Guilherme Martins Lunhani

Variedade de gêneros musicais em três improvisações de Andrew Sorensen.

31 de outubro de 2015

Guilherme Martins Lunhani

Variedade de gêneros musicais em três improvisações de Andrew Sorensen.

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

31 de outubro de 2015

,

Agradecimentos

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

Aos dez mil seres que fugiram do repouso e da obscuridade e que chegaram em direção ao movimento e ao brilho. E tocaram o som imaterial que constituiu a harmonia.

Aos homens que detestaram o abandono, a indignidade e a obscuridade. Individualmente, qualificaram a si mesmos assim.

Aos que diminuiram para que outros possam crescer, e aos que cresceram para outros fossem diminutos.

À Educação, cumeeira de ferro que a morte não atinge, esforço-me para agir de acordo.

Aos professores Luiz Eduardo Castelões, Alexandre Fenerich e Flávio Luiz Schiavonni por serem fundamentais nas críticas e no apoio institucional, no momento em que vislumbramos o sucateamento acadêmico. À FAPEMIG por apoiar esta pesquisa. Aos amigxs que estão, ou moraram em Juiz de Fora, fundamentais para a continuação do trabalho (Glerm, Anna Flávia, Tiago Rubini). Aos amigxs de campinas que de alguma forma me ajudaram a suportar a distância (Doshi, Larissa, Dani, Tati, Catatau, Rebechi, Felício).

TODO...

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

---

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

Curtis Roads (2001)

Resumo

Esta pesquisa foi construída a partir da seguinte pergunta: ao discutir uma prática conhecida como *live coding*, quais categorizações sonoras (ou gêneros musicais) são contextualizadas?

Contudo, a variedade dos diferentes exemplos da pergunta forçou a redução do *universo de conceitos* inerente às categorizações sonoras estudadas. Neste sentido, o texto deste trabalho busca partir de um suposto espaço conceitual generalizado (Capítulo 1); um método de pesquisa que contemple as multiplicidades deste espaço conceitual (Capítulo 2); uma contextualização histórica, anterior à concepção deste espaço conceitual (Capítulo 3); apresentação de alguns casos do compositor Australiano Andrew Sorensen (Capítulo 4).

Estes casos foram escolhidos por representarem, em um mesma pessoa, uma plaralidade de práticas musicais. Não é nossa intenção a estéticas musical, mais sim alguns meios técnicos pelos quais a variedade delas emerge. Mais especificamente, com uma mesma técnica de improvisação (ou *live coding*), Sorensen improvisa desde *Jazz*, *Minimalismo* de um Steve Reich, *Músicas-populares massivas*, e sínteses sonoras. Dois destes casos são de interesse por envolver uma busca pelo controle do Piano MIDI e o Piano Acústico. O terceiro é interessante por lembrar uma mistura das estéticas da *Computer Music* dos anos setenta e 80, e a música de dança para ambientes sociais noturnos, algo que denominado na cena européia por *algorave*. Por outro lado, os três são paradigmáticos por envolver replicação de estilos.

Palavras-chaves: *Live coding*. Categorizações Sonoras. Sorensen

Sumário

Introdução

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

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

Deste último *software*, foi possível extrair o objeto de pesquisa. Ao utilizar este ambiente de síntese sonora e composição algorítmica, é possível realizar improvisações musicais, durante a atividade de codificação de um programa. Esta atividade peculiar ficou conhecida como *livecoding*:

> *Programação imediata* (ou: programação de conversa, 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)[5]

A definição é apropriada para um verbete de um documento técnico. Porém reduz o *livecoding* ao modo de operação com o computador, e não descreve possíveis resultados musicais. Teoricamente, qualquer estética musical pode ser reproduzida durante uma sessão de improvisação (*livecoding session*). Nesse sentido, investiguei o *livecoding* como, nas palavras de Alex McLean (2006), um *universo de conceitos*. Dependendo de como

[1] Disponível em <https://csound.github.io/>.
[2] Disponível em <http://cycling74.com/products/max>.
[3] Disponível em <http://puredata.info/>.
[4] Disponível em <https://supercollider.github.io/>.
[5] 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.*

este universo é configurado, diferentes estéticas musicais podem emergir. Este tema será discutido no Capítulo 1.

No Capítulo 2 delimito os conceitos deste universo.

No Capítulo 3 descrevo quais são, segundo o Programa de Investigação Científica do *livecoding*, os predecessores da prática. Por outro lado, existem também manifestos que sedimentaram uma ideologia e uma heurística.

No Capítulo 4 investigar multiplicidades estéticas do *live coding* em alguns casos específicos, apresento„ três trabalhos do compositor e programador australiano Andrew Sorensen. O objetivo não é realizar uma análise pormenorizada de casos, mas sim destacar um fenômeno percebido durante a pesquisa. Trabalhei com a hipótese de que, durante os (aproximadamente) 15 anos de emergência do *live coding*, ocorreu uma mistura de várias categorizações musicais no seio da prática. A análise pormenorizada desfoca a atenção na variedade, algo que moveu o pesquisador a pesquisar.

