Guilherme Martins Lunhani

Live coding: um comportamento criativo de Andrew Sorensen em *Study in Keith*

29 de dezembro de 2015

Guilherme Martins Lunhani

Live coding: um comportamento criativo de Andrew Sorensen em *Study in Keith*

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

29 de dezembro de 2015

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

Agradecimentos

Ao Inominável.

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

Aos amigxs que estão (ou moraram em Juiz de Fora) e que foram fundamentais (de alguma forma) no caminho: Glerm, Anna Flávia, Tiago Rubini, Aline. Aos amigxs de Campinas e São Paulo que de alguma forma me ajudaram ultrapassar a distâncias, nostalgias e amizades: Celso, Dani, Evandro, Fábio (sivuca), Felício, Frederico (Doshi, Larissa (zé e diva), Rebechi (vivi), Simone (foucault), Tati (Ruan e Ruanzitx).

Aos vizinhos que riem apenas de ver a lua, branca, amarela, ou vermelha, Dhiego e Luisa. Ao Gustavo e Patrick (Pretinha). Aos colegas de mestrado (ou relativos), Diego, Nayse, Analu e Paula. Ao velho amigo que não vejo a muito tempo, Bruno.

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

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

Ao Professor Hans Joachim Koellreutter pelo centenário. Tocar sua peça ajudou a começar a perceber a Via.

Àqueles que passaram pelo caminho, que ajudaram ou atrapalharam. Suas potências permitiram um novo espaço conceitual.

Às arrudas, alecrins, ipês e cebolinhas que enfeitaram o universo de possibilidades.

Alguém poderia imaginar uma interface musical na qual um músico especifica o som resultante desejado, em uma linguagem descritiva, na qual poderia ser então traduzida 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”

Curtis Roads (2001)

Resumo

Este trabalho discute o *live coding*, uma técnica de improvisação audiovisual, como ferramenta para aglutinação de conceitos. O tema é derivado do Modelo de Improvisação de Alex McLean (2006), um dos precurssores do *live coding* em sua forma corrente.

A formalização deste Universo é acompanhada de uma reflexão sobre a multiplicidade de Espaços Conceituais possíveis, seus limites e transições. Um primeiro espaço conceitual é investigado como uma construção histórica das regras heurísticas do *livecoding*. O segundo espaço conceitual é uma algutinação; um estilo de improvisação de *jazz* é apropriado para elaboração da improvisação *Study in Keith* de Andrew Sorensen (2009).

Mais do que a descrição formal para um modelo de improvisação, para esta performance específica, a intenção deste trabalho é construir espaços conceituais para trabalhos posteriores, bem como oferecer uma pequena introdução em língua portuguesa sobre o que é *live coding*.

Palavras-chaves: *Livecoding*. *Study in Keith*. Sistemas criativos. Andrew Sorensen. Keith Jarret.

Sumário

Leia me

O espaço conceitual desta pesquisa se localizou entre os limites da musicologia e de modelos computacionais. Este espaço é vasto e sua localização precisa é ambígua, reduzido a um exemplo, o que acarreta em problemas.

Qualquer critica será bem-vinda. Porém, para organização delas, o autor solicita que seja realizado por meio de um sistema de controle de versões, localizado em <https://www.bitbucket.com/cravista/mestrado/issues>.

Introdução

Na transição do séc. XX para o XXI, DJs e programadores ingleses utilizam linguagens de programação para improvisação musical; posteriormente agregam esta prática para uma improvisação audiovisual.

Collins e McLean (2014) descreve vários momentos para aquilo que ficou conhecido como o lado *algorave* da prática *live coding*; irei pontuar apenas três. No ano de 1997, *Aphex Twin* (Richard David James) utiliza o programa *SuperCollider 1* para produzir um *live club algorithm*; no ano 2000, o duo *Slub* (Adrian Ward, Alex McLean, e poteriormente um trio, com Dave Grifths), utilizam *laptops* para programar uma replicação de estilos como *techno* e *gabba*; em uma boate em Hamburgo, 2004, é fundada uma organização internacional chamada TOPLAP.

Princípios desta organização são publicados e divulgados por Ward et al. (2004), Griffiths (2008), McCallum e Smith (2011) como regras heurísticas do *live coding*, de como lidar com instruções computacionais, improvisadas ou escritas previamente, para uma performance artística; são aceitas como regras bastante razoáveis para um número de praticantes. No entanto, seria possível discutir se tais prescrições são heranças de um período anterior à formalização do *live coding*, período nomeado como *Live Computer Music*:

> A idéia de usar sistemas eletrônicos eles mesmos como atores musicais, oposto a meramente uma ferramenta, iniciou com compositores como David Tudor e Gordon Mumma. Uma continuação natural desses exemplos pode ser encontrada em compositores locais [Baía de São Franscisco] que tocaram com sintetizadores analógicos auto-modificados interligados. Um desess executantes foi o falecido Jim Horton (1994-1998)(BROWN; BISCHOF, 2013, online)[1].

Water Surface (1985), de Ron Kuivila se utiliza da atividade de programação para expor, durante o processo musical, uma falha do sistema FORTH, que finaliza com um sobrecarregamento e um consequente silêncio (TOPLAP, 2007; COLLINS, 2014).

John Bischoff e Tim Perkis (membros do grupo *the Hub*), no final da década de 70, ajustavam uma rede de microcontroladores como parte de um *happening* (BROWN; BISCHOF, 2013).

---

[1] Tradução de *The idea of using the electronic system itself as a musical actor, as opposed to merely a tool, had started with composers like David Tudor and Gordon Mumma. A natural continuation of their example could also be found in the local composers who performed with self-modifying analog synthesizer patches as well. One of these players was the late Jim Horton (1944-1998)*

"Spiral 5 PTL (Perhaps The Last)" (1979), de Dan Sandin, Tom DeFanti, e Mimi Shevitz é atualmente discutido como primeira performance de *live coding*, ao gerar interativamente materiais audiovisuais com a linguagem GRASS[2].

Por outro lado, Mori (2015b) apresenta o compositor Pietro Grossi (1917-2002) como um caso histórico de *live coding* anterior aos casos citados.

Problema

Existem múltiplos contextos de definição do que é, e quando começou o *live coding*. Uma definição amplamente divulgada, através do *software* SuperCollider[3] apresenta aspectos técnicos, mas não históricos:

> *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)[4]

Ward, McLean e Shulgin (2003) definem o *live coding* como uma Música computacional ao vivo com performance visual, controladas por processos algorítmicos[5]. Ward et al. (2004) definem como "atividade da escrita (ou partes de) um programa enquanto ele é executado" [6]. O que concorda com a definição de Sorensen (2014), "programar sistemas de tempo-real durante o tempo de execução"[7]. McLean (2006-07-30), Collins e McLean (2014) discutem o *live coding* pela perspectiva da música eletrônica de entretenimento, ou *algorave*. Magnusson (2011), Collins (2014) sintetizam o *live coding* como improvisação audiovisual.

Por último, Mori (2015a), Prospero (2015) delineiam o estudo do *live coding* como um ramo da etnomusicologia. Os autores discutem a utilização da etnografia (a vivência

[2] Disponível em <http://lurk.org/groups/livecode/messages/topic/5abPazJSxfegYfVFOzN4T6>.

[3] Disponíveis 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.*

[5] Adaptação de *Live computer music and visual performance can now involve interactive control of algorithmic processes.*

[6] Tradução nossa de *Live coding is the activity of writing (parts of) a program while it runs.*

[7] Traduçao adaptada de *programming real-time systems in real-time*. A tradução literal "programar sistemas de tempo-real em tempo-real" nos pareceu pouco explicativa. De fato, esta expressão possui um sentido mais completo na língua inglesa, mas pode confundir o leitor de língua portuguesa. Substituímos o *in real-time* por *during runtime* (durante o tempo de execução, de um *software*) para enfatizar a produção de códigos durante o momento em que está sendo feito. Esta questão pode ser melhor compreendida a partir do GROOVE de Mathews e Moore (1970).

com uma comunidade em observação, os *live coders*), e a revisão histórica, como métodos para a re-discussão dos limites do que é *live coding*.

Hipótese

Neste trabalho definimos o *live coding* como um *Universo de Conceitos*, contendo diversos limites entre seus *Espaços conceituais*, próprios para diferentes *Modelos de Improvisação*. Seus intérpretes programam sistemas criativos durante o tempo de sua execução.

Como exemplo, selecionamos *Study in Keith* de Andrew Sorensen (2009) para análise.

Estrutura dos capítulos

No Capítulo 1 discutimos várias abordagens do *live coding*, seguido por elementos históricos. *Espaços Conceituais*. No Capítulo 2 a formalização destes espaços conceituais é realizada através do Modelo de Improvisação de Alex McLean (2006). No Capítulo 3, *Study in Keith* (2009) de Andrew Sorensen [8] representa um caso particular que envolve, além de música, *replicação do estilo* de improvisação de *jazz* de Keith Jarret, durante os concertos *Sun Bear* (1976-1979), para uma sessão de improvisação com o computador.

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

1 Trabalhos relacionados

Giovanni Mori (2015a, p. 117) sugere que conceitos de práticas diversas, especialmente aqueles desenvolvidos no campo da Música, são agregados durante uma *sessão* de *live coding*: "*Live coding* é uma técnica artística de improvisação. Pode ser empregada em muitos contextos performativos diferentes: dança, música, imagens em movimento e mesmo tecelagem." (MORI, 2015a, p. 117)[1] (ver Figura 1).

Figura 1 – Performance por Alex McLean, David Griffiths, e *Wavecoding group* (2015) utilizando uma máquina manual de tecelagem para realização de uma improvisação audiovisual em Falmouth, Cornwall (<http://fo.am/kernow/>) **Fonte**: <http://www.pawfal.org/dave/blog/category/slub/>.

Estas agregações dependem muito do contexto. Por exemplo, o *live coding* pode ocorrer formalmente ou informalmente. Um exemplo da primeira situação é a performance *screenBashing* (2015) de Magno Caliman (ver Figura 2), realizada durante um concerto em Campinas-SP no ano de 2015.

