que definem cada gênero musical.

A partir de Simon Frith, Janotti Jr. (2003, p. 32-37) discute o gênero musical como "um modo de definição da música em relação ao mercado, do potencial mercadológico presente na música". Simone Pereira de Sá (2006) reforça esta concepção de uma lógica de mercado. Ao repetir a pergunta de Janotti Jr, podemos ter uma idéia mais clara de gênero musical. **Com que se parece este som? ou do ponto de vista da indústria,'quem vai consumir esta música?** (JR., 2003 apud Sá, 2006, p. 15).

Ambos concordam que, o mercado musical é um importante dispositivo para regulação das categorias de gênero. Mas ele sozinho não é capaz de definir como uma sonoridade será classificada . Jr. lembra de "(. . .) exemplos que mostram que algumas

divisões comerciais, antes de se basearem em gêneros musicais, são efetuadas por padrões temporais, gêneros sexuais e/ou feixes linguísticos e geográficos". Uma *categorização de sonoridades* existe nas esferas de negociação entre mercado musical e os sujeitos participantes deste mercado, isto é, os "fãs, músicos, críticos e produtores". Também existe nas esferas musicais e técnicas, como uma música como é tocada, tipos de vozes, famílias de instrumentos existentes, sons eletronicos, acústicos, ritmo (se existe), melodia (se existe), drones (se existe). Existe não apenas no universo psicoacústico ou estético (fatores *intra-musicais*), mas também nas convenções sociais locais para performatizar este ouvir (ritos), e o capital simbólico derivado. Se de um lado a categorização das sonoridades é um modo de definição da música em relação ao mercado, por outro são "competências diferenciadas para que se construam determinados quadros de valor em relação a certas expressões musicais".

Simone Pereira de Sá (2009, p. 4-10) argumenta que *gênero musical* é uma categoria tradicional de classificação no universo da cultura da *música-popular massiva*. Ela mesma não dá conta dos fatores *extra-musicais* em rede. Para os sujeitos (fãs, produtores, críticos) é estar atualizado quanto às novidades de sua área de atuação, locais de eventos, novas técnicas, novos grupos, possivelmente o que se veste e o que se consome em reuniõs sociais (sempre lembrando que este consume envolve capital material). Para o mercado musical, significa saber o que estes sujeitos sociais tocam, ouvem, falam, vestem, dançam, compram. Busco então utilizar o termo para significar o gênero musical.

Sá (2009) ainda define *processos de rotulação*, onde *disputas simbólicas* permeiam uma *música-popular massiva*. Neste universo, grupos de indivíduos são reduzidos a *comunidades de gosto*. Para o Capítulo 4 busco discutir : *i)* qual é o mercado musical do *live coding*, i.e, quem produz, ouve, grava, vende, compra? *ii)* é uma música massiva ou se utiliza da música massiva? *iii)* existe um aspecto *intra-musical* ? *iv)* quais são os aspectos *extra-musicais*?

3.1.1 Contexto de performance do *live coding*

No capítulo 4 discuto um estudo de caso a respeito do que se toca e o que se ouve em um mercado específico (*Soundcloud*). Dados são derivados de buscas utilizando *hashtags*. As *hashtags* são um dispositivo utilizado para definir um gosto ou ua prática musicalDeixo ao leitor o julgamento se eles foram adequados ou não para descrever o *live coding*.

Destes resultados observei a rotulação de diversas categorias musicais. Ao que se entende por *live coding*, podemos associar o *algorave*, que por sua está relacionado a categorias como *electroacoustic*, *noise*, *electronic*, *algorithmic*, *algopop*. Segundo McLean (2014), estas manifestações podem ser contextualizadas em quatro tipos peculiares de performance sociais, com o público ou entre os executantes, com o corpo e uma premissa

técnica:

> Codificar musica ao vivo traz pressões particulares e oportunidades para o planejamento da linguagem de programação. Para reiterar, isto é onde o programador escreve o código para gerar música, onde um processo em execução contnuamente assume mudanças no seu código, sem quebrar com o resultado musical. A situação arquetípica tem o programador no palco, com sua tela projetada para a audiência ver seu trabalho. Isso poderia ser tarde da noite com uma audiência de uma casa noturna (...), ou durante o dia para uma audiência sentada em uma sala de concerto (isto é, perforamnce de um conjunto de *laptops*; Ogborn 2014), ou em uma performance colaborativa de longa duração (isto é, *slow coding*; Hall 007). O executante pode se juntar a outros *live coders* ou músicos instrumentais, ou mesmo com coreógrafos e dançarinos (McLean, 2014, p. 63) [1]