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

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

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

O fragmento acima sugere que a Música, usada sozinha ou com outra linguagem (artes do corpo e audiovisual), articula os afetos entre pessoas, e portanto, possue uma natureza permissiva. Não apenas tangente às linguagens audiovisual e do corpo, mas também na capacidade de agregar pessoas. Essa natureza permissiva é característica de um tipo de comunidade.

Através de Victor Turner e Harris (1969), Carolina Prospero (2015) discute se, no *live coding*, o conjunto de intérpretes e público do *live coding* forma uma *communita*, ou "comunidades abertas, que diferem das estruturas de comunidades fechadas"(TURNER; HARRIS, 1969 apud PROSPERO, 2015, p. 71).

Ao tratar esta comunidade como *communita*, é aberto espaço para discutir uma *liminaridade* própria do *live coding*. Liminaridade é definida por Di Prospero como "modelos que são, em um nível, reclassificações da realidade e da relação do homem com a sociedade, natureza e cultura" (*Ibdem*, *idem*):

> Liminaridade é apresentada de algumas formas no *live coding*: novos projetos e propostas, a procura por desmistificar a relação com a tecnologia, tornar o código um artesanato ou um produto artístico, mas, mais do que isso, na construção de uma comunidade participativa, uma comunidade coletiva imaginada. Liminaridade do espaço para se expressar e construir várias propostas que suscitam transformações não somente nos campos artísticos e culturais, mas também institucional, a cena do *live coding* envolve construir o mundo inteiro, um mundo da arte nos termos de Becker (1982). **De Acordo com o autor, aquele que colabora na produção de uma obra de arte não o faz a partir do nada, mas em acordos passados ou costumes, convenções, que geralmente cobrem as decisões tomadas, e isso torna as coisas mais simples** (. . .), o caso do *live coding* contribui amplamente para a aceitação da mudança como uma constante, dentro de um quadro em que a expressão artística é um processo em vez de produto acabado. (*Ibdem*, *idem*). [2]

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

[2] Tradução nossa de: *Liminality is present in some ways in live coding: new projects and proposals, the search to demystify the relationship with technology, making the code a craft or artistic product, but, more than anything, in the construction of its "participatory community", a collectively imagined community. Liminality of space to express themselves and build various proposals raises transformations not only in the artistic or cultural field but also institutional, the live coding scene involves building an entire world, an art world in terms of Becker (Becker 1982). According to the author, who cooperates in*

O*live coding* possui um ritual específico, costumes que caracterizam a prática como tal. Este ritual urbano, busca constantemente acessar um estado subjetivo onde limites conceituais estão em um espaço de transição. A forma deste ritual, no entanto, depende do espaço físico onde uma improvisação é realizada. Alguns casos acontecem salas de concerto, em salas de universidade, em uma casas noturna, ou em um ambiente informal. Por outro lado, estes espaços podem ser simplismente virtuais, como o caso de dois vídeos analisados na **??**.

A autora faz um apontamento que será trabalhado no Capítulo 3. O *live coding* não é uma prática emergente sem respaldo em práticas musicais anteriores. Tais exemplos serão apresentados como predecessores, fundamentais para a definição do universo de conceitos do *live coding*.

Neste sentido, buscamos definir na próxima seção o que é um universo de conceitos.

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, diferentes performances de *live coding* possuem diferentes configurações destas classes lógicas. Os objetos, características e processos de uma improvisação de *Jazz* são diferentes daqueles de uma improvisação de *Música eletroacústica* ou de uma improvisação de *Músicas-populares massivas*[3]

Esta contraposição, específica entre o *Jazz* e *Músicas-populares massivas*, será trabalhada no Capítulo 4, ao descrever *Study in Keith* (2009)[4], *Day Of The Triffords*

producing a work of art do not do it from nothing but rest on past agreements or custom / conventions, which usually cover the decisions to be taken, and this makes things simpler(...), the case of live coding broadly contributes to the acceptance of change as a constant, within a framework in which artistic expression is a process rather than finished product.

[3] Sobre Músicas-populares massivas e regras de gêneros musicais, Cf. Sá, 2009.

[4] Disponível em <https://vimeo.com/2433947>.

(2009)[5].

1.2 A questão da tradução

Giovanni Mori denomina a prática com dois termos separados, isto é, *live coding*. Tomando esta separação, o prefixo *live* é taduzido como "ao vivo", e o sufixo *coding* como "codificar". Uma performance cujo ato é improvisar leis ou fórmulas dispersas. - Porém a tradução falha em identificar questões-satélites, como por exemplo, a própria Música. Para identificar algumas palavras-chaves, uma compilação dos dos anais do ICLC 2015 (MCLEAN et al., 2015) pode ser esclarecedor.