Informalidades incluem performances conhecidas como *algoraves*. Apresentamos um exemplo do duo mexicano Mico Rex, em um circuito que percorreu as cidades de Brighton, Londres, Karlesruhe, Colonia e Dusseldorf (2013), evento permeado de música eletrônica de pista produzida pela programação de algoritmos (ver Figura 3). Collins e

[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.*

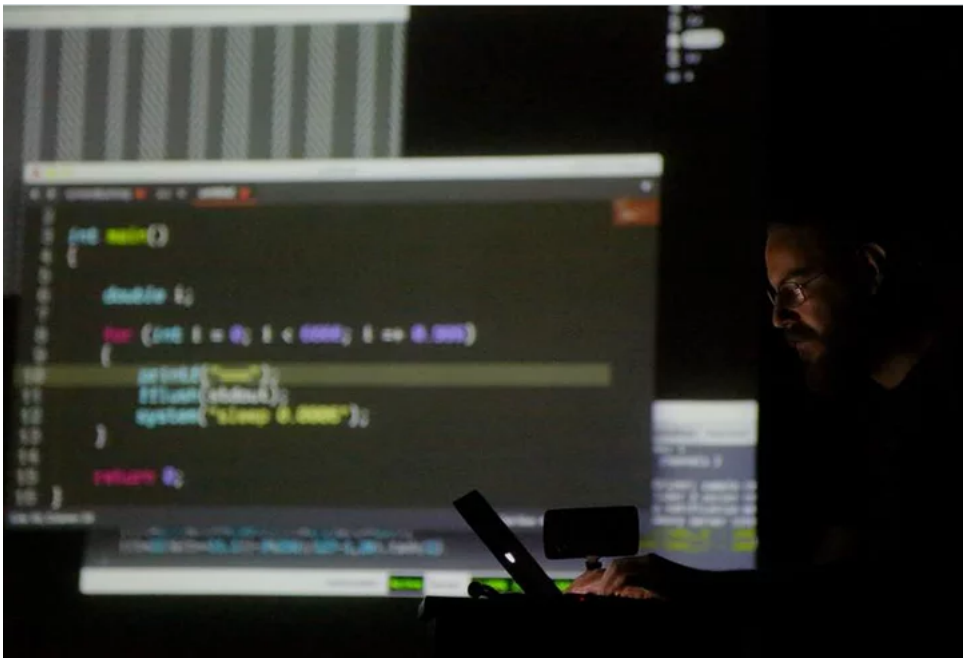


Figura 2 – Performance de *screenBashing* durante o XIII ENCUN 2015 - Campinas (SP). A performance é um exemplo claro da estética *noise* dentro do *live coding*. Utiliza a linguagem C para gerar saídas textuais, utilizadas como texturas visuais, e o programa *SuperCollider* para síntese sonora. **Fonte**: <https://vimeo.com/148626379>.

McLean (2014, p. 356) descreve o *algorave* como uma atividade de *code DJing*, ou um "Disk Jockey codificado":

> Executantes de *algorave* apresentam um campo eclético de musicistas eletrônicos, usando predominantemente um laptop, mas também incluindo alguns experimentos no controle de hardware (...) Por exemplo, *sick lincoln* simultaneamente combinou DJ codificado no *SuperCollider*, *live coding* em um aplicativo *Web Audio API/javascript* e refabricação de *patchs* de um sistema algorítmico no Max/MSP.[2]

O campo de atuação do *live coding* é dinâmico, e depende de sua aplicação. Expomos três limites da prática, mas certamente existem outras abordagens. Enumerar todas pode ofuscar aquilo que queremos chamar a atenção: a multiplicidade de conceitos, seus limites e transições.

Uma primeira exposição desses diversos conceitos pode ser visualizada na Figura 4; a figura foi feita com o auxílio de um programa descrito no Apêndice A, e melhor explorada no Apêndice B.

[2] Tradução nossa de *Algorave performers present an eclectic range of electronic musicians, predominantly using laptop alone, but also including some experiments in control of hardware (...) For example,* sick lincoln *has simultaneously combined code DJing from SuperCollider, live coding from a Web Audio API javascript app and live repatching of a Max/MSP algorithmic hip hop system.* Sobre *Web Audio API* e Javascript, ver Roberts e Kuchera-Morin (2012). Javascript é uma linguagem de programação utilizada para realização de rotinas em páginas *web*. Sobre Max/MSP, <>

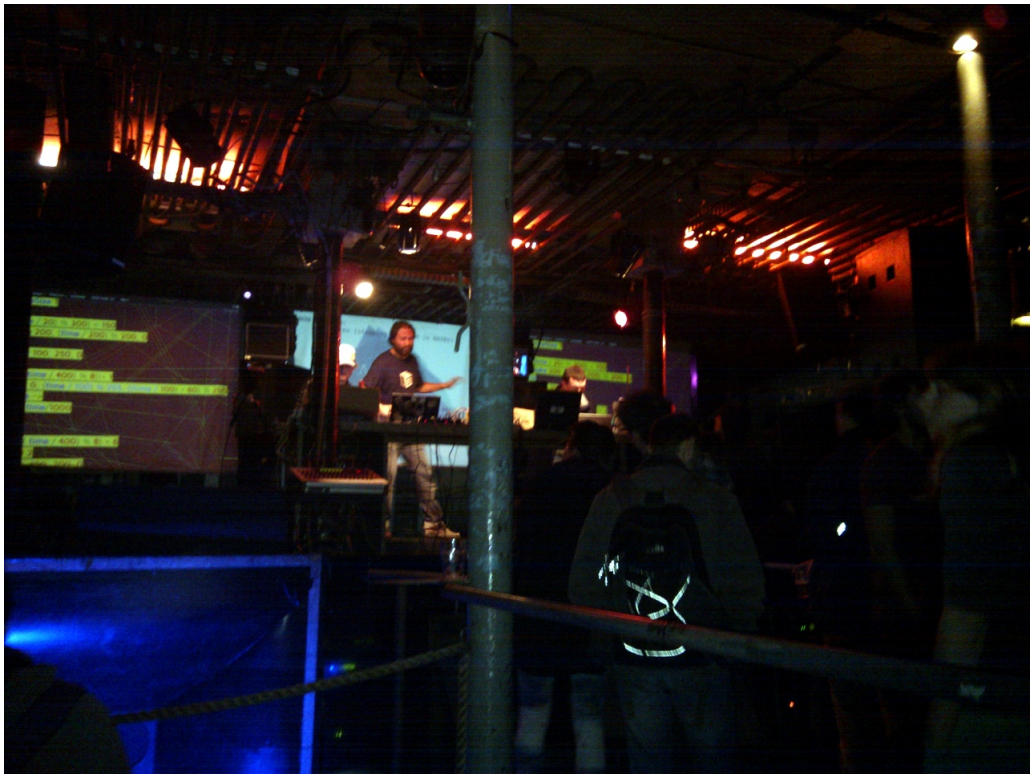


Figura 3 – Performance de Mico Rex durante o *Algorave tour* (2013) **Fonte**: <http://www.pawfal.org/dave/blog/2013/04/life-on-an-algorave-tour/>.

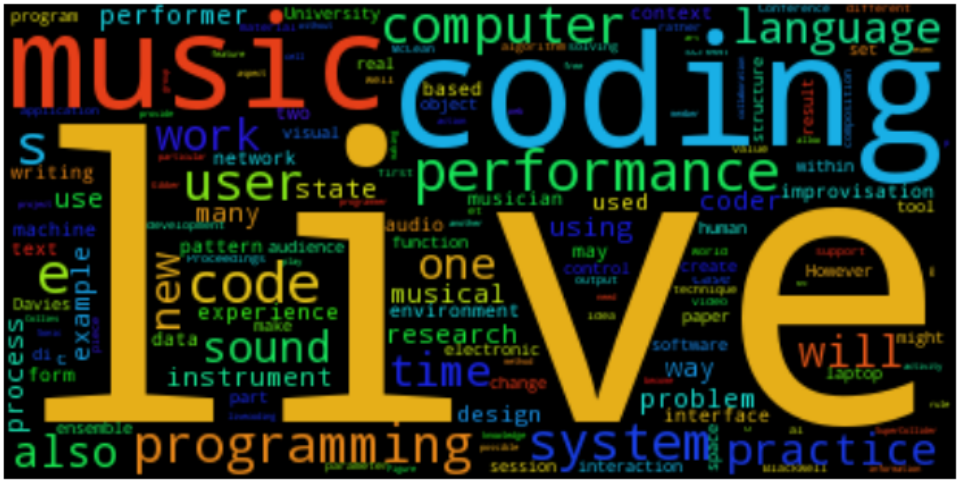


Figura 4 – Nuvem de palavras do McLean et al. (2015), 1º Congresso Internacional de Live Coding. **Fonte**: autor.

1.1 Improvisação e Sistemas Criativos

Qualquer que seja a categorização dada ao *live coding*, o termo *live* transparece um assunto articulador: *improvisação*. Definições do *Universo de Conceitos para McLean*, e o *Modelo de Improvisação* de Pressing, serão discutidos na Capítulo 2 como o método de investigação realizado.

1.2 *live coding*

Na subseção 1.2.1, descrevemos 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. Não foi usado para ambientes de performance, mas a peça *The expanding universe* da compositora Laurie Spiegel (1975) foi considerada.

Na subseção 1.2.2, 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.

Descrevemos, na seção subseção 1.2.3, as atividades dos grupos *The Hub* e *The League of Automatic Composers* como fundamentais para o entendimento histórico do *live coding*.

Sugerimos na subseção 1.2.2.1 apontar a tecnologia JIT (AYCOCK, 2003) como um sujeito sócio-técnico fundamental para que o *live coding* fosse possível.

Uma revisão de um trecho da publicação "*Live Algorithm Programming and Temporary Organization for its Promotion*", de McLean e Wiggins, será feita na subseção 1.2.5 para discutir identidade cultural da organização TOPLAP.

Na subseção 1.2.6, "Show us your screens", é revisto como o manifesto que define as regras heurísticas do *live coding*.

1.2.1 GROOVE

GROOVE, ou *Generated Real-time Operations On Voltage-controlled Equipment* foi um computador desenvolvido na Bell Labs por (MATHEWS; MOORE, 1970). Alex Nunzio (2010) discute como um precedente direto do MUSIC V, mas como um projeto bifurcado da família MUSIC N[3].