[1] Tradução nossa de: *Live coding of music brings particular pressures and opportunities to programming language design. To reiterate, this is where a programmer writes code to generate music, where a running process continually takes on changes to its code, without break in the musical output. The archetypal situation has the programmer on stage,with their screen projected so that the audience may see them work.This might be late at night with a dancing nightclub audience (e.g.at an algorave; Collins and McLean 2014), or during the day to aseated concert hall audience (e.g. performance by laptop ensemble;Ogborn 2014), or in more collaborative, long form performance(e.g. slow coding; Hall 2007).The performer may be joined byother live coders or instrument al musicians, or perhaps even choreographers and dancers (Sicchio 2014), but in any case the programmer will want to enter a state of focu sed, creative flow and work beyond the pressures at hands.*

Os dados abaixo são parciais de um levantamento de músicas no *Soundcloud* utilizando palavras-chave[2] relativas à produção da prática discutida neste trabalho organizadas em categorias de informação.

Palavras-chave são os termos que apresentaram maior consistência de dados (considerados pelo ponto de vista abstrato do autor do trabalho): *a) algorave/algopop*: parte considerável da produção do *livecoding* realizada em ambientes noturnos, informais ou de entretenimento (possue relação com o elemento dança); *b) livecoding/live-coding*: apresentaram dados coerentes com a bibliografia pesquisada[3]; *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 prodção pode se encaixar como *livecoding* (um desses programas é o *Wavepot*); *d) algorithmic music*: utilizei o termo para apontar o quanto da música algorítmica publicada no *Soundcloud* poderia ser classificado como *livecoding*, em conjunto com outros termos que não foram abordados nesta pesquisa; *e) wavepot*: performances feitas com o aplicativo *Wavepot*, são por definição *livecoding*;

Categorias de informação são: *a)* ano de publicação (*created_at*); *b)* países (*country*); *c)* cidade (*city*); *d)* gênero musical(*genre*);e *e)* licensa de uso (*license*)

3.2 Plotagem em forma de torta

Os dados foram levantados entre janeiro e fevereiro de 2015; não realizamos desde então levantamentos por utilizarmos um outro sistema de visualização. Estão disponíveis na **??**

3.3 Plotagem em forma de círculos empacotados

Os dados utilizados são os mesmos. A diferença é o tratamento dos dados: correlação de dados por duas categorias diferentes. Nos exemplos abaixo, salientei comparações de ano e gênero (Figura 6 e Figura 7) e país e gênero (Figura 8 e Figura 9).

[2] No sentido de um parâmetro de procura em *Query-strings*; para mais informações, ver <https://en.wikipedia.org/wiki/Query_string>

[3] 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*.