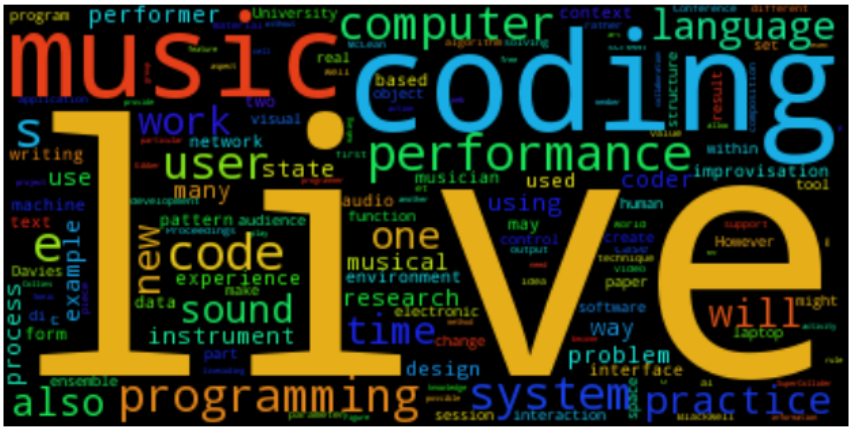

Figura 1 – Nuvem de palavras dos anais ICLC2015 **Fonte**: autor.

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). [6].

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*[7]. Ambos são ambiente de programação para de síntese sonora

[5] Disponível em <https://www.youtube.com/watch?v=DGcE8P5F29A>.
[6] O método de extração será explicado em um apêndice oportuno.
[7] Disponível em <http://gibber.mat.ucsb.edu>

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

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 3, exploro estas ações sob a ótica da Música de Processos de Steve Reich.

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

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.

2 Metodologia

McLean et al. (2010) corrobora com definição de *live coding* 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" [1], o segundo descreve como um construto social que derivou em um Programa de Investigação Científica (PIC):

> 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).[2]

Atualmente um outro sítio, *Live Code Research Network*[3], é outra referência. As liminaridades expandem, a o *live coding* se torna mais uma técnica do que estética. Por exemplo: a utilização de linguagens visuais, como *PD* ou *Max/Msp*, ao vivo (*live patching*) são um caso à parte ou devem ser incluídas no objeto de pesquisa? É pertinente chamar de *live coding* uma performance que utiliza o *Live*!*Ableton*? Ou apenas atividades em que se codificam textos? Utilizar dispositivos diferentes do *laptpop*, como corpos dançantes, é *live coding*?

Uma pergunta mais específica irá delinear o método, de um universo de conceitos U, contendo infinitos conceitos c, para um subconjunto de três conceitos: $jazz$, $minimalismo$, $algorave$.**O que pode ser pressuposto musicalmente, entre Andrew Sorensen, e seu público, em três improvisações utilizando a linguagem LISP?**

2.1 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. Notamo as seguintes referências aos assuntos: (1) Música; (2) Cinema; (3) Instalações Artísticas; (4) Educação; (5) Ciências da Computação; (6) Semiologia (7) Lógica; (8) Etnografia.

Para discutir esta multiplicidade de assuntos, recorremos ao sociólogo Boaventura de Souza Santos (2007), Santos (2008). Sugerimos o *live coding* como uma feira de idéias,

[1] 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

[2] 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).*

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

onde se compra e vende conceitos conforme a necessidade, nesse caso, a própria tese de mestrado. Santos sugere um método consciente de pluralidades e desproporções daquilo que se quer saber.

Entendemos que pesquisador será forçado, na academia, a limitar aquilo que cada vez mais se expande. Santos discute isso da seguinte maneira:

> 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 a aplicação dos estudos sociais, defino, este trabalho, a *ecologia de saberes* como **um estudo de um subconjunto de três elementos categóricos no universo de conceitos U do *live coding***.

2.1.1 Exemplo da multiplicadade de um universo de conceitos

Durante a pesquisa, investigamos um campo específico, mas que não será utilizado como trabalho final. Por outro lado, permitiu visualizar os conceitos c de universo de conceitos U em uma mídia social (*Soundcloud*). Não quantificamos, por outro lado, as variáveis O, F e P.

As Figura 2, **??**, **??** e **??** ilustram a multiplicidade de conceitos c. Agrupados por data, país, cidade e *hashtags* que "delimitam" um gênero musical, possibilitaram reduzir um pouco o estudo, mas não completamente.