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*,

[3] Desenvolvidos a partir de 1957. *softwares* MUSIC I, II, III, IV, IV-B, IV-BF, V (que passou por modificações no IRCAM), MUSIC 360, MUSIC 11

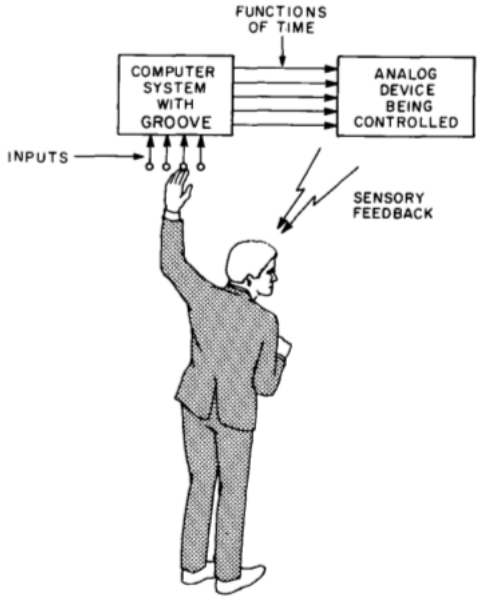


FIG. 1. Feedback loop for composing functions of time

Figura 5 – Esquema de concepção do projeto GROOVE descrito no artigo homônimo por Max Mathews **Fonte**: (MATHEWS; MOORE, 1970).

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:

> 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 5]. Assim ele é capas de modificar as funções instantâneamente como um resultado de suas observações daqueles efeitos.(MATHEWS; MOORE, 1970, p. 715) [4]

O terceiro conceito observa a existência de uma relação entre um humano e uma máquina. Mathews descreve-o como uma *engenharia humana*. Esta engenharia consistiu na observação de um tempo diferencial entre o que o(a) musicista cria e o que edita:

> 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. (...) O intérprete de computador não deve tentar definir todo o som em tempo real. Ao invés, o computador deve ter uma partitura e o intérprete deve influenciar a forma como a

[4] 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.*

> partitura é tocada. Seus modos de influência podem ser mais variados do que aqueles que um regente convencional, que pode principalmente controloar o tempo, intensidade, e estilo.(MATHEWS; MOORE, 1970, p. 715-716) [5]

Como exemplo , selecionamos uma descrição da compositora Laurie Spiegel (1975) (ver Figura 6) para sumarizar as características do GROOVE, durante a produção de *The Expanding Universe* [6], entre as salas 2D-506 da Bell Labs (contendo o computador DDP-224) e a sala analógica 2D-562 (laboratório de Mathews). A "performance" da obra era realizada, com a programação de funções temporais e a manipulação de parâmetros dessas funções através de dispositivos físicos:

> Todas as músicas no GROOVE eram representadas na memória digital como funções abstratas do tempo, séries paralelas de dois pontos, cada ponto sendo um instante no tempo e um valor instantâneo. A taxa de amostragem para essas funções, usada principalmente como controle de voltagem, era cronometrada por um grande e antiquado oscilador analógico que era normalmente fixado em 100 Hertz, cada ciclo do oscilador pulsando à frente do código, o computador lia, em cada uma das funções, naquele ponto do tempo, todos dispositivos de entrada e executava todas amostras. (...) Tínhamos uma pequena caixa com 4 potenciômetros e quatro chaves (alternadores fixados onde você os colocava) e dois botões de disparo.[7]

Embora não declare ser uma peça minimalista, a descrição de *The Expanding Universe* considera, de maneira asséptica, os fenômenos psicoacústicos como elementos composicionais. Por exemplo, a utilização da continuidade progressiva de sons (ou *drones* transitórios) como elemento criativo permite, segundo a compositora, à sensibilização do ouvido, o que não seria possível na música minimalista instrumental:

> A violência da perturbação sonora, disjunção, descontinuidade e mudanças súbitas desanitizam o ouvinte e nos afastam, de forma que não estamos mais abertos aos sons mais sutis. Mas com continuidade e gentileza, o ouvido se torna re-sensibilizado para mais e mais fenômenos auditoriais sutis dentro do som que estamos imersos. Em vez de sermos

[5] 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. (...) 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.*.

[6] Disponível em <https://www.youtube.com/watch?v=dYUZmsfm4Ww>.

[7] Tradução de *All music in GROOVE was represented in digital memory as abstract functions of time, parallel series of point pairs, each point being an instant in time and an instantaneous value. The sampling rate for these functions, which would be used mostly as control voltages, was clocked by a big old-fashioned analog oscillator that was usually set to 100 Hertz, each cycle of the oscillator pulsing one run through the code, the computer reading all of the real time input devices and playing of all of the samples at that time point in each of the time functions. (...) We had a small box with 4 knobs, 4 set switches (toggles that stay where you put them) and 2 momentary-contact push buttons on it.*

Figura 6 – Laurie Spiegel configurando a saída analógica do GROOVE, durante a produção de *The Expanding Universe*. **Fonte**: (SPIEGEL, 1975).

> arrastados, como nas cascatas de muitas notas executadas em blocos de tempo que mudam repentinamente, tal como tantas vezes consite a música "minimalista", abrimos nossos ouvidos mais e mais para os mais fenômenos que nos envolvem. Isto também não é música ambiente, um termo que veio a ser usado alguns anos depois. Esta é música para atenção concentrada, uma experiência musical do através, pensando que, lógico, existe também um pano de fundo. [8]

Nesta citação podemos sumarizar um conceito para o *live coding*: processo. Porém, o significado de processo pode ser desenvolvido infinitamente. Isso não será realizado. O significado de processo para Spiegel é diverso daquele considerado no *live coding*, e uma digressão desta pode afastar demais o foco do trabalho principal. Para compreender o significado de processo no *live coding*, será necessário continuar.

1.2.2 Pietro Grossi

Embora pouco conhecido no contexto geral da música européia, o compositor Pietro Grossi foi um dos pioneiros da *Computer Music* Italiana. O pensamento musical

[8] Tradução de *The violence of sonic disruption, disjunction, discontinuity and sudden change desensitizes the listener and pushes us away so we are no longer open to the subtlest sounds. But with continuity and gentleness, the ear becomes increasingly re-sensitized to more and more subtle auditory phenomena within the sound that immerses us. Instead of being swept along, as with cascades of many running notes in suddenly-changing blocks of time, such as "minimalist" music so often consists of, we open up our ears more and more to the more minute phenomena that envelope us. This is also not "ambient music", a term that came into use some years later. This is music for concentrated attention, a through-composed musical experience, though of course it also can be background.*

que rege seus programas de computador sacrifica questões timbrísticas para concentrar na performance. 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) [9]

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."[10]. Neste sentido, defendia que o desenvolvimento de novos timbres deveria esperar por melhores implementações de *hardware*.

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"[11], uma execução ultra-virtuosística dos *Capricci op.1* de Niccolò Paganini; foi executada no computador Olivetti GE-115 Mori (2015b, p. 126).

Sumarizamos o seguinte conceito: *reflexividade*, ou a "habilidade de um de um programa manipular como dados algo que representa o estado do programa durante

[9] 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.*

[10] 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.*

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

sua própria execução, o mecanismo para codificação de estados de execução é chamado *reificação.*(MALENFANT; JACQUES; DEMERS, 1996, p. 1)" [12]

1.2.2.1 Just In Time (JIT)

A reflexividade é uma característica de diversos ambientes de *live coding*. O *SuperCollider* foi o primeiro dos ambientes de programação musical a implementar esta caracterísitca, a partir da tecnologia *Just In Time Compilation* (Compilação JIT)

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).

Atualmente, esta técnica têm sido largamente implementada para navegadores de internet (ROBERTS; WAKEFIELD; WRIGHT, 2013). Programas como Gibber[13] (ROBERTS; KUCHERA-MORIN, 2012) e *wavepot*[14] são exemplares; permitem a agregação do conceito de *reflexividade* ao conceito *redes*, que será apresentado na próxima seção.

1.2.3 Baía de São Franscisco

> Com o florescimento da indústria de computadores pessoais na Baía de São Franscisco, o acesso às novas tecnologias e pessoas que desenvolveram elas era talvez o melhor no mundo. Mas se para todos os jovens com fortunas como panos para suas mentes (e seus futuros) que perseguiam um excitamento aditivo na construção de máquinas eletrônicas, também existiam políticos utópicos que sonhavam com uma nova sociedade construída no livre e aberto acesso à informação, e na abrangente tecnologia baseada em sistemas inteligentes. Esta também é a cultura que deu ao mundo a música "New Age", uma versão aguada e comercializada das músicas com base em modos e drones que Terry Riley, Pauline Oliveros, e LaMonte Young inventaram durante os anos cinquenta e sessenta. Mas a música da Costa Oeste também incluía livre-restrição, barulho, e improvisações com bordas que sobraram das revoluções contra-culturais dos anos 60(BROWN; BISCHOF, 2013, online)[15].

[12] Tradução nossa de *the ability of a program to manipulate as data something representing the state of the program during its own execution, the mechanism for encoding execution states as data being called reification.*.

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

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

[15] Tradução de *With the flowering personal computer industry in the Bay Area, access to the new digital technologies and to the people who developed them was perhaps the best in the world. But for all the young men with fortunes in the back of their minds (and in their futures) who pursued the addictive excitement of building electronic machines, there were also the political utopians whose dream was of a new society built on the free and open access to information, and on a comprehensively designed technology based on embedded intelligence. This was also the culture that gave the world "New Age"music, a watered-down and commercialized version of the musics based on modes and drones that Terry Riley, Pauline Oliveros, and LaMonte Young invented here during the late fifties and early sixties. But West Coast music-making also included a free-wheeling, noisy, improvisational edge left over from the counter-cultural revolutions of the sixties.*

Na segunda metade da década de setenta, Jim Horton começou a adquirir microcontroladores KIM-1[16]. Segundo Brown e Bischof, não demorou para que outros interessados comprassem. Discussões informais posteriores, que incluiam, além de Horton, David (Behrman), John Bischoff, Tim Perkis, Rich Gold, Cathy Morton, Paul Robinson, e Paul Kalbach, sugeria a formação de uma "orquestra de silício" (*silicon orchestra*).

