

Guilherme Martins Lunhani

***Live coding: uma improvisação de códigos
tonais em A Study in Keith (2009)***

12 de fevereiro de 2016

Guilherme Martins Lunhani

Live coding: uma improvisação de códigos tonais em A Study in Keith (2009)

Dissertação apresentada para pré-banca de qualificação no Programa Mestrado em Artes, Cultura e Linguagens do Instituto de Artes e Design da Universidade Federal de Juiz de Fora (UFJF), linha de Artes Visuais, Música e Tecnologia.

Universidade Federal de Juiz De Fora – UFJF

Instituto de Artes e Design – IAD

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

Orientador: Prof. Dr. Luiz Eduardo Castelões

12 de fevereiro de 2016

*O Tao produziu o Um.
O Um produziu o Dois.
O Dois produziu o Três.
O Três produziu os dez mil seres.*

¹

¹ Lao Tsé. *Tao Te King*. Versão integral e comentários. 3^a edição. Editorial Attar.<<http://tao-te-king.org/42.htm>>.

Agradecimentos

Ao que é impossível pronunciar o verdadeiro Nome, mas cuja potência é o próprio sentido da palavra criativa.

Aos sem nome, anônimos da Rua de Juiz de Fora, que deram sentido à fraqueza da pergunta deste trabalho.

Para uma família em cada canto do Universo, Jair, Olímpia e Júlia (e humanos pitucha, bila, pituca).

Aos Professores Dr. Luiz Eduardo Castelões, Dr. Alexandre Fenerich e Dr. Flávio Luiz Schiavonni, fundamentais no apoio institucional; na sugestão de leituras; na cobrança de prazos; nas críticas; nas conversas sobre Música. À FAPEMIG por suprir esta lacuna, em um momento delicado nas finanças da Universidade Brasileira.

Aos amigxs que estão (ou moraram em Juiz de Fora): Glerm (Soares), Tiago Rubini, Anna Flávia. Aos amigxs de Campinas e São Paulo, que estiveram presentes ou na memória: Celso, Dani, Evandro (bixin), Fábio, Felício, Frederico (d menor), Gustavo, Larissa (humanos zé e diva), Rebechi (humana vivi), Simone (humano foucault), Tati e Ruan (bebê Noah), Dhiego e Luisa , ao pessoal da república Lado C, João, Heron, Igor. Ao Gustavo e Patrick (pretinha). Ao velho amigo Picchi!

Ao Professor Hans Joachim Koellreutter pelo centenário.

Resumo

Este documento é uma nota pedagógica para interessados em improvisação de códigos. Indicamos eventos históricos ligados à Música, e situamos a análise de um trecho exemplar (e pedagógico) desta história. Através do Quadro Conceitual de Sistema Criativo de [McLean \(2006\)](#), [Wiggins \(2006\)](#), [McLean \(2011\)](#) analisamos o primeiro algoritmo gerador de uma sonoridade tonal em *A Study in Keith* de Andrew [Sorensen e Swift \(2009\)](#). Por outro lado, consideramos este exemplo representativo por vestir o programador com a figura de intérprete musical, e ao mesmo tempo, manter o espírito do pianista concertista em um ambiente rico de concepções musicais.

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

Sumário

Errata	ix
Introdução	xi
1 TRABALHOS RELACIONADOS	1
1.1 Tecelagem	2
1.1.1 Charles Babbage e Joseph-Marie Jacquard	2
1.1.2 Weavecoding	3
1.1.2.1 Slub	7
1.2 Dança	8
1.2.1 Algorave	8
1.2.2 Coreografia	12
1.3 Música computacional	16
1.3.1 Música de concerto e audiovisual	17
1.3.2 Telepresença e espaços virtuais	21
1.3.3 Proto-história musical	22
1.3.3.1 Pietro Grossi	22
1.3.3.2 Baía de São Francisco	24
1.3.3.3 Ron Kuivila	26
1.4 LAPTOP	27
1.4.1 TOPLAP	29
1.4.1.1 <i>Show us your screens</i>	30
1.4.1.2 Obscurantismo é perigoso, mostre-nos suas telas	31
1.4.1.3 Algorithms are Thoughts, Chainsaws are Tools	32
1.5 Discussão	34
2 METODOLOGIA	37
2.1 Criatividade e Sistemas Criativos	37
2.1.1 O programa será transcendido - Língua Artificial é o caminho	38
2.1.2 Bricolagem como estratégia transversal	42
2.2 Sistemas criativos	44
2.2.1 Quadro Conceitual de sistemas criativos	45
2.3 O modelo de improvisação	47
2.4 Diagramação dos espaços conceituais	49
2.5 Formalização	51

3	ESTUDO DE CASO	53
3.1	Objetivo	54
3.2	Referentes Opcionais	54
3.2.1	Concertos Sun Bear	54
3.2.2	NI-Akoustik Piano	56
3.2.3	Ambiente e Linguagem: Impromptu	58
3.2.4	Extempore	58
3.2.5	Scheme	59
3.3	Blocos de Eventos	61
3.3.1	Primeiro evento sonoro: a sensação de quietude	61
3.3.2	Segundo Evento Sonoro	64
3.3.3	Terceiro Evento Sonoro	65
	Conclusão	67
	Referências	71
	APÊNDICE A – CÓDIGO FONTE DE UM UNIVERSO CONCEITUAL COMO NUVEM DE PALAVRAS SOBRE O IMPROVISO DE CÓDIGOS	77
A.1	Classes qualitativas de um Universo de conceitos do <i>live coding</i>	79
	APÊNDICE B – SUGESTÕES PROTO-HISTÓRICAS	83
B.1	GROOVE	83
B.2	SuperCollider	86
B.2.1	Just In Time Library (JITLib)	89

Lista de ilustrações

Figura 1 – Tecido resultante da prática <i>Weaving code</i> . Fonte: Griffths (2015a).	5
Figura 2 – Performance no Foam Kernow. Fonte: Griffths (2015b).	6
Figura 3 – Alex McLean manipulando uma Matriz de botões para tecelagem, conectado a um Raspberry Pi. Fonte: Griffths (2015b).	6
Figura 4 – Performance do duo Canute (Karlesruhe, 2015) Fonte: Canute... (2015-27-01).	10
Figura 5 – Performance do duo Mico Rex (Londres, 2013) Fonte: Griffths (2013).	11
Figura 6 – Performance do duo Mico Rex (Londres, 2013) Fonte: Griffths (2013).	11
Figura 7 – Dançarina (anônima) controlada por Kate Sicchio (2015) através de uma codificação improvisada. Fonte: < https://www.youtube.com/watch?v=uAq4BAbvRS4 >.	16
Figura 8 – Performance de <i>screenBashing</i> . Fonte: < https://vimeo.com/148626379 >.	18
Figura 9 – Definição do significado de TOPLAP. Fonte: Ramsay (2010).	29
Figura 10 – Modelo de bricolagem para o processo criativo realizado por um artista-programador. Fonte: McLean (2011, p. 122).	39
Figura 11 – Exemplo de uma característica de viscosidade e notação secundária no PureData. Fonte: autor.	41
Figura 12 – Representação da justaposição entre dois espaços conceituais. A região em marrom representa um grupo de conceitos transitórios, bem como os limites desta transição. Fonte: autor.	48
Figura 13 – Transcrição do motivo gerador do disco Kyoto, parte 1. Fonte: autor.	56
Figura 14 – Sistema de tomada de som para produção de um VSTi. Fonte: < http://www.native-instruments.com/en/products/komplete/keys/definitive-piano-collection/ >.	57
Figura 15 – Piano Disklavier de armário, com a parte interna exposta para exibir a placa-mãe. Fonte: wikimedia.org	57
Figura 16 – Primeiros eventos musicais gerados a partir das primeiras estruturas válidas de código. Fonte: autor.	65
Figura 17 – Primeiros perturbações sistêmicas. Fonte: autor.	65
Figura 18 – Definição de <i>live coding</i> : “Insira a definição aqui”. Fonte: Collins (2014).	68
Figura 19 – Resultado da conversão do arquivo <i>pdf</i> para <i>txt</i> resulta em problemas de codificação que necessitaram ser corrigidos por comparação com o arquivo original. Fonte: autor.	78
Figura 20 – Esquema de concepção do projeto GROOVE. Fonte: (MATHEWS; MOORE, 1970).	84