Os seguintes os termos delimitados foram: *a) livecoding/live-coding*: dados coerentes com a bibliografia pesquisada[4]; *b) algorave/algopop*: parte considerável da produção do *livecoding* realizada em ambientes noturnos, informais ou de entretenimento (possue relação com o elemento dança); *c) bytebeat*: parte considerável de uma técnica de programação musical descrita pela primeira vez por Heikkilä (2011) e aplicada no *livecoding*, isto é, apenas um fragmento dessa prodção pode se encaixar como *livecoding* (um desses programas

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

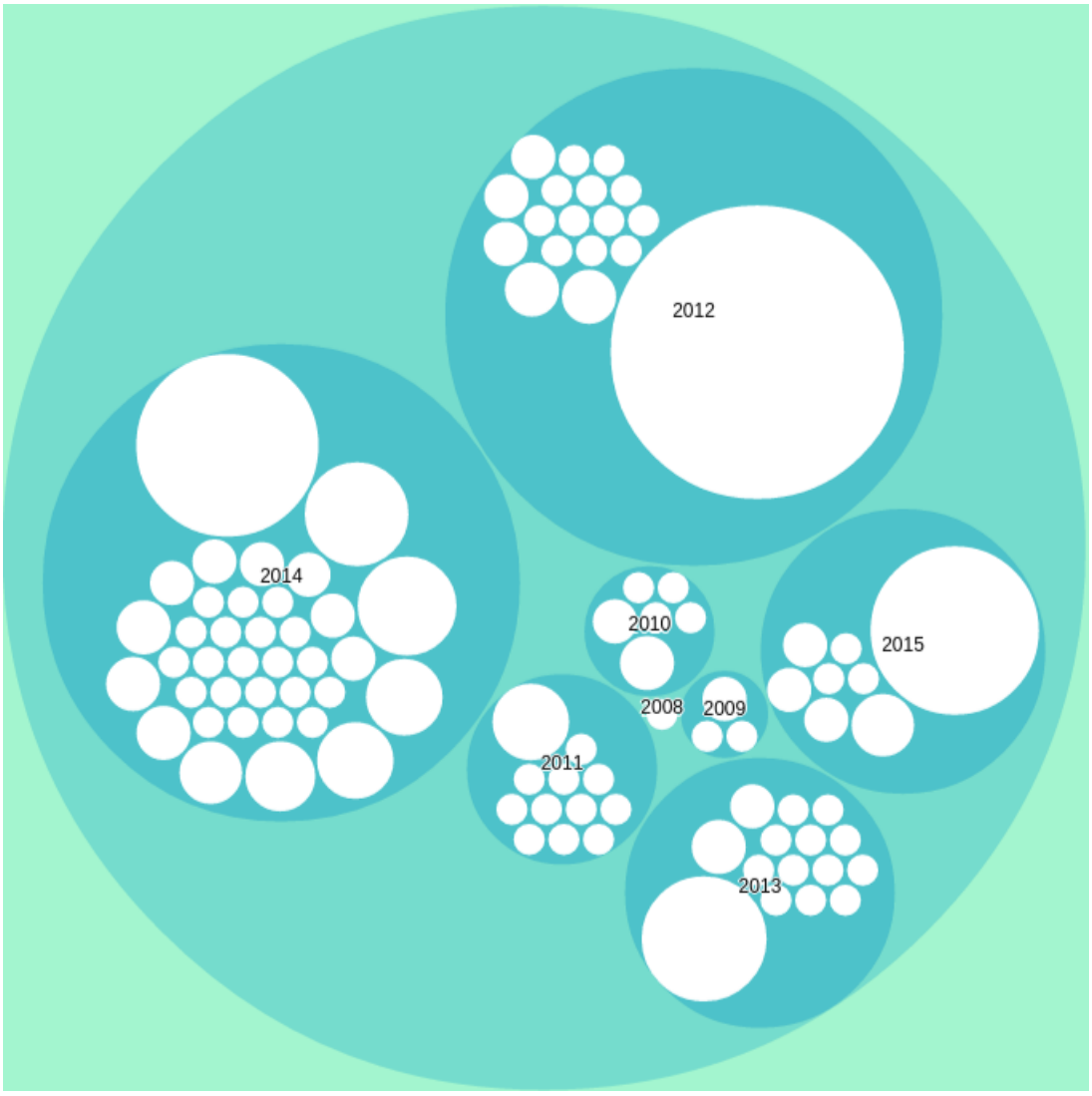

Figura 2 – Empacotamento de informações a respeito de gênero musical a partir de anos de produção

é o *Wavepot*); *d) algorithmic music*:música algorítmica, seguindo a subdivisão Música Gerada por Computador, ou *Computer Generated Music*(COPE, 2008).

Os dados levantados são de janeiro e fevereiro de 2015; não realizamos mais levantamentos. Os motivos foram: *i)* a dificuldade de análise das assimetrias observadas, que requerem maior experiência com técnicas estatísticas. *ii)* os dados ilustram as variedades de categorizações musicais no *live coding*. Pode ser interessante observar se tais variedades se aplicam em um único *live coder*.

Antes de discutir três casos específicos de um *live coder* (Sorensen), contextualizamos os *live coding* no Capítulo 3. Alguns precedentes históricos do *live coding*, como *GROOVE*, o compositor *Pietro Grossi*, a *Live Computer Music* da Baía de São Franscisco entre os

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

anos 70 e 80 nos Estados Unidos, e a emancipação de uma organização, TOPLAP[5], e o destaque de *live coders* ingleses, foram fatores para descrever, neste trabalho, o *live coding* como técnica de improvisação artística original de um Norte político e econômico; delimitar o *live coding* como um Programa de Investigação Científica em universidades; ou descrever o *live coding* como dispositivo de improvisação para replicação de categorizações musicais.