Ademais, em 1977, Horton colaborou com duas peças que interligavam estes microcontroladores. A primeira era construída sobre algoritmos inspirados nas teorias matemáticas de Leonard Euler (séc. XVIII). A segunda peça também explorava a comunicação entre os microcontroladores, de forma que "notas ocasionais da minha (Bischof) máquina faziam a máquina de Jim transpor atividades melódicas de acordo com minha nota base(BROWN; BISCHOF, 2013, online)" [17].

Em 1978, Bischof, Behrman, Gold e Horton gravaram um *Extended Play* (EP)[18] a partir de uma performance em 26 de Novembro de 1978, lançado pela pela Lovely Music (NY) em 1980 como *The Hub: Computer Network Music*. Durante este tempo, foi formado o grupo *"The League of Automatic Music Composers"*, que além de Bischof, Perkis, Brown, contava com Scot Gresham-Lancaster, Mark Trayle e Phil Stone. Aquele era um momento onde os *happenings* já eram manifestações artísticas consolidadas. Não demorou muito para que o público participasse da atividade:

> 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)[19]

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

[17] Tradução nossa de *the occasional tones of my machine caused Jim's machine to transpose its melodic activity according to my "key"note.* .

[18] Gravação muito longa para um *demo* e insuficiente para um disco, de vinil na época.

[19] 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 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*

Nesta seção sumarizamos os conceito *rede de composições*. Este conceito pode ser melhor compreendido através de uma descrição do processo criativo da banda:

> Os membros da liga geralmente adaptavam composições solo para usar dentro da banda. Estes solos eram desenvolvidos independentemente por cada compositor, e eram tipicamente baseados em esquemas de algoritmos de um tipo ou outro. Existiam características de improvisação diferentes para muitas delas, como bem as músicas eram diferentes em detalhes. Teorias matemáticas sistemas de afinação experimentais, algoritmos de inteligência artificial, projetos de instrumentos de improvisação, e performance interativa eram algumas das áreas exploradas nestes trabalhos (. . .) Os solos tocavam simultaneamente no cenário de grupo, se tornando "sub"-composições que interagem, cada uma enviando e recebendo dados pertinentes para o funcionamento musical(BROWN; BISCHOF, 2013, online)[20].

1.2.4 Ron Kuivila

Segundo McLean e Wiggins (2009), Ron Kuivila realiza a primeiras performance de *live coding* em 1985 com a performance *Water Surfaces* na STEIM[21], em Amsterdã (BLACKWELL; COLLINS, 2005).

O disco "*TOPLAP001 - A prehistory of live coding*" traz uma reconstrução da peça realizada em 2007 [22]; uma nota sobre a performance descreve o seguinte: "Esta obra usou programação FORTH ao vivo; Curtis Roads testemunhou e relatou a performance de Ron Kuivila feita na STEIM em Amsterdã, em 1985; a performance original termina com a quebra do sistema." [23,24].

Ge Wang (2005), em uma comunicação pessoal com Curtis Roads, cita a seguinte declaração: "Eu vi o *software* FORTH de Ron Kuivila quebrar e queimar no palco em Amsterdã em 1985, mas antes disso, não fez uma música muito interessante. A performance consistiu de digitação" [25]

---

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.

[20] Tradução de *League members generally adapted solo compositions for use within the band. These solos were developed independently by each composer and were typically based on algorithmic schemes of one kind or another. There was a distinctly improvisational character to many of these as the music was always different in its detail. Mathematical theories of melody, experimental tuning systems, artificial intelligence algorithms, improvisational instrument design, and interactive performance were a few of the areas explored in these solo works. (. . .) The solos, played simultaneously in the group setting, became interacting "sub-compositions, each sending and receiving data pertinent to its musical functioning.*

[21] *STudio for Electro-Instrumental Music*, disponível em <http://steim.org/about/>.

[22] Disponível em <http://toplap.org/wiki/TOPLAP_CDs>.

[23] Tradução nossa de *This work used live FORTH programming; Curtis Roads witnessed and reported a performance by Ron Kuivila at STEIM in 1985; the original performance apparently closed with a system crash. . . .*

[24] FORTH é uma linguagem de programação elaborada por Charles Moore (1938-). Entre seus paradigmas de programação, utiliza da *reflexividade*.

[25] Tradução nossa de *I saw Ron Kuivila's Forth software crash and burn onstage in Amsterdam in 1985, but not before making some quite interesting music. The performance consisted of typing.*.

Nenhuma fonte sonora foi encontrada disponível online. Porém este conceito é revisitado por Magno Caliman em *scriptBashing* (2015): a recodificação de um programa escrito em linguagem C cria centenas (na ordem de 800) processos paralelos só no plano visual. Outros processos, com a criação de sintetizadores no SuperCollider, embora parcialmente visíveis, são adicionados na pilha de memória. A performance acaba com um silêncio decorrente da saturação da memória RAM.

1.2.5 Live Algorithm Programming and Temporary Organization for its Promotion

Este acrônimo deriva do manifesto "*Live Algorithm Programming and Temporary Organization for its Promotion*", LAPTOP, (WARD et al., 2004; BLACKWELL; COLLINS, 2005).

No *Wiki* do site oficial[26], a cada visita, a ordem das letras são permutadas, criando diferentes significados. Por exemplo, *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.

Este comportamento, de permutar ordem das letras, de algum título, ou até mesmo do próprio nome para gerar pseudônimos, já era praticada por Click Nilson (Nick Collins), que define bem o *live coding*: "Live coding celebra a efemeralidade de sua própria definição" [27]

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 a Música de processos:

> 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

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

[27] Tradução nossa de *Live coding celebrates the ephemerality of definition itself. Disponível em <http://lurk.org/groups/livecode/messages/topic/ofAxZpxsKFpDRLnoA48Bh>.*.

> a tomada de decisão e tentar obter um melhor acesso ao processo de pensamento do compositor. (WARD et al., 2004, p. 245) [28]

O trecho acima fornece informações a respeito do espaços conceitual do *algorave*, ou das pŕaticas de *Disk Jockey* com a utilização da a atividade de programação em linguagem de computador. Adiante podemos ver outros dois conceitos aglutinados: Música de Processos, ou Música de algoritmos simples, e Música Generativa [29]:

> 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". [30]

1.2.5.1 *Algoraves*

Algorave é um tipo de música eletronica de pista, que se utiliza de algoritmos, processos, e teorias generativas; seus processos criativos replicam alguns regimes de escuta, tais como *dance*, *drum'n'bass*, *cyberpunk*. No entanto, *algorave* é um termo anterior ao advento do *live coding*:

> *Algorave* não é sustentado exclusivamente por *live coders*, mas estes têm mantido uma forte presença em todos os eventos até agora. É assim talvez,

[28] 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.*

[29] Para mais informações, ver Reich (1968) e Mailman (2013, p. 128). Sobre Música generativa, Eno (1996)

[30] 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".*

> 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) [31]

Collins e McLean (2014) apresentam dados a respeito da história da *algorave*, de 1992 a 2004. Em 1992, Charles Ames disponibiliza o *Cybernetic Composer*, "um *software* com um sistema baseado em Inteligência Artificial que compõe música em uma variedade de estilos populares." [32,33]. Em 1994, o duo *Koan*, formado pelos DJs Daniel Roeth e William Grey, realizam adaptações para entretenimento com base no *ambient music* de Brian Eno (1978). *Aphex Twin* (Richard David James) reinvindica em 1997 o termo *live club algorithm*. Em 1999, o protocolo para edição audiovisual ao vivo *bbcut* (COLLINS; OLOFSSON, 2003) é incluído nos *opcodes* do *CSound*[34], e do *Supercollider*. Em 2000 o *Slub*, realizam performances, autodenominadas *generative techno*, com abordagem *gabba*. 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. [35]

De maneira sumarizada, escolhemos os conceitos: *música de processos, música generativa, Inteligência Artifical, e música de pista* como conceitos relacionados ao manifesto "*Live Algorithm Programming and Temporary Organization for its Promotion*".

1.2.6 *Show us your screens*

Além das performances inaugurais nos festivais Europeus, e do manifesto "Live Algorithm Programming and Temporary Organization for its Promotion", "*Show Us Your Screens*" (GRIFFITHS, 2008; McCallum; SMITH, 2011, p. 22; online), contêm as regras heurísticas do *live coding*:

> 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.

[31] 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.*

[32] Tradução nossa de *an AI based software system that composes music in a variety of popular styles.*.

[33] Disponível em <http://www.kurzweilai.net/charles-ames>

[34] Disponível em <https://csound.github.io/>.

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

> • 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. [36]

"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*[37] *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

[36] 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.*

[37] 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.

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)[38] 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)[39]

"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

[38] 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.*

[39] 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.*

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 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)[40]

[40] 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.*

2 Metodologia

Carolina Di Prospero (2015) descreve uma dinâmica das pluralidades conceituais do *live coding* através dos estudos sobre *liminaridade* de Turner e Harris (1969a). Tais pluralidades são resultantes de períodos onde entidades (pessoas ou instituições) marginais produzem artefatos de conhecimento que podem gerar mudanças estruturais em um espaço de convivência:

> Liminaridade, marginalidade e inferioridade estrutural são condições no qual são frequentemente gerados mitos, rituais, símbolos sistemas filosóficos e obras de arte. Estas formas culturais provêm aos homens um conjunto de modelos que são, em um nível, reclassificações periódicas da realidade e da relação do homem com a sociedade, natureza e cultura. Mas, eles são mais do que classificações, uma vez que incitam à ação bem como ao pensamento. Cada uma dessas produções tem um caractere multivocal, tendo muitos significados, e cada um capaz de mover as pessoas. (TURNER; HARRIS, 1969a apud PROSPERO, 2015, p. 71)[1]

Liminaridade é discutida por Turner e Harris (1969b, p. 96) através da coexistência, justaposta e alternada, de duas diferentes estruturas de relação humana:

> É como pensar que existem dois grandes "modelos"para a inter-relação humana, justapostas e alternadas. O primeiro é da sociedade como um sistema estruturado e diferenciado, e muitas vezes [como] sistema hierárquico de posições político-jurídico-económicas com muitos tipos de avaliação, que separa os homens em termos de "mais"ou "menos". O segundo, que emerge reconhecidamente no período liminar, é da sociedade como um não-estruturado, ou comitato rudimentarmente estruturado e relativamente indiferenciado, a comunidade, ou mesmo comunhão de indivíduos iguais que juntos se submetem à autoridade geral dos anciãos rituais.[2]

Di prospero esclarece o que seria essa liminaridade no *live coding*:

> Liminaridade é apresentada de algumas formas no *live coding*: novos projetos e propostas, a procura por desmistificar a relação com a tecnologia,

[1] Liminality, marginality, and structural inferiority are conditions in which are frequently generated myths, rituals, symbols, philosophical systems and works of art. These cultural forms providemen with a set of templates or models which are, at one level, periodical reclassifications of reality and man's relationship to society, nature and culture. But, they are more than classifications, since they incite to action as well as to thought. Each of these productions has a multivocal character, having many meanings, and each is capable of moving people

[2] Tradução de *It is as though there are here two major"models"for human interrelatedness,juxtaposed and alternating. The first is of society as a structured, differentiated, and often hierarchical system of politico-legal-economic positions with many types of evaluation, separating men in terms of"more"or "less."The second, which emerges recognizably in the liminal period, is of society as an unstructured or rudimentarily structured and relatively undifferentiated comitatus, community, or even communion of equal individuals who submit together to the general authority of the ritual elders.*

> 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 (*Ibdem*, *idem*). [3]

Com base nisso, propomos que a investigação sobre uma liminaridade do *live coding* seja construída sobre a noção de *improvisação*, *criatividade* e *espaços conceituais*, no contexto de improvisação de Andrew Sorensen. Para isso, recorremos ao artigo "*Music improvisation and creative systems*" de Alex McLean (2006).