Figura 21 – Laurie Spiegel configurando a saída analógica do GROOVE, durante a produção de *The Expanding Universe*. **Fonte:** (SPIEGEL, 1975). 85

Errata

Existe uma seção com problemas de compilação L^AT_EX (ver [subseção 2.1.2](#), p. [42](#)). Isto é, caracteres que deveriam aparecer como \\$ aparecem como \\\$. A segunda seção de análise está sendo finalizada (ver [subseção 3.3.2](#), p. [64](#)) e necessita de edições e mais transcrições. A terceira seção de análise não foi concuída ainda (ver [subseção 3.3.3](#), p. [65](#)).

Introdução

Em seu artigo “A filosofia à venda, a dourada ignorância e a aposta de pascal”, Santos (2008, p. 11–12) descreve a imagem mental de uma feira do conhecimento, onde teorias são antropomorfizadas, escravizadas e vendidas: “determinismo, livre arbítrio, universalismo, relativismo, realismo, construtivismo, marxismo, liberalismo, neoliberalismo, estruturalismo, pós-estruturalismo, modernismo, pós-modernismo, colonialismo, pós-colonialismo, etc.”. As idéias perderam a utilidade para os ex-adeptos, que não estão mais interessados em comprá-las. E vendem aos que supõem algum valor. Para efetuar a venda, é necessário estabelecer uma relação de custo-benefício, negociadas através de respostas às perguntas: “qual a utilidade que esta ou aquela teoria poderá ter para mim? Qual o preço?”. A valorização ocorre quando esta teoria se torna mais apelativa que aquela. Com a concorrência, a livre-associação dos vendedores regulamentará compras e vendas de conhecimentos conforme seu interesse mais fundamental: se todas teorias forem vendidas não existirá teoria para se vender amanhã.

Podemos pensar que Santos realiza uma metáfora de um Mercado contemporâneo do conhecimento; nos termos deste documento, espaços conceituais da Filosofia e da Economia se conectam por imagens mentais e metáforas. No entanto Santos argumenta que tal imagem mental não é uma metáfora contemporânea, mas é anterior à formação do espírito científico moderno: no texto satírico *A venda de filosofias* (165), Luciano de Samósata (125 – 181?), escreve sobre um mercado estimulado por Zeus e gerenciado por Hermes:

Hermes atrai os potenciais compradores, todos comerciantes, gritando alto e bom som “À venda! Uma variedade sortida de filosofias vivas! Posições de todo o tipo! Pagamento à vista ou mediante garantia!” (1905: 190). A “mercadoria” vai sendo exposta, os comerciantes vão chegando e têm o direito de interrogar cada uma das filosofias à venda, começando invariavelmente com a pergunta pela utilidade para o comprador e a sua família ou grupo. O preço é estabelecido por Zeus que, por vezes, se limita a aceitar ofertas feitas pelos comerciantes compradores. A venda tem pleno êxito e Hermes termina, ordenando às teorias que deixem de oferecer resistência e sigam com os seus compradores, ao mesmo tempo que avisa o público: “Senhores, esperamos vê-los amanhã. Estaremos oferecendo novos lotes úteis para homens comuns, artistas e comerciantes”

Recolhendo esta imagem mental do Mercado de conhecimentos escravizados, espelhamos a metáfora para o assunto específico deste documento. As filosofias vendidas estão em uma feira chamada *live coding*, que traduzimos por improvisação de códigos. A tradução literal *codificação ao vivo* é o termo mais adequado; mas não cria a imagem mental necessária para apresentar um modelo lógico de improvisação musical (ver Capítulo 2,

p. 37). Segundo McLean (2011, p. 129), “O termo *live coding* emergiu em torno de 2003, para descrever a atividade do grupo de praticantes e pesquisadores que começaram a desenvolver novas abordagens na criação de música computacional e animação de vídeo.”². Ali vendem o jazz (SORENSEN; SWIFT, 2009; SORENSEN, 2013), a música algorítmica (MAGNUSSON, 2011), o minimalismo, *live computer music*, música ambiental, música *rave* (WARD et al., 2004), e mais recentemente, a música-ruído. Além disso são vendidas teorias da tecelagem, do audiovisual, da dança, e do lado científico, as ciências antropológicas e cognitivas. Compramos uma amostra, cujo exemplares foram divididos em três grupos (ver Objetivos, p. xiii).

Espaço Conceitual do Pensamento Ortopédico no *live coding*

Neste documento, atentamos para três pontos interdisciplinares (os pontos sempre tendem à improvisação de códigos musicais). No decorrer da pesquisa fomos pouco a pouco limitados a reduzir o escopo para uma investigação mais detalhada. No entanto esta própria redução, orientada pela prática científica moderna é comentada e classificada por Santos. São dois os tipos de limitação que podem acontecer: o Pensamento Ortopédico e Ecologia de Saberes. Não é nossa intenção discutir a Ecologia de Saberes³, mas atentar para o Pensamento Ortopédico que permeia esta prática de improvisação (ou pelo menos foi assim que percebemos a prática).

A própria noção de Pensamento Ortopédico pode ser tema de investigação na área de cognição, mas limitamo-nos a fazer alguns comentários sobre o termo, para que seja coerente com a metáfora feita no início deste capítulo. A descrição simbólica do objeto de pesquisa, como uma feira filosofias musicais acompanhadas de pesquisas científicas. A primeira metáfora é o PENSAMENTO ORTOPÉDICO (isto é, “o processo especialista do médico responsável em corrigir deformidades do corpo”)⁴. Por exemplo, enquanto Ward et al. (2004) comentam a apropriação do termo **generativo** na improvisação de códigos, esta apropriação é revista por McLean (2011) como um processo descritivo (ver subseção 1.4.1.1, p. 30). A segunda metáfora A RAZÃO INDOLENTE (isto é, “insensibilidade com respeito às consequências do processo de correção”): “A carência a respeito da finitude transforma-se num problema técnico-científico, enquanto a carência a respeito da diversidade infinita é ignorada como um não-problema.” (SANTOS, 2008, p. 15). Por exemplo, o problema da multidão de conceitos, metáforas e imagens mentais de uma improvisação de códigos é reduzido como um objeto com diversas propriedades, que além da música, incluem as ciências cognitivas (Pressing, 1984; Pressing, 1987) e lógica (McLean, 2006; FORTH;

² Tradução nossa de *The term live coding emerged around 2003, to describe the activity of group of practitioners and researchers who had begun developing new approaches to making computer music and video animation.*

³ Cf. SANTOS, 2007.