2.2 Justificativa

A investigação dos precedentes históricos permitem observar o *núcleo de pesquisa* e a *heurística* (Lakatos, 1970; Neto, 2008) do *live coding*, e separá-lo, por exemplo, da *live Computer Music*.

[5] Disponível em <http://www.toplap.org>.

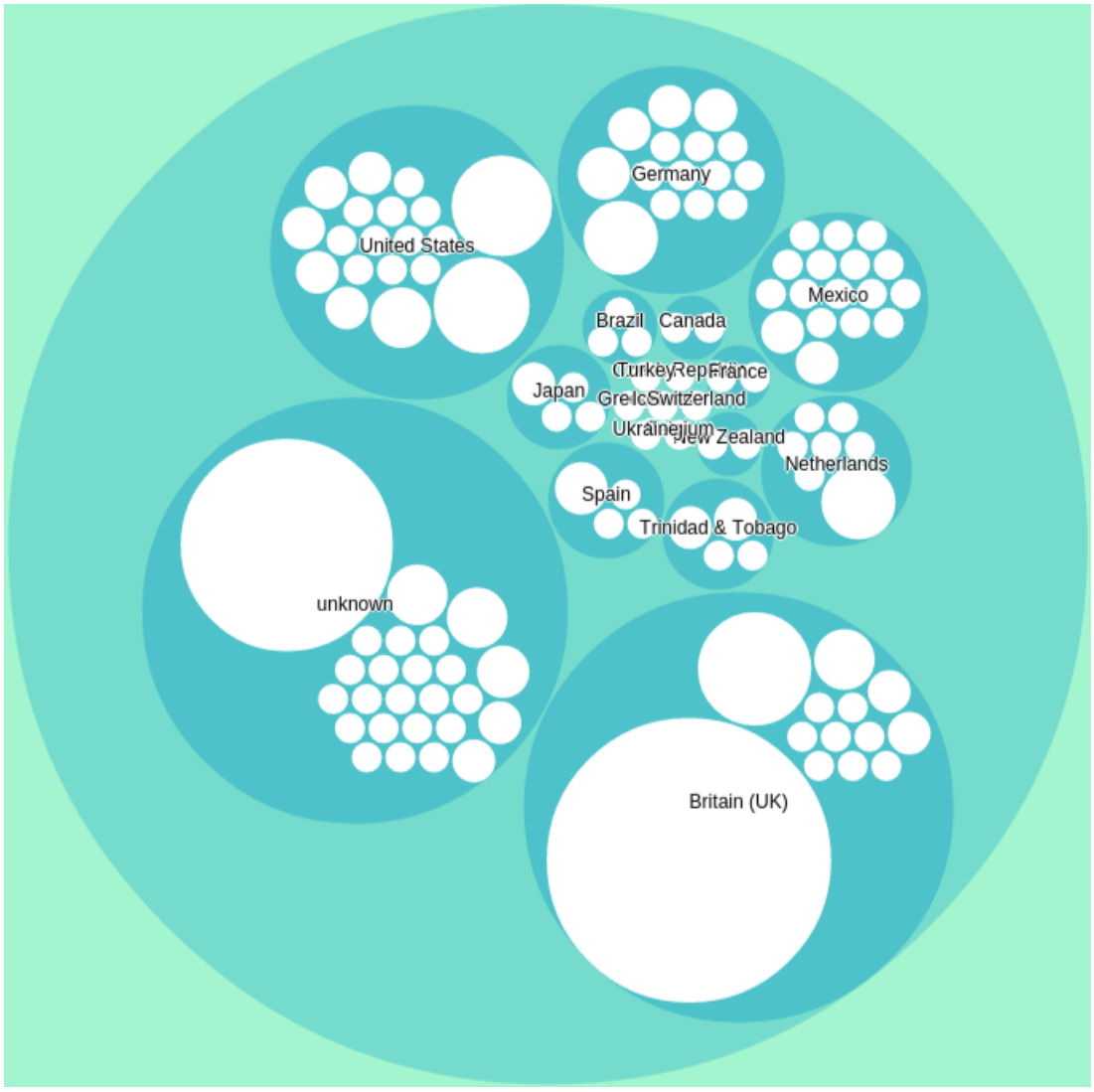

Figura 4 – Empacotamento de informações a respeito de gênero musical a partir de países onde ocorreram as produções

Um paradigma comum entre todos exemplos citados no último parágrafo da seção anterior, dos mais antigos, até os mais atuais, é a utilização de um ou mais computador(es), ou partes deles (microchips), em um contexto de performance artística.