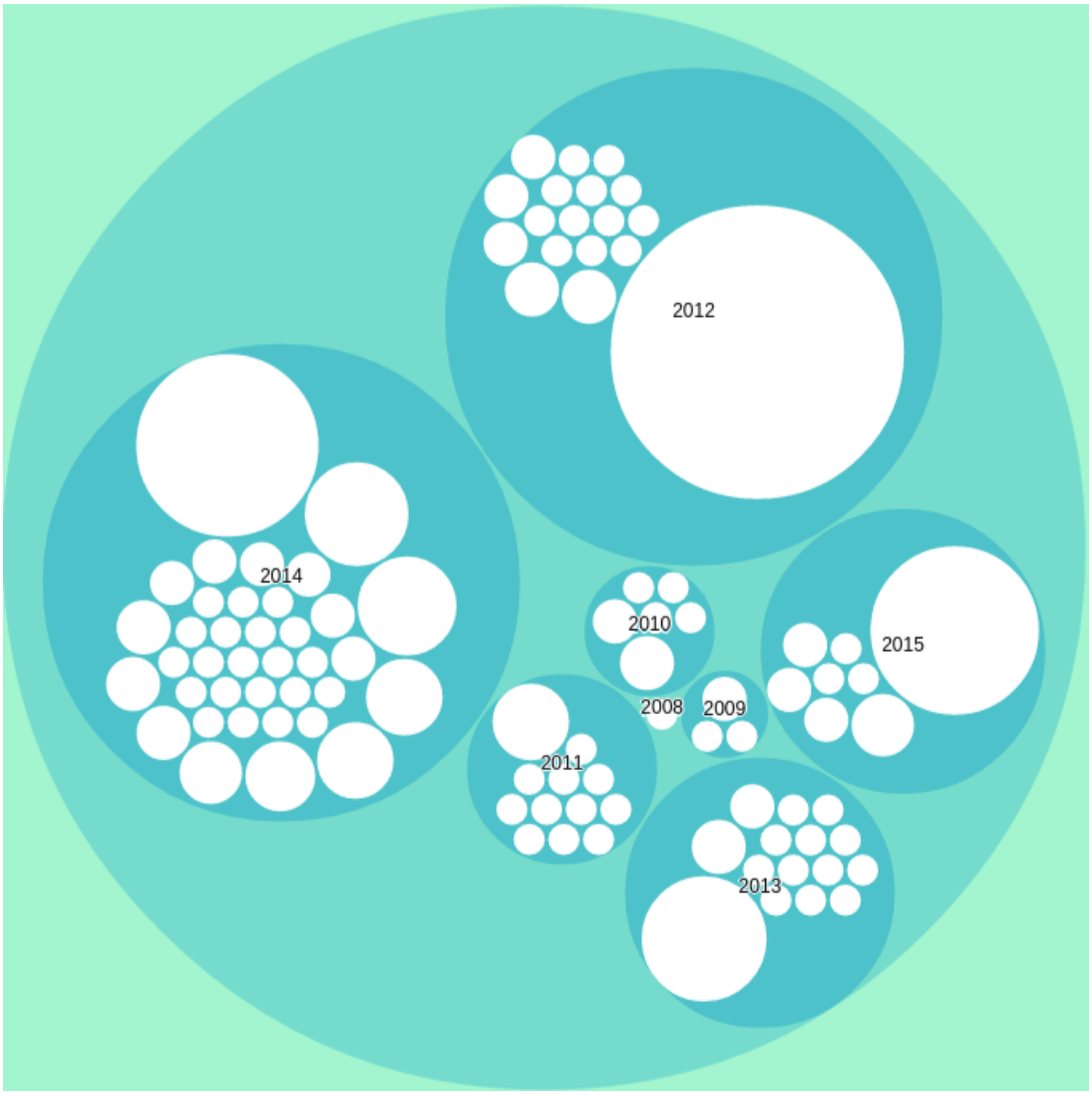

Figura 6 – Empacotamento de informações a respeito de gênero musical a partir de anos de produção

Figura 7 – Detalhamento de informações a respeito de gênero musical a partir de anos de produção