⁴ Disponível em <<http://www.priberam.pt/dlpo/ortopedia>>

(WIGGINS; MCLEAN, 2010). A terceira metáfora é O PENSAMENTO ABISSAL (isto é, “a percepção de uma distância que delimita conhecimentos”) (SANTOS, 2007, p. 1-4):

Consiste num sistema de distinções visíveis e invisíveis, sendo que as invisíveis fundamentam as visíveis. As distinções invisíveis são estabelecidas através de linhas radicais que dividem a realidade social em dois universos distintos: o universo ‘deste lado da linha’ e o universo ‘do outro lado da linha’. A divisão é tal que ‘o outro lado da linha’ desaparece enquanto realidade, torna-se inexistente, e é mesmo produzido como inexistente. (...) **O pensamento abissal moderno salienta-se pela sua capacidade de produzir e radicalizar distinções.** Contudo, por mais radicais que sejam estas distinções e por mais dramáticas que possam ser as consequências de estar de um ou do outro dos lados destas distinções, elas têm em comum o facto de pertencerem a este lado da linha e de se combinarem para tornar invisível a linha abissal na qual estão fundadas.⁵

Na improvisação de códigos, as radicalizações das distinções foram elaboradas a partir de modificações em uma pergunta, e o tratamento dado a elas pela comunidade de improvisadores-programadores. “Como as pessoas improvisam?”⁶ (Pressing, 1987). No caso específico deste trabalho, como improvisam os programadores? O caráter simplista da pergunta não esconde um contexto. A pergunta foi anteriormente elaborada por Pressing (1984): “como pode o psicólogo não-especialista acessar os resultados de tais improvisações quando os especialistas podem discordar entre eles?”⁷. No caso da improvisação de códigos, como o musicólogo não-especialista pode acessar os resultados de improvisações de programadores? Deste problema interdisciplinar, de um lado entre o específico e o generalizado, e de outro, entre a psicologia e música, outras ordens de conhecimento são acessadas. Buscamos extrair o que foi útil: *i*) descrever alguns comportamentos criativos históricos; *ii*) descrever uma perspectiva cognitivista aplicada à computação musical (o pensamento ortopédico) e ;*iii*) analisar conceitos que permeiam um algoritmo musical.

Objetivos

- Investigar um Universo de Conceitos sobre a improvisação de códigos (*live coding*);
- Investigar um método de análise/criação para uma improvisação de códigos;
- Investigar um Espaço Conceitual de uma sonoridade de um algoritmo musical de uma improvisação de códigos;

⁵ Grifo nosso.

⁶ Tradução nossa de *How do people improvise?*.

⁷ Tradução nossa de *how will the non-specialist psychologist assess the results of such improvisations when the specialists may disagree among themselves?*.

Estrutura dos Capítulos

No [Capítulo 1](#) selecionamos três abordagens, escolhidas por manterem alguma conexão com a improvisação de códigos no contexto musical. No [Capítulo 2](#) apresentamos um modelo de formalização da criatividade, do ponto de vista do Modelo de Improvisação discutido por [McLean \(2006\)](#), [Forth, Wiggins e McLean \(2010\)](#), [McLean \(2011\)](#). No [Capítulo 3](#), organizamos conceitos para analisar o contexto de uma sonoridade de *A Study in Keith* (2009) de Andrew Sorensen. O [Apêndice A](#) foi adicionado para expor o material que estimulou o interesse pelo tema discutido. O [Apêndice B](#) sugere a inclusão de um trabalho de [Mathews e Moore \(1970\)](#) no âmbito proto-histórico da improvisação de códigos.

1 Trabalhos relacionados

Giovanni Mori (2015a, p. 117) define a improvisação de códigos em relação à Música, Imagens em Movimento, Dança ou Tecelagem. É importante esclarecer que essa definição denota a aplicação em qualquer outra área, não apenas como metáfora, mas como técnica para gerenciar o fluxo criativo:

“*Live coding* é uma técnica artística de improvisação. Pode ser empregada em muitos contextos diferentes de performance: dança, música, imagens em movimento e mesmo tecelagem. Eu concentrei minha atenção no lado musical, que parece ser o mais proeminente.”¹

O problema desta definição é que ela não contempla uma classe puramente técnica da improvisação de códigos, como por exemplo os registros audiovisuais de um tutorial de como escrever um aplicativo *web*². No entanto, a premissa da pesquisa é, assim como Mori, situar a improvisação de códigos do ponto de vista musical. Descartamos uma discussão específica sobre imagens em movimento para evitar cair em uma digressão infinita, já que este é outro campo proeminente; mas uma menção será feita em relação ao contexto musical. A discussão sobre tecelagem (ver [seção 1.1](#), p. 2) contextualiza a história da computação com a prática artística investigada. Além disso, dois improvisadores-programadores chave para a prática, Alex McLean e Dave Griffiths, serão apresentados. O primeiro é a Música Eletrônica para Dançar³ (ver [subseção 1.2.1](#), p. 8), e o segundo nega o som como resultado da improvisação de códigos (ver [subseção 1.2.2](#), p. 12). Adiante, a característica musical da improvisações de códigos será explorada do ponto de vista histórico (ver [seção 1.3](#), p. 16). Por último, será discutido um conjunto de regras práticas que emergiram durante uma disputa acadêmica. Isto é, a partir de uma problematização feita por Schloss (2003), sobre o papel cênico (e visual) do músico que carrega um computador no palco de performance, uma resposta conjunta de McLean, Griffiths, Amy Alexander, Adrian Ward, Fredrik Olofsson, Julian Rohrhuber e Nick Collins, ([WARD et al., 2004](#)), possibilitou a proliferação de idéias e a estruturação de um programa de investigação científica próprio para a improvisação de códigos (ver [seção 1.4](#), p. 27).

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

² Disponível em <<https://www.youtube.com/watch?v=dHtyDron5ik>>

³ Cf. RIETVELD, 2013.

1.1 Tecelagem

Contextualizar a atividade têxtil é uma forma de criar uma imagem mental, de como funciona o processo de computação. Seria possível usar a imagem de um ábaco. Mas essa última imagem não considera potenciais leitores de um programa de pesquisa que inclui investigadores na área de Moda (como é o caso do Programa de Pós Graduação em Artes, Cultura e Linguagens da Universidade Federal de Juiz de Fora). Não aprofundaremos o assunto de Moda, mas sim buscamos ilustrar um código de computador. Nas palavras de Griffiths (2015a),

“Um dos potenciais da tecelagem que eu fiquei mais interessado é a capacidade de demonstrar fundamentos de *softwares* por fios – parcialmente tornar a natureza física da computação auto-evidente, mas também como uma maneira de modelar novas formas de aprender e a entender o que são os computadores.”⁴

Buscamos demonstrar na próxima seção esta natureza física da computação. Através de retrospecto na história da computação irá auxiliar na inclusão da tecelagem na definição de Mori.

1.1.1 Charles Babbage e Joseph-Marie Jacquard

Os computadores atuais são máquinas desenvolvidas com base no modelo teórico elaborado por Alan Turing (1912-1954). Uma representação simplista considera uma fita abstrata de tamanho variável (o quanto for necessário), dividida em células, cada uma com um alfabeto finito. Cada alfabeto possui uma quantidade de símbolos de representação finita. Um cabeçote leitor desta fita lê as instruções escritas em cada célula, e depois passa para a próxima célula. Um registrador de estados desta fita, memoriza qual foi a última operação realizada na última célula executada. Uma tabela de ações indicará novas instruções, que serão escritas nesta fita.