Dos exemplos mais atuais, as publicações de manifestos no início dos anos 2000, são considerados, neste trabalho, como os primeiros pontos delimitadores. Tais manifestos incluem, além da Música de processos (de um Steve Reich, Alvin Lucier, ou até mesmo o minimalismo de Philip Glass), O que diferencia este manifesto de seus pr

Para lidar com as liminaridades do *live coding*, as seguintes improvisações foram escolhidas: *i) Study in Keith* (2009)[6], uma improvisação de códigos que busca replicar o estilo de Keith Jarret); *ii) The Disklavier Sessions* (2010)[7], um *teaser* demostrativo.

[6] Disponível em <https://vimeo.com/2433947>.
[7] <https://vimeo.com/50061269>.

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

Demonstra o controle de um piano híbrido da Yamaha, *Disklavier*, para replicar *Jazz* e um *Minimalismo* semelhante a um Steve Reich; *iii) The Day of Triffords* (2009)[8], ilustração do gênero musical *algorave*, como replicação músicas eletrônicas de ambientes urbanos noturnos. .

[8] <https://vimeo.com/2735394>.

3 Trabalhos Relacionados

3.1 Predecessores

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

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

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

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

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

Além deste primeiro manifesto, existe outro de grande importância, "Show us your screens", que define alguns cânones do *live coding*. Na subseção 3.1.6 tratarei do caso específico.

3.1.1 GROOVE

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

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

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

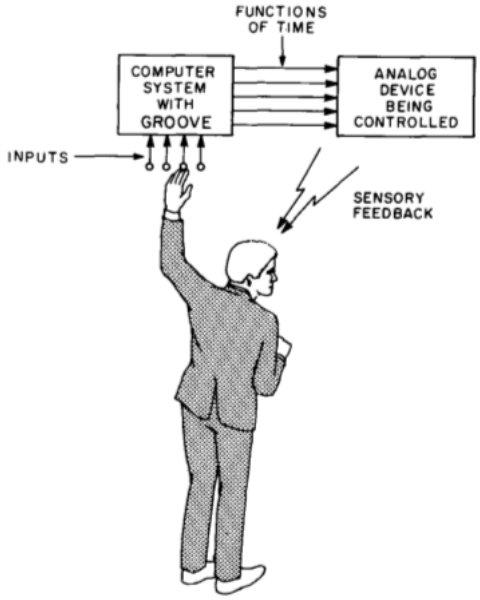

FIG. 1. Feedback loop for composing functions of time

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

coding:

> O GROOVE provê oportunidades para uma retroalimentação imediata de observações dos efeitos das funções temporais para as entradas do computador, que compõem a função. No modo de composição do sistema GROOVE, um ser humano está em um ciclo de retroalimentação, como mostrado na figura 1 [Figura 6]. Assim ele é 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

[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.*

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

3.1.2 Pietro Grossi

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

> (...) o 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*

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).

3.1.3 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]

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.

[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*

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

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

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.*

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

3.1.4 Just In Time (JIT)

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

---

[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>.

3.1.5 Live Algorithm Programming and Temporary Organization for its Promotion

Na **??**, Blackwell e Collins (2005) comenta a emancipação de um grupo conhecido como TOPLAP. Este acrônimo é de difícil definição. Deriva do manifesto "*Live Algorithm Programming and Temporary Organization for its Promotion*" Ward et al. (2004). No *Wiki* do site oficial[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

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:

[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.*

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

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

3.1.6 *Show us your screens*

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

> Exigimos:
>
> • Acesso à mente do intérprete, para todo o instrumento humano.
>
> • Obscurantismo é perigoso. Mostre-nos suas telas.
>
> • Programas são instrumentos que podem modificar eles mesmos.
>
> • O programa será transcendido - Língua Artificial é o caminho.
>
> • O código deve ser visto assim como ouvido, códigos subjacentes visualizados bem como seu resultado visual.
>
> • Codificação ao vivo não é sobre ferramentas. Algoritmos são pensamentos. Motosserras são ferramentas. É por isso que às vezes algoritmos são mais difíceis de perceber do que motosserras.
>
> Reconhecemos contínuos de interação e profundidade, mas preferimos:
>
> • Introspecção dos algoritmos.
>
> • A externalização hábil de algoritmo como exibição expressiva/impressiva de destreza mental.

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

[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".*

> • 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. [22]

"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*[23] *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.

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

[23] 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.

"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)[24] que possibilitam criar programas para criar um espetáculo audiovisual ou musical. Segundo Collins e McLean (2014), tais DSLs estariam no formato de "mini-linguagens" bem desenvolvidas para a tarefa específica de codificar música ao vivo, operando técnicas composicionais como a transformação de um padrão musical (como por exemplo, técnicas barrocas como inversão e retrogradação ou técnicas aleatórias, como 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)[25]

"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.