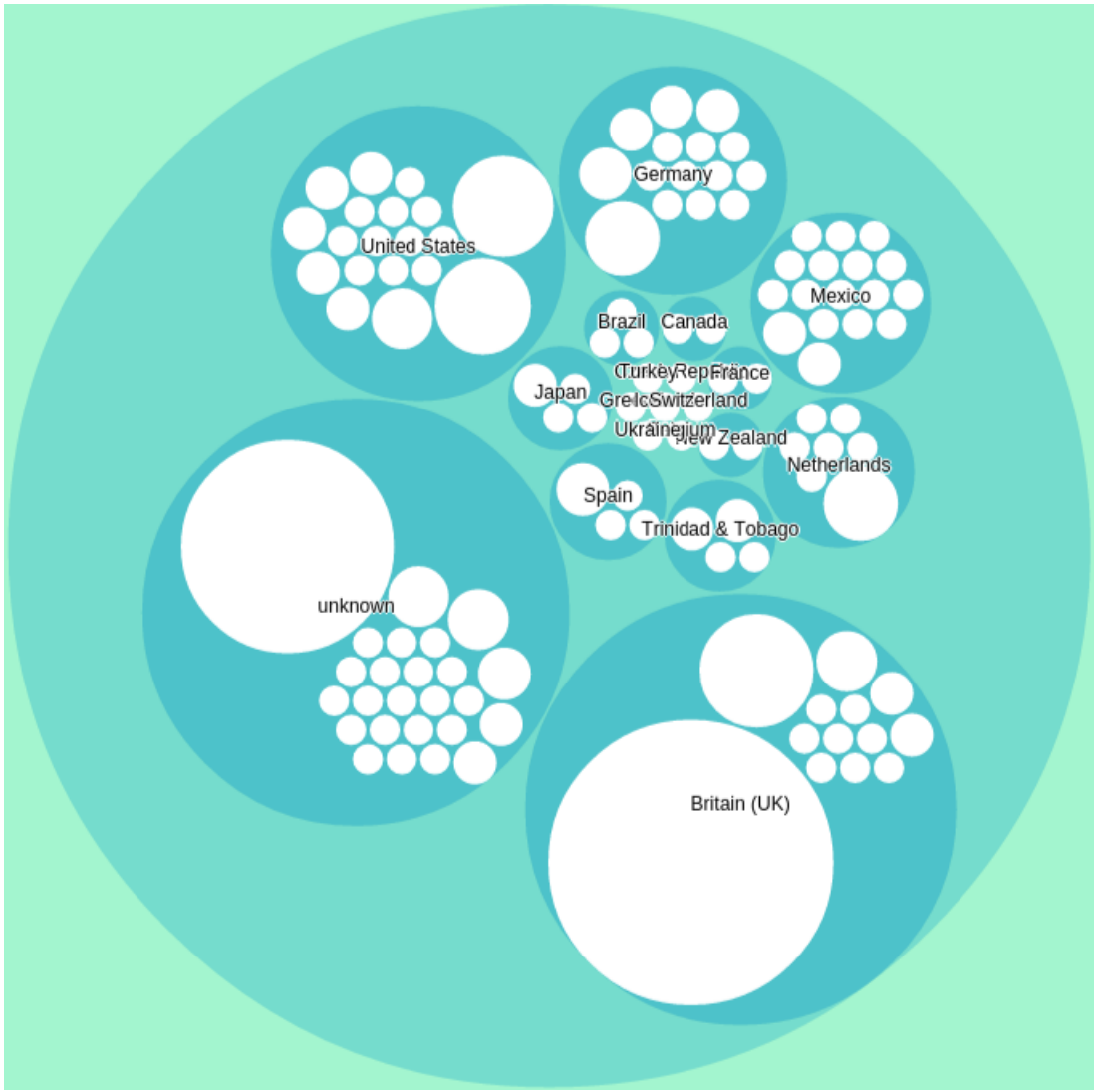

Figura 8 – Empacotamento de informações a respeito de gênero musical a partir de países onde ocorreram as produções

Figura 9 – Detalhamento de informações a respeito de gênero musical a partir de países onde ocorreram as produções.

4 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 16, 20, 24 e 25.

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>. Citado 3 vezes nas páginas 17, 24 e 25.

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 23.

CHOMSKY, N. Three models for the description of language. MIT, p. 12, 1956. Citado na página 30.

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 24.

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 34, 35 e 37.

ENO, B. Base de dados, *Music for Airports liner notes*. 1978. Disponível em: <http://music.hyperreal.org/artists/brian_eno/MFA-txt.html>. Citado 2 vezes nas páginas 29 e 30.

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 27.

GRIFFITHS, D. Fluxus: Scheme livecoding. 2008. Disponível em: <http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>. Citado na página 35.

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 45.

IAZZETTA, F. *Música e mediação tecnológica*. [S.l.]: Ed. Perspectiva-Fapesp, 2009. ISBN 9.788527308724E12. Citado 2 vezes nas páginas 33 e 34.

JR., J. S. J. À procura da batida perfeita: a importância do gênero musical para a análise da música popular massiva. v. 6, n. 2, p. 31–46, 2003. Citado 2 vezes nas páginas 12 e 42.

Junior, A. D. d. C.; Lee, S. W.; Essl, G. Supercopair: Collaborative live coding on supercollider through the cloud. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 152–158. ISBN 978 0 85316 340 4. Citado na página 12.

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 na página 8.

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

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>. Citado 2 vezes nas páginas 27 e 28.

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 7.

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 7.

MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 4 vezes nas páginas 7, 20, 21 e 22.

McCallum, L.; SMITH, D. *Show Us Your Screens.* Vimeo, 2011. Disponível em: <https://vimeo.com/20241649>. Citado na página 35.

McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: <https://www.academia.edu/467101/Music_improvisation_and_creative_systems>. Citado na página 13.

McLean, A. Making programming languages to dance to: Live coding with tidal. p. 63–70, 2014. Disponível em: <https://www.academia.edu/12765888/Making_programming_languages_to_dance_to_Live_coding_with_Tidal>. Citado 2 vezes nas páginas 43 e 44.

McLean, A. et al. *Visualisation of Live Code.* Alex McLean; making music with a text, 2010. Disponível em: <http://yaxu.org/visualisation-of-live-code/>. Citado na página 17.

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 14.

McLean, A.; WIGGINS, G. *Patterns of movment in live languages.* 2009. Disponível em: <https://www.academia.edu/7249277/Patterns_of_movement_in_live_languages>. Citado 3 vezes nas páginas 8, 20 e 24.

McLean, A.; WIGGINS, G. *TIDAL – PATTERN LANGUAGE FOR LIVE CODING OF MUSIC.* [S.l.]: Centre for Cognition, Computation and Culture; Department of Computing Goldsmiths, University of London, 2010. Citado na página 32.

MORI, G. Analysing live coding with etnographical approach. In: *ICLC2015 Proceedings.* [S.l.: s.n.], 2015. p. 117–124. ISBN 978 0 85316 340 4. Citado na página 12.

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 5 vezes nas páginas 7, 16, 20, 22 e 23.

Neto, J. B. Imre lakatos e a metodologia dos programas de investigação cientifica. (apresentação do pensamento do filósofo da ciência imre lakatos). p. 12, 2008. Disponível em: <http://people.ufpr.br/~borges/diversos/publicacoes.html>. Citado na página 8.

NUNZIO, A. D. *Genesi, sviluppo e difusione del software "MUSIC N"nella storia della composizione informatica*. phdthesis — Facoltà di Lettere e Filosofia - Università degli Studi di Bologna, 2010. Citado na página 20.

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>. Citado 3 vezes nas páginas 20, 29 e 30.

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 36.

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 na página 28.

ROADS, C. Times scales of music. In: *Microsound*. First MIT paperback edition 2004. [S.l.]: MIT press, 2001. p. 1–41. ISBN 978-0-262-18215-7. Citado 2 vezes nas páginas 31 e 32.

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 16 e 25.

ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado na página 25.

ROWE, R. J. *Machine Listening and Composing: Making Sense of Music with Cooperating Real-Time Agents*. Media Arts and Sciences — MIT, 1991. Disponível em: <http://r.duckduckgo.com/l/?kh=-1&uddg=http%3A%2F%2Fdspace.mit.edu%2Fbitstream%2Fhandle%2F1721.1%2F29083%2F32147968.pdf>. Citado na página 32.

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>. Citado 2 vezes nas páginas 13 e 18.

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>. Citado 2 vezes nas páginas 13 e 18.

SOARES, G. R. *Luteria Composicional de algoritmos pós-tonais v1.1FINAL*. Prévia da dissertação para a banca de quali- ficação para o Mestrado em Arte, Cultura e Linguagens do IAD-UFJF. — UFJF, 2015–03–13. Disponível em:

<https://github.com/glerm/luteria/raw/master/LUTERIA_2015janeiro.pdf>. Citado na página 30.

Stockhausen, K. How time passes by. p. 10–39, 1959. Disponível em: <http://www.artesonoro.net/artesonoroglobal/HOW%20TIME%20PASSES%20BY.PDF>. Citado na página 23.

SuperCollider.ORG. *SuperCollider Overviews: JITLib - An overview of the Just In Time library*. 2014. Citado na página 7.

Sá, S. P. d. A música na era de suas tecnologias de reprodução. v. 12, n. 19, p. 19, 2006. Disponível em: <http://www.compos.org.br/seer/index.php/e-compos/article/viewFile/92/92>. Citado 2 vezes nas páginas 12 e 42.

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 3 vezes nas páginas 12, 42 e 43.

VIEIRA, V. et al. Vivace: A collaborative live coding language. arXiv, n. 1502, p. 16, 2015. Disponível em: <http://arxiv.org/abs/1502.01312>. Citado 2 vezes nas páginas 12 e 13.

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 24, 25, 26, 27, 29, 33 e 37.

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 27.

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 16.