Este artigo discute a improvisação musical com base em outros dois artigos, "*A preliminary framework for description, analysis and comparison of creative systems*", de Wiggins (2006), e "*Improvisation: methods and models*" de Jeff Pressing (1987). McLean propõe "traduzir o Modelo de Improvisação para funcionar em um modelo computacional" (McLean, 2006, p. 5)[4] uma adaptação entre a *Ferramenta de Estruturação de Sistemas Criativos*[5] de Wiggins, e o *Modelo de Improvisação* de Pressing.

2.1 Criatividade

> Por definição, criatividade cria, i.e., produz alguma coisa nova. Mas se estamos comprometidos com uma abordagem mecanicista do mundo – nenhum milagre é permitido – iremos acreditar que tudo o que ocorre é, em princípio, previsível. Iremos acreditar também que qualquer coisa nova deve ser construída de componentes existentes. Isso implica que nada pode ser intrinsicamente novo. (Thornton, 2007, p. 2)[6]

Ao discutirem o paradoxo de Boden, Wiggins (2006, p. 450) e Thornton concordam que o ato criativo é sempre mediado por desenvolvimentos conceituais (ver Tabela 1)

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

[4] Tradução parcial de *The above analysis highlights a number of areas for consideration while translating the IM to a working computer model.*

[5] *Creative System Framework.*

[6] Tradução de *By definition, creativity creates, i.e., it produces something new. But if we are committed to a mechanistic account of the world — no miracles allowed — we believe that everything that occurs is predictable in principle. We also believe that any new thing must be constructed from existing components. This implies that nothing can ever be intrinsically new.*

Tabela 1 – Definições formais do Universo de criatividade para fins quantitativos.

Autor	Interpretação da criatividade
Wiggins	"Boden concebe o processo de criatividade como uma identificação e/ou localização de novos objetos conceituais em um espaço conceitual" [7].
Thornton	"Qualquer ato criativo é fundado na conceitualização ou realização de um ponto dentro de um espaço conceitual particular" [8]

Podem ocorrer dois tipos de de conceitualização: a que identifica novos pontos no espaço conceitual (comportamento explorador), e a que gera novos pontos (comportamento transformador). Por outro lado, existem os conceitos do tipo pessoais (P-conceitos), elaborados por uma pessoa através da exploração, e os comportamentos do tipo históricos (H-conceitos), que envolvem o comportamento transformacional. Wiggins problematiza esta abordagem ao tocar a esfera social. Segundo o autor, "pode ser possível, por exemplo, um comportamento ser P-criativo em uma sociedade, mas H-criativo em outra" (*Ibdem*, *idem*). [9].

Também existe uma separação entre as "idéias novas" e aquilo que seria "genuinamente criativo". Idéias novas envolvem os P-Conceitos e comportamentos exploradores. O que não quer dizer que, sejam desconsiderados como criativos: "Processos exploradores são considerados criativos se eles forem guiados de alguma forma por uma "heurística" ou "mapas".Thornton (2007, p. 3)" [10] Por outro lado, "a criatividade genuína" ocorre através de regras delineadoras da transformação conceitual: "Uma mera idéia nova é uma que pode ser descrita e/ou produzida pelo mesmo conjunto de regras generativas como outras idéias familiares. Uma genuína, original ou idéia criativa é uma que não pode (Thornton, 2007 apud BODEN, 1990, p. 3;p. 40)" [11].

Para lidar formalmente com o tema "criatividade" e seus Espaços Conceituais, Wiggins realiza uma categorização de comportamentos critaivos em humanos, e em sistemas computacionais (ver Tabela 2):

[7] Tradução nossa de *Boden conceives the process of creativity as the identification and/or location of new conceptual objects in a conceptual space.*.

[8] Tradução nossa de *Any creative act is thus founded on conceptualisation or the realisation of a point within a particular 'conceptual space'.*

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

[10] Tradução nossa de *Exploratory processes are only to be considered creative if they are guided in some way by 'heuristics' or 'maps'.*.

[11] Tradução nossa de *A merely novel idea is one which can be described and/or produced by the same set of generative rules as are other, familiar ideas. A genuinely original, or creative, idea is one which cannot.*.

[12] Tradução de *The performance of tasks which, if performed by a human, would be deemed creative.*.

[13] Tradução de *The study and support, through computational means and methods, of behaviour exhibited by natural and artificial systems, which would be deemed creative if exhibited by humans.*.

[14] Tradução de *One or more of the behaviours exhibited by a creative system.*

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

Criatividade	"A performance de tarefas que, quando executados por um humano, são consideradas criativas" [12]
Computação criativa	"O estudo e suporte, através de meios e métodos computacionais, do comportamento exibido por sistemas naturais e artificiais, que são considerados criativos". [13]
Sistemas criativos	"Uma coleção de processos, naturais ou automáticos, que são capazes de alcançarem ou simularem comportamentos que em humanos seria considerado criativo"
Comportamento Criativo	"Um ou mais dos comportamentos exibidos por um sistema criativo"[14]

2.2 Ferramenta de Estruturação de Sistemas Criativos

> O universo, $\mathcal{U}$, é um espaço multidimensional, no qual dimensões são capazes de representar qualquer coisa, e todos os possíveis conceitos distintos correspondentes àqueles pontos em $\mathcal{U}$ (...) Para tornar a proposta um espaço-tipo possível, permitirei que $\mathcal{U}$ contenha todos os conceitos abstratos, bem como os concretos, e que é possível representar os artefatos tanto completos e incompletos (WIGGINS, 2006, p. 451).[15]

Wiggins esclarece que Boden não reconhece de forma explícita $\mathcal{U}$, "ela borra a distinção entre as regras que determinam a adesão do espaço (...) e outras disposições que possam permitir a construção e/ou detecção de um conceito representado por um ponto no espaço" (*Idem, ibdem*).

Espaços conceituais $\mathcal{C}$, finitos ou infinitos são definidos como restrições de um universo $\mathcal{U}$, caracterizando um conjunto não-determinístico de conhecimentos: "A noção-chave na teoria de Boden é aquele do espaço conceitual. Enquanto nenhuma definição formal é provida, é comum interpretar esta frase literalmente, tomando o espaço conceitual sendo um espaço de conceitualizações, ou representações de conceitos (Thornton, 2007, p .7)." [16]

McLean (2006) ainda descreve regras que validam concepções diferentes entre espaços conceituais $\mathcal{C}$ diversos em um Universo de Conceitos $\mathcal{U}$ (ver Tabela 3).

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

[16] Tradução nossa de *The key notion in Boden's theory is that of the conceptual space. While no formal definition has been provided, it is common to interpret the phrase literally, taking the conceptual space to be a space of conceptualisations or concept representations.*.

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

Representação	Nome	Significado
c	Conceito	Uma instância de um conceito, abstrato ou concreto (WIGGINS, 2006).
$\mathcal{U}$	Universo de Conceitos	Superconjunto não restrito de conceitos. (WIGGINS, 2006). "Um universo de todos conceitos possíveis" (McLean, 2006) [17]
$\mathcal{L}$	Linguagem	Linguagem utilizada para expressar regras.
$\mathcal{A}$	Alfabeto	Alfabeto da linguagen que contêm caracteres apropriadospara expressão das regras
$\mathcal{R}$	Regras de validação	Validam os conceitos em um universo, se apropriados ou não para o espaço trabalhado.
$[[.]]$	Função de interpretação	"Uma função parcial de $\mathcal{L}$ para funções que resultam em números reais entre [0, 1] (...) 0.5 [ou maior] significa uma verdade booleana e menos que 0.5 siginifica uma falsidade booleana; a necessidade disso para valores reais se tornará clara abaixo" (WIGGINS, 2006, p. 452)[18]
$[[\mathcal{R}]]$	Regras de validação	"Uma função que interpreta $\mathcal{R}$, resultando em uma função indicando aderência ao conceito em $\mathcal{R}$"[19]
$\mathcal{C} = [[\mathcal{R}]](\mathcal{U})$	Espaço Conceitual	"Todos espaços conceituais são um subconjunto não-estrito de $\mathcal{U}$"[20]. Um subconjunto contido em $\mathcal{U}$ (WIGGINS, 2006). Uma função que interpreta $\mathcal{R}$, resultando em uma função que indica aderência ao conceito em $\mathcal{R}$ [21]
$\mathcal{T}$	Regras de detecção	"Regras definidas dentro de $\mathcal{L}$ para definir estratégias transversais para localizar conceitos dentro de $\mathcal{U}$" (McLean, 2006)[22]
$\mathcal{E}$	Regras de qualidade	"(...) conjunto de regras que permitem-nos avaliar qualquer conceito que nós encontramos em $\mathcal{C}$ e determinar sua qualidade, de acordo com critérios que nós considerarmos apropriados" (WIGGINS, 2006, p.453)[23]"Regras definidas dentro de $\mathcal{L}$ para avaliar a qualidade ou a desejabilidade do conceito c" (McLean, 2006)[24]
$<<< \mathcal{R}, \mathcal{T}, \mathcal{E} >>>$	Função de interpretação	Uma regra necessária para definir o espaço conceitual, "independentemente da ordem, mas também, ficcionalmente, enumerá-los em uma ordem particular, sob o controle de $\mathcal{T}$ – isto é cricial para a simulação de um comportamento criativo de um $\mathcal{T}$ particular (WIGGINS, 2006) [25]. "Uma função que interpreta a estratégia transversal $\mathcal{T}$, informada por $\mathcal{R}$ e $\mathcal{E}$. Opera sobre um subconjunto ordenado de *mathcalU* (do qual tem acesso randômico) e resulta em outro subconjunto ordenado de $\mathcal{U}$."[26]