[24] 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.*

[25] 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.*

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

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

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

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

3.2 *Live coding* e Música de algoritmos

3.2.1 Música de algoritmos simples

Segundo Wooler et al., a música de processos é uma "Música resultante de um conjunto de processos colocados em movimento pelo compositor, tais como 'In C' de Terry Rilley e 'It's gonna rain' de Steve Reich" (WOOLER et al., 2005 apud ENO, 1996, p. 1)[27]; o que é chamado de "procedural" por Wooler et al., é chamado por Joshua Mailman

[26] 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.*

[27] 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*

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

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

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[29].

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**

[28] 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.*

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

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

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.

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[31], 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"[32]). 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[33]. Esta técnica de retroalimentação também pode ser utilizada no *live coding*, por exemplo, no SuperCollider (ver **??**).

---

[30] 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.*

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

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

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

3.2.2 Música de algoritmos complexos

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

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

> O conceito de música projetada especificamente como pano-de-fundo ambiental foi um pioneirismo da indústria Muzak nos anos cinquenta, e tem sido conhecido genericamente como Muzak. As conotações que este termo carrega são particularmente associadas com um tipo de material que 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 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) [34]

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 [35]; geralmente essas abordagens podem criar *drones* que ficam variando em seu espectro.

Um exemplo interessante de *live coding* com *drones*, que utiliza pequenos impulsos sonoros no SuperCollider (ver Figura 7). Aqui o processo de escuta é contínuo, semelhante à práticas da música eletroacústica,onde algoritmos são pré-concebidos e modificados durante a performance. Cole Ingraham utiliza um objeto que gera impulsos sonoros aperiódicos, e o 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

[34] *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.

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

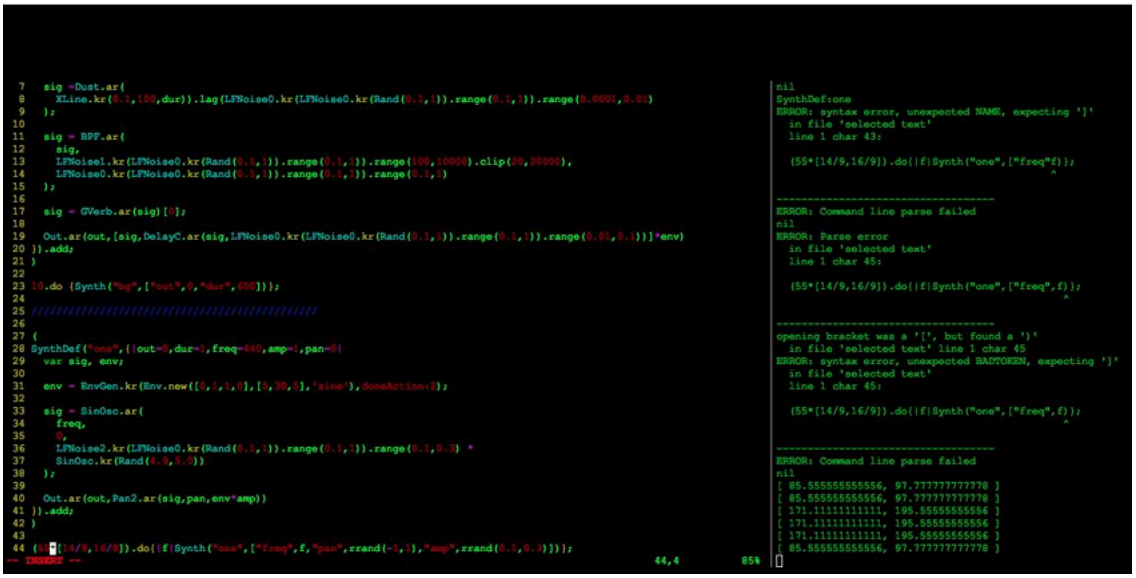

Figura 7 – Improviso com *drones* no SuperCollider **Fonte**: <https://www.youtube.com/watch?v=b8j4umQ2lIE>.

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

3.3 Prática DJ

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

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

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

Primeiro é necessário esclarecer se a apropriação foi intencional ou não. O discurso do manifesto de Ward et al. (2004) indica uma consciência parcial dos autores a respeito desta sinergia. No entanto é discutível se esse cruzamento de gêneros é mais um reflexo das transformações sociais e culturais que o computador trouxe consigo do que algo intencional: o uso deliberado de muitas ferramentas de um estúdio portátil, a expansão de possibilidades de produção musical através das redes de computadores, a troca de informações a respeito do uso de novos *softwares* entre músicos acadêmicos e não-acadêmicos , trouxe à tona diferentes comunidades daquela ou outra tecnologia. De forma semelhante, no *livecoding*, as novas possibilidades tecnológicas em contato com modos de fazer música já estabelecidos 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) [36]

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 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. [37]

[36] 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.*

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

4 Estudos de casos

Comparo três casos do pesquisador australiano Andrew Sorensen que considero simbólicos. As improvisações *Study in Keith* (2009), seção 4.1, *Day of Triffords* (2010), e *The Disklavier Sessions* (2012), serão analisadas com a intenção de expor uma variedade em um único *live coder*.