Um modelo anterior ao de Turing foi elaborado por Charles Babbage (1791 – 1871), a *máquina analítica*, entre 1834 e 1836, revisado em 1837. Sua construção ocorreu após um colapso na construção de sua *máquina diferencial*. O projeto não vingou, mas a partir de 1838, Babbage se envolveu com a exploração intelectual dos conceitos elaborados, para otimizar o projeto e reduzir seu custo de construção. Uma sequência de seminários em Turin (1840) resultou em uma publicação sobre a máquina analítica, em francês, escrita por um cientista italiano (L.F. Menebrea). A Condessa de Lovelace (Ada Augusta Byron King), traduziu, sob supervisão de Babbage, esta publicação para o inglês. Historicamente, os primeiros programas de computador (para serem executados na máquina analítica)

⁴ Tradução nossa de *One of the potentials of weaving I'm most interested in is being able to demonstrate fundamentals of software in threads – partly to make the physical nature of computation self evident, but also as a way of designing new ways of learning and understanding what computers are..*

foram escritos ambos por Ada e Babbage. O primeiro programa escrito era dedicado ao cálculo de uma sequência numérica conhecida como *Números de Bernoulli*⁵. Apenas uma parte da máquina foi construída antes da morte de Babbage⁶.

Segundo McLean (2011, p.14–21), o mecanismo do projeto de Babbage é inspirado na máquina de tear de Joseph-Marie Jacquard (1752 – 1834). A principal contribuição da invenção, para a computação, foi um sistema que consiste em um cabeçote leitor de cartões perfurados. Na máquina de tear de Jacquard, a organização dos furos indicam, até hoje, uma rotina têxtil (ver ??, p. ??). Já no computador mecânico de Babbage, o cartão perfurado indicava estados binários que conduzem ao cálculo numérico:

“A indústria têxtil vislumbrou a primeira máquina programável de ampla utilização: a cabeça de tear Jacquard, uma tecnologia ainda usada. Longas tiras de cartão são alimentados na cabeça de tear Jacquard, que lê padrões perfurados no cartão para guiar a intrincada padronização de tecidos. O cabeçote Jacquard não computa, mas foi admirado por Charles Babbage, inspirando o trabalho na sua máquina analítica mecânica, a primeira concepção de um computador universal programável. Embora Babbage não tenha obtido sucesso em construir a máquina analítica, seu projeto inclui um mecanismo de entrada de cartão similar ao cabeçote Jacquard, mas com padrões perfurados descrevendo cálculos abstratos ao invés de fios têxteis.”⁷

1.1.2 Weavecoding

Se até hoje o mesmo sistema de Jacquard é utilizado para materializar imagens mentais de formas geométricas, como improvisadores de códigos elaboram uma *estratégia transversal* (ver subseção 2.1.2, p. 42), de transformar a imagem mental destas formas geométricas em códigos, e de códigos para o resultado desejado? De outra forma, como é realizada uma improvisação de códigos audio visuais, musicais e têxteis? Exemplificamos o caso com um grupo que criou o conceito *weavecoding*. Sua definição será dada conforme apresentamos o grupo *Weaving codes*⁸; ilustramos uma investigação informal, um encontro no Foam Kernow⁹.

⁵ Allan G. Broomley, *Charles Babbage's Analytical Engine*, 1838. Disponível em <<http://athena.union.edu/~hemmend/Courses/cs80/an-engine.pdf>>

⁶ Disponível em <http://www.sciencemuseum.org.uk/objects/computing_and_data_processing/1878-3.aspx>

⁷ Tradução nossa de *The textile industry saw the first programmable machine to reach wide use: the head of the Jacquard loom, a technology still used today. Long strips of card are fed into the Jacquard head, which reads patterns punched into the card to guide intricate patterning of weaves. The Jacquard head does not itself compute, but was much admired by Charles Babbage, inspiring work on his mechanical analytical engine (Essinger, 2004), the first conception of a programmable universal computer. Although Babbage did not succeed in building the analytical engine, his design includes a similar card input mechanism to the Jacquard head, but with punched patterns describing abstract calculations rather than textile weaves.* .

⁸ Disponível em <<http://kairotic.org/about/>>

⁹ Disponível em <<http://fo.am/kernow/>>

O grupo *Weaving codes* foi formado para investigar “padrões a partir das perspectivas de tecelagem e música, e através do desenvolvimento de uma linguagem de computador e código para descrever a construção de tecidos”¹⁰. É formado por membros da Universidades de Leeds, Nottingham Trent, Cambridge, Aberdeen, Copenhague; um museu (*Albert Museum*), uma rede de laboratórios transdisciplinares (FoAM Kernow); o Centro Dinamarquês para Pesquisa Têxtil, e Escola Robert Schumann de Música e Mídia de Düsseldorf.

Uma pequena digressão: dois membros deste grupo, Alex McLean e Dave Griffiths, são praticantes e organizadores de improvisações de código como artistas-programadores. A principal contribuição dos autores foi uma heurística da improvisação de códigos, *Lubeck04*, mais conhecido como “Mostre-nos suas telas”¹¹, dentro de um manifesto publicado como “Programação de Algoritmo Ao vivo e Organização Temporária para sua Promoção”¹² (WARD et al., 2004). Este tema será discutido adiante (ver [subseção 1.4.1.1](#), p. 30).

Do manifesto à codificação têxtil, [Griffiths \(2015a\)](#) apresenta um interessante exemplo. A partir de quatro tarefas fundamentais (rotinas), descritas no Exemplo 1.1, é possível elaborar um padrão como o apresentado na [Figura 1](#). A primeira rotina é *repeat*, uma repetição de ações por contagem [*loop*]; a segunda é *twist*, ou dar a volta em determinados pontos; a terceira, *weave-forward*, tecer à frente do ponto; e a quarta, *weave-back*, tecer atrás do ponto . Do lado direito da imagem (ver p. 5), é simbolizado o “código criptografado de tecido”, ou as operações fundamentais para um determinado padrão têxtil. Do lado esquerdo, seu resultado-padrão, uma textura de losangos e zigue-zagues.

Exemplo 1.1 (Um código-fonte que gera um tecido semelhante à [Figura 1](#).)

```
(twist 3 4 5 14 15 16)
```

```
(weave-forward 3)
```

```
(twist 4 15)
```

```
(weave-forward 1)
```

```
(twist 4 8 11 15)
```

```
(repeat 2
```

```
  (weave-back 4)
```

```
  (twist 8 11)
```

```
  (weave-forward 2)
```

```
  (twist 9 10)
```

```
  (weave-forward 2)
```

```
  (twist 9 10)
```

```
  (weave-back 2)
```

¹⁰ Tradução nossa de *We pursue these questions in the Weaving Codes- Coding Weaves project, by investigating patterns from the perspectives of weaving and music, and by developing a computer language and code for describing the construction of weaves.*

¹¹ Tradução nossa de *Show Us Your Screens*.