[17] Tradução de *A universe of all possible concepts.*

[18] Tradução de *(...) a partial function from $\mathcal{L}$ to functions yielding real numbers in [0, 1]. (...) 0.5 to mean Boolean true and less than 0.5 to mean Boolean false; the need for the real values will become clear below.*

[19] Tradução de *A function interpreting $\mathcal{R}$, resulting in a function indicating adherence of a concept to $\mathcal{R}$*

2.3 O modelo de improvisação

McLean realiza uma comparação entre o *Universo de possibilidades* de Wiggins com o *Modelo de Improvisação* de Pressing. No entanto, McLean argumenta que:

> Pressing discute comportamento criativo no contexto do Modelo de Improvisação, e de fato é parte da Ferramenta de Estruturação de Sistemas Criativos. (...) Durante a transferência de notação do Modelo de Improvisação para a Ferramenta de Sistemas Criativos, nós consideramos improvisação musical de uma maneira clara e temos uma linguage comum na qual comparar com outros modelos [27].

Segundo Pressing, o Modelo de Improvisação é "um esboço para uma teoria geral da improvisação integrada com preceitos da Psicologia Cognitiva (...) teoria do comportamento de improvisação na música" (Pressing, 1987, p. 2).

Este modelo será utilizado para especificar performances exemplares, como o caso investigado neste trabalho, *Study in Keith*. Por exemplo, uma improvisação particionada em diferentes sequências pode ser parcialmente mapeada em categorias, como blocos sonoros, referentes conceituais e normas estilísticas, conjuntos de objetivos e processos.

Um sumário sobre o modelo de improvisação é apresentado na Tabela 4.

---

20 Tradução de *All conceptual spaces are non-strict subset.*

21 Tradução de *A function interpreting $\mathcal{R}$, resulting in a function indicating adherence of a concept to $\mathcal{R}$.*

22 Tradução de *Rules defined within $\mathcal{L}$ to define a traversal strategy to locate concepts within $\mathcal{U}$*

23 Tradução de *(...) set of rules which allows us to evaluate any concept we find in C and determine its quality, according to whatever criteria we may consider appropriate.*

24 Tradução de *Rules defined within $\mathcal{L}$ which evaluate the quality or desirability of a concept c.*

25 Tradução de *We need a means not just of defining the conceptual space, irrespective of order, but also, at least notionally, of enumerating it, in a particular order, under the control of $\mathcal{T}$ – this is crucial to the simulation of a particular creative behaviour by a particular $\mathcal{T}$.*

26 Tradução de *A function interpreting the traversal strategy $\mathcal{T}$, informed by $\mathcal{R}$ and $\mathcal{E}$. It operates upon anordered subset of mathcalU (of which it has random access) and results in another ordered subset of $\mathcal{U}$.*

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

28 *A cluster of sound events.*

29 A sequence of E event clusters, where event cluster onsets do not overlap with those of a following one

30 An improvisation, partitioned by interrupts into a number of K sequences

31 An optional referent, such as a score or stylistic norm

32 A set of current goals.

33 Long term memory.

34 An array of objects.

35 An array of objects Features.

36 An array of Process

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

Representação	Significado
E'	Um bloco de eventos sonoros[28]
K'	Uma seqüência de blocos de eventos E, onde um bloco de eventos não se sobrepõe com o seguinte[29]
I'	Uma improvisação, particionada por interrupções em um número de K sequências[30]
R'	Um referente opcional, tal como uma partitura ou uma norma estilística[31]
G'	Um conjunto de objetivos [32]
M'	Uma memória de longo prazo[33]
O'	Um conjunto de objetos[34]
F'	Um conjunto de características dos objetos[35]
P'	Um conjunto de processos[36]

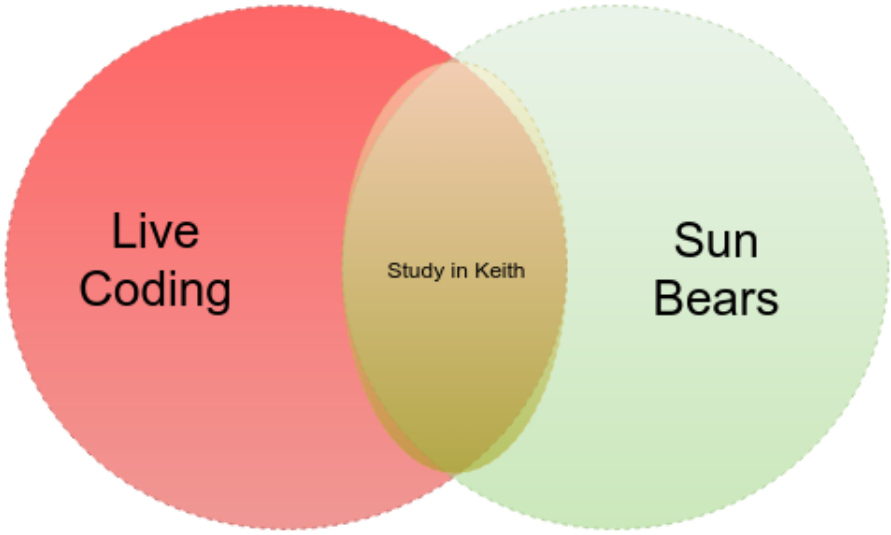


Figura 7 – Representação da justaposição entre dois epaços conceituais. A região em marrom representa um grupo de conceitos transitórios, bem como os limites desta transição. **Fonte**: autor.

2.4 Diagramação dos espaços conceituais

Formalmente, a figura acima pode ser representada como na Equação 2.2 , se desconsiderarmos qualquer outros espaços conceituais.

Example 2.1 (Representação formal da Figura 7)

$$\mathcal{C}_{Study\ in\ Keith} = \mathcal{C}_{live\ coding} \bigcup \mathcal{C}_{Sun\ Bears} \tag{2.1}$$

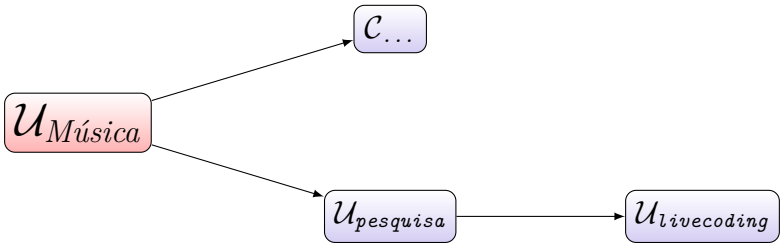
Este grupo também pode ser descrito como uma lista de propriedades como na **??**:

Example 2.2 (Representação formal das propriedades da Figura 7)

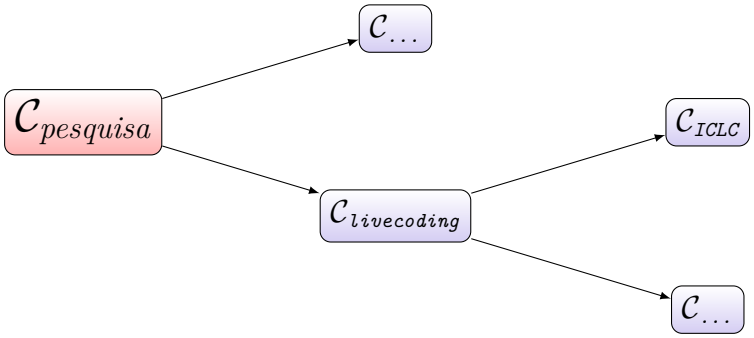
$$\mathcal{C}_{SK} = [\mathcal{E}'_{SK},\ \mathcal{K}'_{SK},\ \mathcal{I}'_{SK},\ \mathcal{R}'_{SK},\ \mathcal{G}'_{SK},\ \mathcal{M}'_{SK},\ \mathcal{O}'_{SK}F',\ \mathcal{P}'_{SK}] \tag{2.2}$$

Nos diagramas abaixo, $C_{...}$ representa qualquer espaço conceitual abstrato (que pode incluir outro previamente apresentado). Entre os elementos iniciais (raízes, vermelho) e transitórios (nós, azul), ocorrem as ramificações (ramos, linhas pretas), isto é, a exploração de conceitos dentro de outros conceitos. De um lado, a aplicação de regras de validação sobre o universo conceitual da pesquisa (tudo aquilo que foi produzido em dois anos de mestrado) gerou o espaço conceitual desta tese. Estas regras de validação foram, em sua maior parte, os processos de orientação e qualificação. Em outras palavras, $\mathcal{C}_{pesquisa}$= $[[\mathcal{R}_{pesquisa}]](\mathcal{U}_{pesquisa})$.