Isto é, se em teoria o *live coding* é uma ferramenta, uma *motoserra*, e pode replicar qualquer gênero musical, em casos específicos serão replicados recortes destes gêneros. Para investigar a replicação destes gêneros, serão analisados os universos de conceitos de cada improvisação.

TODO...

4.1 Study in Keith

Segundo Andrew Sorensen, "*A Study In Keith* é um trabalho para piano solo (NI's Akoustik Piano), inspirado nos concertos *Sun Bear* de Keith Jarrett's" (**??**). Uma nota pertinente sobre esta improvisação: nos primeiros dois minutos do vídeo, existe um silêncio. Este silêncio é decorrente da construção das estruturas lógicas do programa. (Id., **??**)

Este comportamento o tempo de ação decorrido são característicos de outro vídeo analisado na seção 4.3.

4.2 The Disklavier Sessions

Neste trabalho, Sorensen controla um piano. Através da programação com o software *Extempore*, Sorensen controla o mecanismo interno de um piano acústico fabricado pela Yamaha. Esse piano, batizado de *Disklavier*, é peculiar por permitir a memorização (em componentes eletrônicos inseridos no corpo acústico) de uma performance.

TODO...

4.3 Day of the Triffords

TODO ...

5 Conclusão

Referências

AYCOCK, j. A brief history of just-in-time. p. 97–113, 2003. Disponível em: <http://www.cs.tufts.edu/comp/150IPL/papers/aycock03jit.pdf>. Citado 4 vezes nas páginas 14, 26, 30 e 31.

BECKER, H. *Art Worlds*. [S.l.: s.n.], 1982. Citado na página 12.

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 18, 30 e 32.

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 29.

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 30.

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 35, 41 e 42.

COPE, E. D. Prefacio a OM composer's book vol. 2. In: *OM Composer's Book*. [S.l.]: Editions Delatour, 2008. v. 2, p. ix–xv. Citado na página 20.

ENO, B. Base de dados, *Music for Airports liner notes*. 1978. Disponível em: <http://music.hyperreal.org/artists/brian_eno/MFA-txt.html>. Citado na página 39.

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 36.

GRIFFITHS, D. Fluxus: Scheme livecoding. 2008. Disponível em: <http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>. Citado na página 33.

HEIKKILä, V.-M. Discovering novel computer music techniques by exploring the space of short computer programs. p. 8, 2011. Disponível em: <http://arxiv.org/abs/1112.1368>. Citado na página 19.

IAZZETTA, F. *Música e mediação tecnológica*. [S.l.]: Ed. Perspectiva-Fapesp, 2009. ISBN 9.788527308724E12. Citado 2 vezes nas páginas 40 e 41.

Lakatos, I. Falsification and the methodology of scientific research. In: *The Methodology of Scientific Researchs Programmes.* [s.n.], 1970. p. 8–93. Disponível em: <http://strangebeautiful.com/other-texts/lakatos-meth-sci-research-phil-papers-1.pdf>. Citado 2 vezes nas páginas 9 e 21.

MAGNUSSON, T. Herding cats: Observing live coding in the wild. v. 38, n. 1, p. 8–16, 2014. Citado na página 36.

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 na página 37.

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

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

MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 4 vezes nas páginas 9, 26, 27 e 28.

McCallum, L.; SMITH, D. *Show Us Your Screens*. Vimeo, 2011. Disponível em: <https://vimeo.com/20241649>. Citado na página 33.

McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: <https://www.academia.edu/467101/Music_improvisation_and_creative_systems>. Citado 2 vezes nas páginas 9 e 13.

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 18.

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 2 vezes nas páginas 26 e 30.

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 4 vezes nas páginas 14, 26, 28 e 29.

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 2 vezes nas páginas 9 e 21.

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 26.

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 2 vezes nas páginas 26 e 38.

PROSPERO, C. D. Social participation. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 68–73. ISBN 978 0 85316 340 4. Citado na página 12.

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 35.

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 2 vezes nas páginas 37 e 38.

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

ROBERTS, C.; KUCHERA-MORIN, J. *Gibber: live-coding audio in the browser*. [S.l.]: University of California at Santa Barbara: Media Arts & Technology Program, 2012. Citado 2 vezes nas páginas 16 e 31.

ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado na página 31.

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 na página 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 18 e 19.

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 29.

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

Sá, S. P. d. Se você gosta de madonna também vai gostar de britney! ou não? gêneros, gostos e disputa simbólica nos sistemas de recomendação musical. v. 12, n. 2, p. 1808–2599, 2009. Citado na página 13.

TURNER, V. W.; HARRIS, A. *"Liminality and Communitas". The Ritual Process: Structure and Anti-Structure*. [S.l.: s.n.], 1969. Citado na página 12.

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 6 vezes nas páginas 30, 32, 33, 36, 39 e 41.

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 36.

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.