¹² Tradução nossa de *Live Algorithm Programming and Temporary Organization for its Promotion*.

```
(twist 9 10)
(weave-back 2)
(twist 8 11)
(weave-forward 4))
```

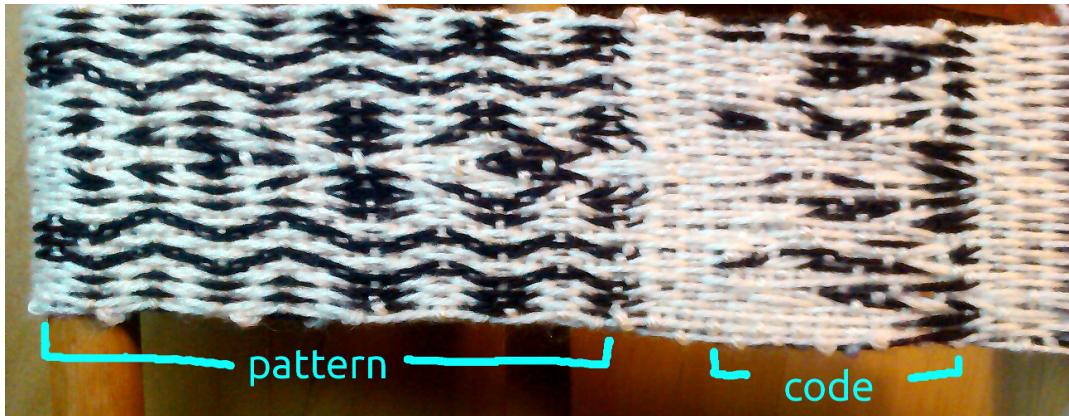


Figura 1 – Tecido resultante da prática *Weaving code*. **Fonte:** Griffths (2015a).

Esta experiência de *weavecoding* pode ser aplicada em uma performance. Griffths ilustra uma performance arquetípica da improvisação de códigos: programadores escrevendo enquanto os resultados são projetados em superfícies planas (ver Figura 2, p. 6). Como veremos, é interessante que o que está sendo escrito também seja projetado (MCLEAN, 2011, p. 129), o que não é o caso desta imagem. A tecelagem é programada por meio de um dispositivo tangível (ver Figura 3, p. 6), uma matriz de botões acopláveis, desenvolvida por Ellen Harlizius-Klück (investigadora da história da matemática, filosofia e tecelagem da Grécia Antiga na Universidade de Copenhague¹³) e Alex McLean (ver Figura 3, p. 6). Imagens em movimento foram projetadas como capturas das atividades têxteis e processadas por Griffths.

“ Uma das idéias originais era combinar tecelagem e codificação em um cenário de atuação, ambos para fornecer uma forma de tornar a codificação ao vivo mais inclusiva com a tecelagem, e ao mesmo tempo esclarecer os processos de pensamentos digitais envolvidos na tecelagem (...) Nossa audiência consistiu de pesquisadores de artesanato, biólogos antropológicos, arquitetos, designers de jogos e tecnólogos – foi mais do que antecipamos! Alex e eu disponibilizamos alguns códigos de música do *slub* para tecer, e minha parte favorita foi projetar a tecelagem ao vivo. ”¹⁴

¹³ Disponível em <<http://www.saumweberei.de/>>

¹⁴ Tradução nossa de *One of the original ideas we had was to combine weaving and coding in a performance setting, to both provide a way to make livecoding more inclusive with weaving, and at the same time to highlight the digital thought processes involved in weaving. (...) Our audience consisted of craft researchers, anthropological biologists, architects, game designers and technologists -- so it all went on quite a lot longer than we anticipated! Alex and I provided some slub livecoded music to weave by, and my favourite part was the live weaving projection. .*



Figura 2 – Performance no Foam Kernow. **Fonte:** Griffths (2015b).

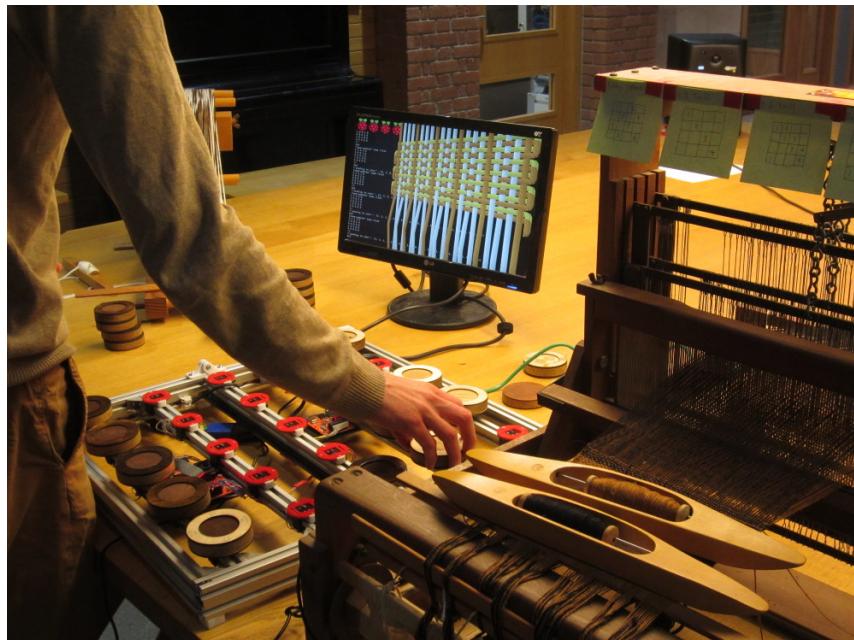


Figura 3 – Alex McLean manipulando uma Matriz de botões para tecelagem, conectado a um Raspberry Pi. **Fonte:** Griffths (2015b).

Esta descrição de Griffths possui uma documentação audiovisual¹⁵. É interessante notar que a menção ao grupo *Slub* auxilia a discussão musical, e ao mesmo tempo articula esta seção com a próxima.

¹⁵ Disponível em <<https://www.youtube.com/watch?v=XrnIVUp9QgM>>

1.1.2.1 Slub

A banda *Slub* começou em 2000, como uma colaboração entre Adrian Ward e Alex McLean. A premissa do duo era utilizar a atividade de programação para realização de uma Música Eletrônica de Dança¹⁶. Sua primeira reunião foi em 2001, no *Paradiso club* em Amsterdã, durante o festival *Sonic Arts*. Em 2005 Griffiths se juntou ao duo durante o festival *Sonar*, o que abriu espaço para o desenvolvimento de uma estética de videogames (MCLEAN, 2011, p. 138–140).