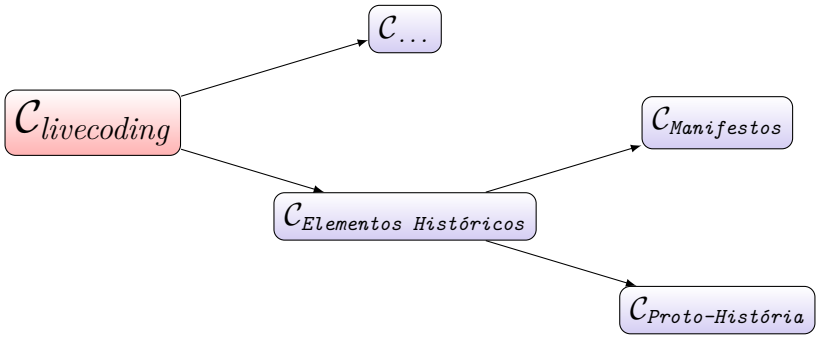
Example 2.3 (Representação do universo conceitual da *pesquisa*)
O Universo de Conceitos da pesquisa, $\mathcal{U}_{pesquisa}$, é um recorte do universo conceitual da música, $\mathcal{U}_{música}$:



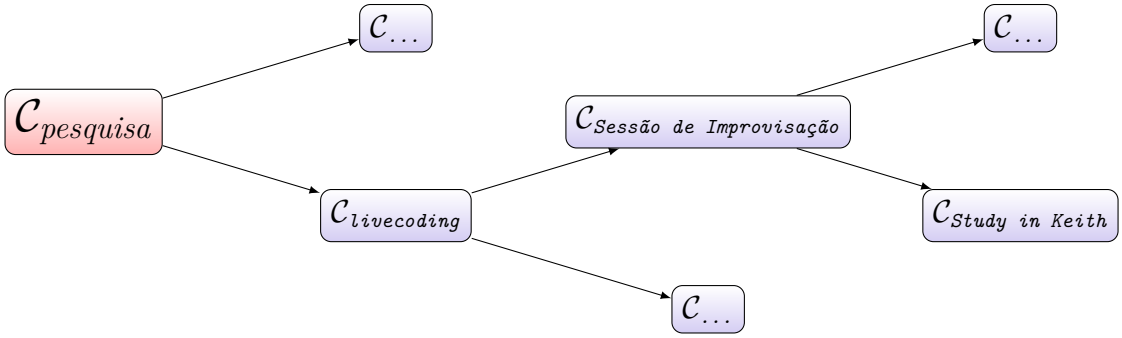
No primeiro capítulo, incluímos um subjconjunto neste Espaço Conceitual da Pesquisa. Este subconjunto é constituído pelos termos representados na Figura 4 (p. 17), e no Apêndice B (p. 57).



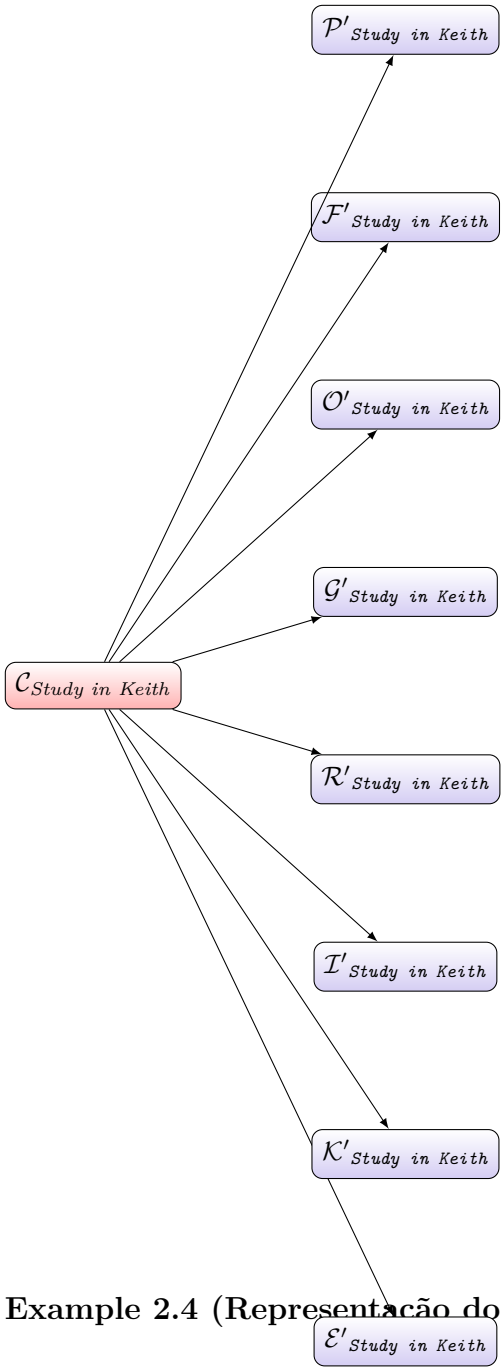
Podemos inclur elementos históricos, o período transitório entre 1970 e 2000 (*circa*), onde emanciparam as práticas e as regras heurísticas.



Por último, $\mathcal{C}_{pesquisa}$ investiga o *live coding* a partir de um caso específico:



Por outro lado $\mathcal{C}_{Study\ in\ Keith}$ pode ser definido pelo modelo de improvisação de Pressing (Tabela 4, 39).



Example 2.4 (Representação do modelo de improvisação para *Study in Keith*.)

2.5 Formalização

O espaço conceitual do *livecoding* é definido como uma função de interpretação das regras de validação (o que pode ser ou não considerado como próprio de uma categorização musical), de gosto (questões de estilo) e de localização transversal de conceitos (conceitos internos que permitem o cruzamento com outros conceitos):

Example 2.5 (Delimitação de regras para o *live coding* e para *Study in Keith.*)

$$\mathcal{C}_{livecoding} =<<< \mathcal{R}_{livecoding}, \mathcal{T}_{livecoding}, \mathcal{E}_{livecoding} >>> \tag{2.3}$$

As regras de validação foram estudadas neste trabalho como as regras heurísticas do *live coding*. Isto é, que conjunto de métodos são utilizados para caracterizar uma performance de *live coding* como tal? Elementos históricos, e ideológicos (divulgados em manifestos), são levantados para responder esta pergunta .

$$\mathcal{R}_{live\ coding} = \mathcal{R}_{Proto\text{-}história} \bigcup \mathcal{R}_{Manifestos} \tag{2.4}$$

Por outro lado, este estudo abandonou a investigação das regras de gosto, tema que pode ser melhor explorado em trabalhos posteriores, a partir de Jr. (2003), Sá (2006), Sá (2009).

A tarefa de localização transversal de conceitos é trabalhada no último capítulo. O espaço conceitual de *Study in Keith* está contido no espaço conceitual do *live coding* através da união entre os conceitos deste último, com os espaços conceituais dos concertos *Sun Bears*, de Keith Jarret, $\mathcal{C}_{Sun\ Bears} \subset \mathcal{C}_{live\ coding}$.

Expomos na equação 2.6 o espaço conceitual multidimensional do *Study in Keith*. Isto é, a aplicação de regras de validação do *live coding* e regras de validação dos Concertos *Sun Bear*:

Example 2.6 (Aplicação)
O espaço conceitual não-estrito da pesquisa é um função de interpretação das regras de validação do *live coding*, e das regras de validação do disco *Sun Bears*, sobre o Universo de conceitos do *livecoding*

$$\mathcal{C}_{pesquisa} = [[\mathcal{R}_{live\ coding} \bigcup \mathcal{R}_{Sun\ Bears}]](\mathcal{U}_{live\ coding}) \tag{2.5}$$

3 Estudo de caso

ESTE CAPÍTULO NÃO ESTÁ CORRIGIDO.

APENAS FOI INSERIDO PARA CONTEMPLAR O SUMÁRIO

Segundo Andrew Sorensen, "*A Study In Keith* é um trabalho para piano solo (NI's Akoustik Piano), inspirado nos concertos *Sun Bear* de Keith Jarrett's" (SORENSEN, 2009).

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

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

Uma nota pertinente sobre esta improvisação feita pelo próprio Sorensen: nos primeiros dois minutos do vídeo, existe um silêncio. Este silêncio é decorrente da construção das estruturas lógicas do programa. (Id., 2009). Este comportamento, do tempo de codificação, ao tempo de ação musical, é similar em outro vídeo analisado na **??**.

3.1 Concertos *Sun Bear*

TODO ...

---

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

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

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

4 http://www.ecmrecords.com/

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

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

3.2 Algorithms are Thoughts, Chainsaws are Tools

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

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

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

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

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

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

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

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

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

descrevendo os parâmetros através dos quais a máquina (midi) ganha expressão [9]

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

4 Conclusão

ESTE CAPÍTULO NÃO ESTÁ CORRIGIDO.

APENAS FOI INSERIDO PARA CONTEMPLAR O SUMÁRIO

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 3 vezes nas páginas 18, 23 e 58.

BECKER, H. *Art Worlds*. [S.l.: s.n.], 1982. Citado na página 34.

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 2 vezes nas páginas 25 e 26.

BODEN, M. *The Creative Mind: myths and mechanisms*. 2. ed. Routledge, Taylor & Francis Group, 1990. ISBN 0-203-34008-6. Disponível em: <http://www.pauladaunt.com/books>. Citado na página 35.

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 4 vezes nas páginas 11, 23, 24 e 25.

COLLINS, N. *Origins of live coding*. Durham University, 2014. Disponível em: <http://www.livecodenetwork.org/files/2014/05/originsoflivecoding.pdf>. Citado 2 vezes nas páginas 11 e 12.

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 5 vezes nas páginas 11, 12, 16, 28 e 30.

COLLINS, N.; OLOFSSON, F. A protocol for audiovisual cutting. p. 4, 2003. Disponível em: <http://quod.lib.umich.edu/cache//b/b/p/bbp2372.2003.011/bbp2372.2003.011.pdf#page=2;zoom=75>. Citado na página 28.

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 28.

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.

FENERICH, A.; OBICI, G.; SCHIAVONI, F. Marulho TransOceânico: performance musical entre dois continentes. p. 12, 2014. Disponível em: <https://www.academia.edu/t/M8Fvh/8178331>. Citado na página 46.

GRIFFITHS, D. Fluxus: Scheme livecoding. 2008. Disponível em: <http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>. Citado 2 vezes nas páginas 11 e 28.

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 na página 43.

MAGNUSSON, T. Algorithms as scores: Coding live music. v. 21, p. 19–23, 2011. Citado na página 12.

MAGNUSSON, T. Herding cats: Observing live coding in the wild. v. 38, n. 1, p. 8–16, 2014. Citado na página 31.

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 27.

MALENFANT, J.; JACQUES, M.; DEMERS, F.-N. A tutorial on behavioral reflection and its implementation. v. 38, n. 1, p. 65–76, 1996. Disponível em: <http://www2.parc.com/csl/groups/sda/projects/reflection96/docs/malenfant/malenfant.pdf>. Citado na página 23.

MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 4 vezes nas páginas 12, 18, 19 e 20.