É interessante notar que a improvisação de códigos já era mencionada em documentações de *softwares*, antes do *Slub* ou do manifesto de Ward et al. (2004). No entanto, as metodologias de improvisação de códigos são diversas. A utilização de ambientes como *SuperCollider*, *iXiLang*, PureData ou Max/MSP, estão restritas ao contexto de linguagens de domínios específicos¹⁷. O então duo *Slub* seguiu o seguinte caminho: a utilização de linguagens de propósito geral¹⁸, como Perl, REALBasic e Scheme (ver subseção 3.2.3, p. 58). Além disso, uma descrição destes sistemas GPLs oferecem a descrição de uma característica interessante. Por exemplo, a justificativa de aceitação de uma música para dançar (ver subseção 1.2.1, p. 8) é feita através de uma postura bastante trivial, projetar o que está fazendo no computador; no entanto esta trivialidade não deve ser subestimada, afinal hoje ela é considerada como uma regra heurística. Segundo McLean (2011, p. 139):

“Um sistema Slub antigo é descrito em detalhes por Collins (2003). De maneira breve, ele apresentava um sintetizador, um antigo sistema de codificação ao vivo escrito por Ward, e uma série de programas geradores de batidas e linhas de baixo escritos por McLean. Embora o seu principal objetivo era musical, o Slub gostava de ser confrontado com o desafio de serem aceitos como programadores que fazem música. Para este fim, começou projetando suas telas de audiências com uma sobreposição conceitual, entre seu *softwares* artesanais, e a música que produziam com seu uso.”¹⁹

Em 2004, com a formação do TOPLAP (ver subseção 1.4.1, p. 29), Ward e McLean focaram seus esforços no desenvolvimento de ambientes de improvisação de códigos. É interessante notar que os sistemas elaborados, são uma mistura de linguagens textuais, *patching* e interfaces gráficas de usuário²⁰: “O *Slub* controlava sua música usando interface criadas por e para eles mesmos. Eles variam desde os aparentemente convencionais para

¹⁶ Cf. RIETVELD, 2013

¹⁷ *Domain Specific Language* ou DSL.

¹⁸ *General-Purpose Language* ou GPL.

¹⁹ Tradução nossa de *An early Slub system is described in detail by Collins (2003). In brief it featured a synthesiser and early live coding system written by Ward, and a Number of beat and bass-line generating programs by McLean. Although their primary aim was musical, Slub enjoyed being faced with the challenge of being accepted as programmers who make music. To this end they began projecting their screens audiences with the conceptual overlap between their and-crafted software and the music they produced using it.*

²⁰ *Graphical User Interfaces* ou GUIs.

os abstratos, e das [interfaces] gráficas para a inteiramente textuais.”²¹ (COLLINS, 2003, p. 323). A descrição abaixo detalhada algumas das funções destes programas, bem como um processo de composição por redes que será discutido em outra oportunidade (ver subseção 1.3.3.2, p. 24):

“Por detrás das interfaces *slub* residem os processos ‘compositonais’ ou ‘musicais’ – muitos pedaços de códigos separados, escritos como exploração de idéias musicais. Cada pedaço de código descreve um experimento em áreas como matemática combinatorial, progressões de acordes, modelos sonificados para as pessoas dançarem, métricas que sofrem transformações, batidas sincopadas algorítmicas, e outros. (...) Estes processos compostionais enviam mensagens de um para o outro através de uma rede TCP/IP usando um protocolo de linha de comando. As mensagens viajam através de um servidor central, que administra a sincronização temporal entre os processos *Slub*. (...) O protocolo de rede resolve um problema que poderia, de outra forma, ser insolúvel: Adrian e Alex muitas vezes tomam abordagens muito diferentes para fazer música. Contudo eles não tem que argumentar sobre como a música é feita. Porque eles concordaram sobre, e implementaram um protocolo de rede, eles são livres para fazer música do jeito que gostarem, sabendo que seus programas irão sincronizar um com o outro.”²²

1.2 Dança

A Dança é ilustrada de duas maneiras. A primeira é dança como fim de uma improvisação musical. O código recodificado, e projetado de maneira semelhante ao *Slub*, corre o risco de ser a atração principal. O segundo caso foge do escopo sonoro; uma coreógrafa codifica apenas a orientação espacial de uma bailarina, resultando em uma sensação de quietude sonora. É uma posição que diverge da maioria dos trabalhos apresentados, mas é pouco discutido no âmbito musical.

1.2.1 Algorave

A compositora colombiana Alexandra Cárdenas, em entrevista com Chesire (2013), cita Nick Collins e Alex McLean como os criadores do termo *algorave* (*algorithm + rave*). Surgiu durante uma *gig* (um termo utilizado no início do jazz para caracterizar um

²¹ Tradução nossa de *Slub control their music using user interfaces created by and for themselves. These vary from the apparently conventional to the abstract, and from graphical to entirely textual..*

²² Tradução nossa de *Behind the slub interfaces lie the ‘compositional’ or ‘musical’ processes – many separate pieces of code written as explorations of musical ideas. Each piece of code describes an experiment in such areas as combinatorial mathematics, chordal progressions, sonified models of dancing people, morphing metres, algorithmic breakbeats, and so on. (...) These compositional processes send messages to one another other across a TCP/IP network using a line-based protocol. The messages travel via a central server, which also manages time sync between all the slub processes. (...) The network protocol solves a problem which might otherwise be unsolvable: Adrian and Alex often take very different approaches to making music. However, they don’t have to argue about how the music is made. Because they agreed upon and implemented a network protocol between their programs, they are free to make music however they like, knowing that their programs will synchronise with each other..*

trabalho temporário). Após sintonizarem em uma estação de rádio, decidiram codificar uma música semelhante:

“ Algorave ‘comecou como uma piada’, de acordo com Alex McLean, um pesquisador de música computacional e um dos três de uma banda chamada *Slub*, que têm improvisado códigos por 13 anos. Ele veio com um termo enquanto conduzia uma *gig* em Nottingham com seu amigo Nick Collins (que tocava “datapop” sob o nome Sick Lincoln) no final de 2011. ‘Nós sintonizamos em uma estação pirata tocando *happy hardcore*, e nós pensamos que seria bom programar alguma música *rave*.’ Deste então, McLean organizou oito *algoraves* informais no mundo. ”²³

Em seu artigo “Algorave: Live Performance of Algorithmic Electronic Dance Music”, Collins e McLean (2014, p. 356) sustentam que as estruturas das práticas do *algorave* são anteriores à improvisação de códigos, e já era utilizado na Música Eletrônica para Dançar. O que mantém a relação entre os dois é a prática de projeção do código (ver seção 1.4, p. 27).

“ *Algorave* não é sustentado exclusivamente por *live coders*, mas estes têm mantido uma forte presença em todos os eventos até agora. É assim talvez porque a tradição do *live coding* de projetar telas motiva todo o esforço; onde algoritmos não estão visíveis por períodos de tempo durante uma *algorave*, se corre o risco das coisas parecerem muito como um evento de música eletrônica padrão. ”²⁴

Focando no aspecto histórico, Collins e McLean descrevem uma sequência de eventos (desenvolvimentos de *softwares* e apresentações). 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.”²⁵ 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) cria 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*²⁶, e do *Supercollider*²⁷. Em 2000 o então duo *Slub*, realizam performances, autodenominadas

²³ Tradução nossa de *Algorave "started as a joke", according to Alex McLean, a computer-music researcher and one-third of a band called Slub that's been live coding for 13 years. He came up with the term while driving to a gig in Nottingham with his friend Nick Collins (who plays "datapop" under the name Sick Lincoln) in late 2011. "We tuned into a pirate station playing happy hardcore, and we thought it would be good to program some rave music." Since then, McLean has organised eight informal algoraves around the world.*

²⁴ 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..*

²⁵ Tradução nossa de *an AI based software system that composes music in a variety of popular styles. Disponível em <<http://www.kurzweilai.net/charles-ames>>.*

²⁶ Disponível em <<https://csound.github.io/>>.

²⁷ Disponível em <http://supercollider.sourceforge.net/audiocode-examples>.

generative techno, com abordagem *gabba*. Em 2001 é identificada a utilização de redes neurais para composição de padrões semelhantes ao *drum'n'bass*. Em 2004 é fundado o TOPLAP em uma casa noturna de Hamburgo.

Ilustramos três casos recentes, onde a improvisação de códigos é uma técnica utilizada. Junto com a improvisação de códigos, são utilizados um instrumento eletrônico, voz, e um instrumento elétrico. O inglês Canute, o mexicano Mico Rex e a colombiana residente na Alemanha, Alexandra Cárdenas.



Figura 4 – Performance do duo Canute (Karlesruhe, 2015) **Fonte:** [Canute... \(2015-27-01\)](#).

O registro audiovisual do duo Canute, Matthew Yee-King (bateria eletrônica) e Alex McLean (*laptop*), reforça o arquétipo comentado anteriormente (p. 5). A recomendação “Obscurantismo é perigoso, mostre-nos suas telas” é seguida à risca. Categorizações musicais como *club* e *chordpunch* são mencionados na descrição do vídeo. É curioso notar que, em alguns momentos do vídeo, certas modificações nos códigos causam uma perturbação brusca em sistema de ritmos, percebido através do fluxo musical. Em alguns momentos Yee-King mantém o fluxo, mas em outros o instante musical codificado leva um curto período de tempo para ser sincronizado, o que leva Yee-King a se confundir, e por um breve instante, escutar o código e aí retornar à execução. Essa quebra no fluxo musical pode atrapalhar o fluxo de movimentos do corpo. No final deste capítulo, discutimos que este pode não ser *a priori* um erro do instrumentista, mas sim um problema entre o não-esforço cênico de McLean e o esforço de Yee-King. Esta questão cênica será mencionada na [subseção 1.4.1.1](#).

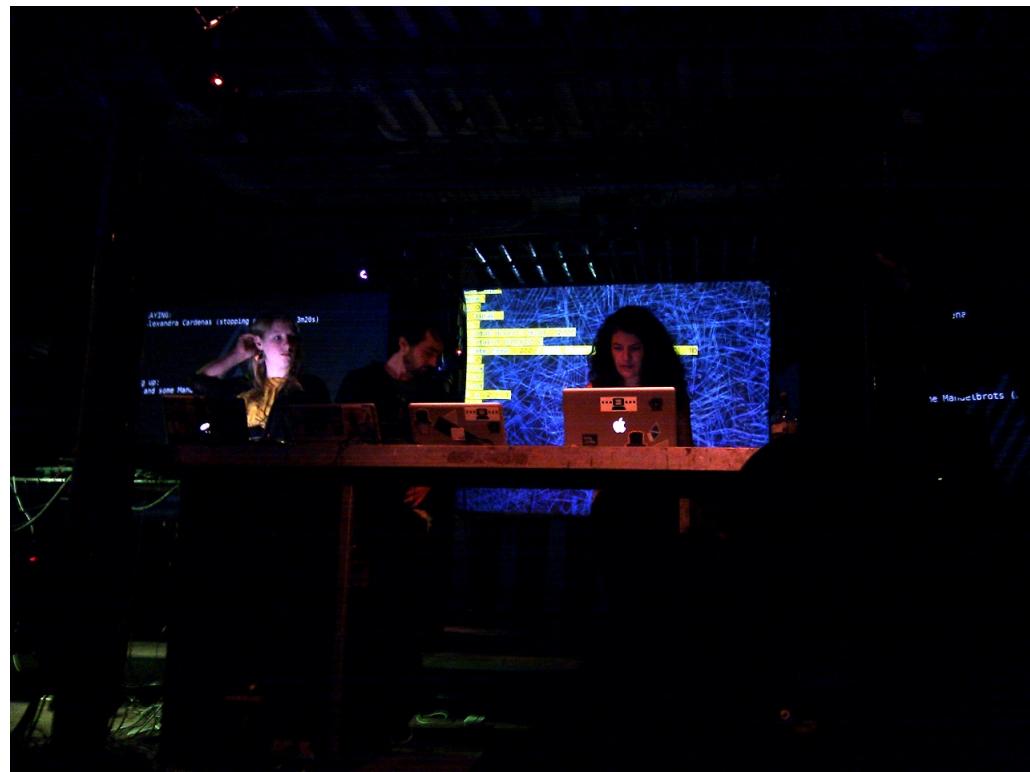


Figura 5 – Performance do duo Mico Rex (Londres, 2013) **Fonte:** Griffiths (2013).

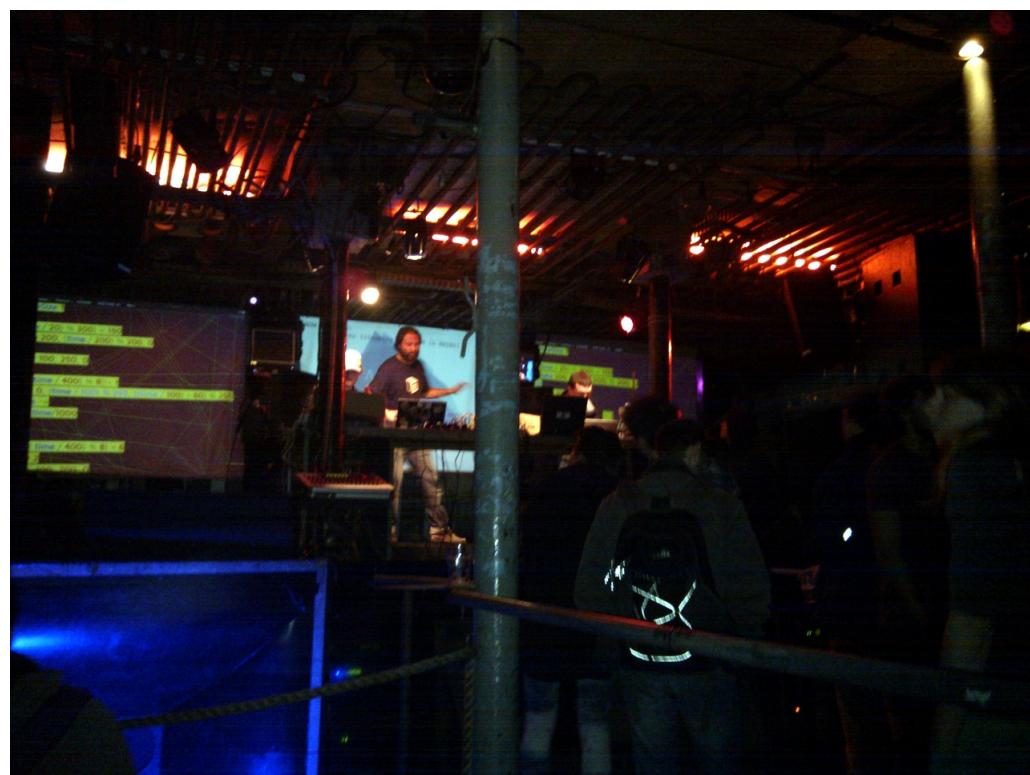


Figura 6 – Performance do duo Mico Rex (Londres, 2013) **Fonte:** Griffiths (2013).