McCallum, L.; SMITH, D. *Show Us Your Screens*. Vimeo, 2011. Disponível em: <https://vimeo.com/20241649>. Citado 2 vezes nas páginas 11 e 28.

McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: <https://www.academia.edu/467101/Music_improvisation_and_creative_systems>. Citado 6 vezes nas páginas 5, 13, 34, 36, 37 e 39.

McLean, A. *hacking perl music*. Youtube, 2006–07–30. Disponível em: <https://www.youtube.com/watch?v=fbefIdbSmD4>. Citado na página 12.

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 17.

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 18 e 25.

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 2 vezes nas páginas 12 e 15.

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 12, 18, 22 e 58.

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 18.

Pressing, J. Improvisation: Methods and models. In: *Generative processes in music*. (ed. J. Sloboda) Oxford University Press, 1987. p. 50. Disponível em: <http://musicweb.ucsd.edu/~sdubnov/Mu206/improv-methods.pdf>. Citado 3 vezes nas páginas 34, 38 e 39.

PROSPERO, C. D. Social participation. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 68–73. ISBN 978 0 85316 340 4. Citado 2 vezes nas páginas 12 e 33.

RAMSAY, S. *Algorithms are Thoughts, Chainsaws are Tools*. Vimeo, 2010. Disponível em: <https://vimeo.com/9790850>. Citado na página 46.

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 30.

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 27.

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 3 vezes nas páginas 16, 23 e 58.

ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado na página 23.

SORENSEN, A. *A Study in Keith*. Vimeo, 2009. Disponível em: <https://vimeo.com/2433947>. Citado 3 vezes nas páginas 5, 13 e 45.

SORENSEN, A. *Programming in Time - Live Coding for Creative Performances*. 2014. Disponível em: <https://www.youtube.com/watch?v=Sg2BjFQnr9s>. Citado na página 12.

SPIEGEL, L. The expanding universe: 1970s computer music from bell labs by laurie spiegel. disponível em <http://www.retiary.org/ls/expanding_universe/>. *Retiary*, 1975. Disponível em: <http://www.retiary.org/ls/expanding_universe/>. Citado 3 vezes nas páginas 18, 20 e 21.

SuperCollider.ORG. *SuperCollider Overviews: JITLib - An overview of the Just In Time library*. 2014. Citado na página 12.

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 na página 43.

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 43.

Thornton, C. A quantitative reconstruction of boden's creativity theory. p. 29, 2007. Disponível em: <http://users.sussex.ac.uk/~christ/papers/boden-reconstruction.pdf>. Citado 3 vezes nas páginas 34, 35 e 36.

TOPLAP. *TOPLAP001 - A prehistory of live coding*. TOPLAP, 2007. Disponível em: <http://toplap.org/wiki/TOPLAP_CDs>. Citado na página 11.

TURNER, V. W.; HARRIS, A. *"Liminality and Communitas". The Ritual Process: Structure and Anti-Structure.* [S.l.: s.n.], 1969. Citado na página 33.

TURNER, V. W.; HARRIS, A. *"Liminality and Communitas". The Ritual Process: Structure and Anti-Structure.* Cornell University Press, 1969. Disponível em: <http://www.iupui.edu/~womrel/Rel433%20Readings/SearchableTextFiles/Turner_RitualProcess_chaps3&5.pdf>. Citado na página 33.

WANG, G. *Read me paper - Revision as of 01:11, 1 August 2005 - A Historical Perspective.* 2005. Disponível em: <http://toplap.org/wiki/index.php?title=Read_me_paper&oldid=60#A_Historical_Perspective>. Citado na página 25.

WARD, A.; McLean, A.; SHULGIN, A. *Listen to the Tools: Adrian Ward and Alex McLean Interviewed by Alexei Shulgin.* 2003. Disponível em: <http://www.m-cult.org/read_me/text/alex_ade.htm>. 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 11, 12, 26, 27, 31 e 58.

WIGGINS, G. A. A preliminary framework for description, analysis and comparison of creative systems. v. 19, n. 3592, p. 449–458, 2006. Disponível em: <http://axon.cs.byu.edu/Dan/673/papers/wiggins.pdf>. Citado 3 vezes nas páginas 34, 36 e 37.

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 58.

APÊNDICE A – Código fonte da nuvem de palavras sobre live coding

Apresentamos no exemplo A.1 o código-fonte, em linguagem python, utilizado para extração parcial do espaço conceitual da pesquisa (Figura 4, p. 17). A biblioteca utilizada, *Wordcloud*, pode ser encontrada em <https://github.com/amueller/word_cloud>.

O Código abaixo considera a seguinte situação: é possível converter um arquivo de texto em formato *.pdf* para formato *.txt*. Feita a conversão, é possível realizar um levantamento estatístico das palavras mais usadas (o que pode, parcialmente, indicar idéias e conceitos).

Existem programas online, como o encontrado no link <http://convertonlinefree.com/PDFToTXTEN.aspx>, que realizam a conversão. É necessária a correção de alguns erros de caracteres (ver Figura 8). Além disso, informações de cabeçalho, códigos-fontes, e outros elementos de editoração, foram descartados por considerarmos que não eram parte do corpo textual. Em outras palavras, descartamos palavras que não faziam parte de um discurso de texto, ou atrapalhavam o processo de criação da imagem. O que por sí não resolve todos os problemas, mas auxilia na elaboração da imagem.

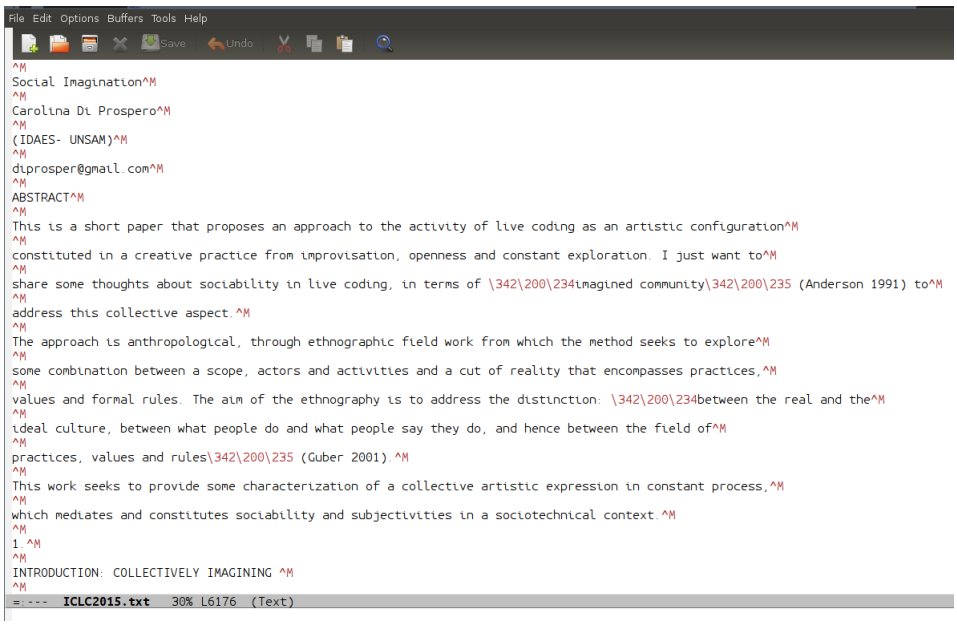


Figura 8 – Resultado da conversão do arquivo *pdf* para *txt* resulta em problemas de codificação que necessitaram ser corrigidos por comparação com o arquivo original. **Fonte**: autor.

Example A.1 (Código-fonte que utiliza a biblioteca wordcloud)

```
# Bibliotecas utilizadas
from urllib2 import urlopen
from bs4 import *
import re
import datetime
from iso import *
import matplotlib.pyplot as plt

from wordcloud import WordCloud

# Abra o arquivo em modo de leitura
t = open("ICLC2015.txt", "r").read()

# Gere uma nuvem de palavras
wc = WordCloud().generate(t)

# Mostre a imagem gerada com o matplotlib
# levando em conta as frequencias das palavras
plt.figure()
plt.imshow(wc)
plt.axis("off")
plt.show()
```

APÊNDICE B – Classes qualitativas de um Universo de conceitos do *live coding*

Com auxílio da biblioteca NLTK[1], categorizamos a nuvem de conceitos de acordo com sua função textual (ver exemplo B.1).

Example B.1 (Categorização dos dados)

```
# Importe bibliotecas
from os import path
from wordcloud import WordCloud
import math
import nltk

# Abra o arquivo em modo de leitura
t = open("assets/ICLC2015.txt", "r").read()
wc = WordCloud().generate(t)

# Organize as palavras por frequencia
# de presenca no texto
groups = [[] for i in range(10)]

for i, t in enumerate(wc.words_):
    freq = t[1]
    word = t[0]
    index = int(math.floor(freq*10))
    if freq >= index/10.0 and freq < (index+1)/10.0:
        print word
        print index
        groups[index-1].append(word)

# CLASSIFIQUE AS PALAVRAS
groups = [nltk.pos_tag(e) for e in groups]
print groups
```

[1] Disponível em <http://nltk.org/>.

O código acima gerou uma saída textual que foi reorganizada na tabela Tabela 5.

Uma breve análise da nuvem de palavras (ver Figura 4, p. 17), pode elucidar parte das questões-satélites do *live coding*. Na Tabela 5 filtrei parte dos resultados 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).

No caso dos sujeitos-humanos, podemos ver nomes de Nick Collins e Alex McLean, praticantes responsáveis pela criação de um manifesto, em parceria com Ward et al. (2004). 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*(ROBERTS; KUCHERA-MORIN, 2012; WYSE; SUBRAMANIAN, 2014)[2]. Ambos são ambientes de programação para de síntese sonora e composição algorítmica. Uma característica em comum destes ambientes, o procedimento de compilação de códigos, é conhecido como *Just In Time* (AYCOCK, 2003), dispositivo técnico que permitiu a execução de códigos durante o tempo de execução.

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*.

Adjetivos destacam características da prática. *Live* é a palavra-chave, e sugere uma prática de performance. *Visual* sugere uma característica fundamental, tanto quanto a Música, para uma performance. *Ensemble* destaca a natureza de grupos, isto é, poucas performances *solo* são realizadas se comparadas às performances de *duos*, *trios*.

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

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

Tabela 5 – 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