Griffiths registra uma *rave* na embarcação MS Stubnitz, em Canary Wharf, Londres, em 2013. Cárdenas e Ernesto Romero/Jorge Ramírez – Mico Rex, (ver [Figura 6](#), p. 11) –

tocam neste evento. Encontramos um registro audiovisual de curta duração da apresentação do duo Mico Rex²⁸, mas não de Cárdenas. Exemplos sonoros específicos estão disponíveis nas redes sociais *SoundCloud* e *Vimeo*²⁹. Notas descritivas nestes perfis especificam instrumentos e linguagens de programação, mas não o processo criativo. MicoRex utiliza voz e *programação* ao vivo com o ambiente de programação *SuperCollider* (ver seção B.2, p. 86). Entre as categorizações musicais mencionadas, *electro-pop*, *8bits/glitch*, *electro*, *punk*, *bolero* e *breakz*. Cárdenas realiza suas performances com guitarra e programação ao vivo com o *SuperCollider*. É interessante que Cárdenas menciona a utilização de *techno*, *dubstep* e *noise* – além de distorções, utiliza um misto de instrumento expandido (objetos diversos jogados, friccionados, apoiados na guitarra) e retroalimentação de sinais com níveis saturados.

1.2.2 Coreografia

O segundo exemplo de um contexto de dança é projeto de Kate Sichio, *Hacking the Body/Hacking Choreography* (SICCHIO, 2014). Dois exemplos (*Hacking Coreography 2.0 v.01* e *v.02*), mais um, *Hacking The Body 2.0*, serão colocados em discussão para ilustrar a ausência do som na improvisação de códigos (o que não significa necessariamente sensação de quietude sonora).

Para Sichio, a relação entre codificação e a dança é parte de um trabalho contínuo entre notação de coreografias e a improvisação. Diferente das indicações estritas de um pentagrama musical, a partitura de dança oferece mais um guia para os movimentos do corpo. Isto é, a partitura de dança é mais próxima do código de computador do que a partitura musical tradicional. A assimilação de algoritmos computacionais na Dança, ou *Sensibilidades Computacionais*, é mencionada por Sicchio (2014, p. 31), através de Downie (2005, cap. 1, p. 3). São dispositivos metafóricos, elaborados por coreógrafos como Merce Cunningham, Trisha Brown, Bill T. Jones, e William Forsythe: “mecanismos de generalização e abstração, representação da coreografia e dança como computação”³⁰ (DOWNIE, 2005, cap. 1, p. 2–4):

“Esta sensibilidade computacional é presente em dois níveis em um trabalho destes coreógrafos. Primeiramente, em seus processos coreográficos – os sistemas, métodos, e notação através dos quais os coreógrafos criam a dança. Segundo, no trabalho ele mesmo, finalizado, que aparece no palco e é interpretado pelo observador. As primeiras invenções e proclamações de Cunningham – a democracia do espaço do palco, e a redescoberta do que está atrás do dançarino como ponto de origem do movimento – pode ser interpretado como generalizações do tipo; qualquer ponto do palco é a “frente”, e conectado por um conjunto de articulações pode ser pensado

²⁸ Disponível em <<https://vimeo.com/65309754>>

²⁹ <<https://soundcloud.com/tiemposdelruido>> e <<https://soundcloud.com/micorex/>>.

³⁰ Tradução nossa de *mechanisms of generalization and abstraction, choreography as representation, dance as computation*.

como um membro. O que eram constantes, uma vez especificados em uma descrição rígida, se tornam variáveis em uma estrutura gerativa.”³¹

O projeto *Hacking the Body/Hacking Choreography* será descrito a partir de *Hacking Choreography beta v.01*, que vai até *Hacking Choreography beta v.04*, apresentados no *Lincoln Performing Arts Centre* em janeiro de 2012 e o último, em maio de 2013 no *Gnarl Festival* (Inglaterra). A experiência parte de uma primeira versão (*v.01*), um *hackeamento* de uma Partitura de Eventos do artista Alison Knowles (mais especificamente a peça de performance #8, de 1965) . Por exemplo, a partitura é projetada atrás do espaço de performance. Executantes lêem a partitura (ver [Um código-fonte que gera um tecido semelhante à Figura 1. 1.2.2, p. 13](#)),

Exemplo 1.2 (Partitura original de Alison Knowles (1965))

Divida uma variedade de objetos em dois grupos. Cada grupo é rotulado com "tudo". Estes grupos podem incluir diversas pessoas. Existe uma terceira divisão do palco, objetos vazios, rotulados com "nada". Cada um dos objetos é "alguma coisa". Um executante combina e ativa os objetos das seguintes maneiras para qualquer duração desejada de tempo :

- "alguma coisa" com "tudo"
- "alguma coisa" com "nada"
- "alguma coisa" com "alguma coisa"
- "tudo" com "tudo"
- "tudo" com "nada"
- "nada" com "nada"

e seguem com pedaços de papéis, segundo a orientação dada:

“Depois que a partitura foi completada, contudo, ela foi *hackeada*. Isso significa que o executante tenta de alguma forma contornar as instruções originais. Isto foi feito sem preparações prévias e a audiência assistiu isso se desdobrar enquanto era realizada. Nesta primeira performance, o papel e os rótulos foram rasgados para criar novas palavras e categorias

³¹ Tradução nossa de *This computational sensibility is present at two levels in the work of these choreographers. Firstly, in their choreographic processes — the systems, methods, and notations through which the choreographers create the dance. Secondly, in the finished work itself, as it appears on stage and as it is interpreted by the viewer. (...) Cunningham's earliest inventions and proclamations — the democracy of the stage space, and the rediscovery of the dancer's back as a point of origin of motion — can be interpreted as generalizations of a kind; any point of a stage can be a “front”, and any connected set of joints can be thought of as a limb. What were once specified constants in a rigid description become variables in a generative framework..*

(...) Então ao invés de “nada”[Nothing], foram formados dois grupos, “não”[No] e “coisa”[Thing].”³²

A segunda experiência, *Hacking Coreography v.02*, é inspirada na proposta de definir termos e associar uma ordem ao termo (como no exemplo anterior). Mas dessa vez, as orientações são escritas como um híbrido de texto discursivo, legível por um executante, e de código de computador em linguagem Java. Isto é, ele não é executável por um computador para resultar em sons, mas por um humano para resultar em movimentos. Isto é, uma seção *set up* define as posições iniciais, *movement* define os tipos de movimentos que serão executados por intérpretes, e *coreography* define uma estrutura de fluxo destes movimentos, e por último, uma ordem de execuções. É interessante notar que Sichio aponta para um outro *hackeamento* da partitura: a utilização de números dificultou a leitura dos intérpretes. Uma alteração na função *coreography* foi feita pelos próprios intérpretes, para alterar a notação numérica por uma descrição textual da ação.

Exemplo 1.3 (Exemplo de um hackeamento de partitura de movimentos)

```
/Dance/
set up()
{
dance a centre, right
dance b centre, left
}

movement()
{
move1 (dance a = rotate) (dance b = jump)
move2 (dance a = brush) (dance b = lie down)
move3 (dance a = push) (dance b = run)
move4 (dance a = step) (dance b = kneel)
}

coreography()
{
if (dancer a = rotate right 180)
then both jump = 2 feet to 1
if (dancer b = travels)
then brush = right foot
}

run(){
move1
move4
move4
move1
```

³² Tradução nossa de *After the score was completed, however, it was then hacked. This meant that the performer had to try to somehow circumvent the original instructions. This was done with no previous preparation and the audience watched this unfold as the piece was performed. In this first performance, the paper and the labels were torn up to create new words and categories (...). So instead of “Nothing” there were two new groups, “No” and “Thing.”*

```

move2
move3
move1
move2
move3
move4
}

/hack/
{
if (dancer a = kneel)
dancer a = kneel
if (dancer a = rotate)
dancer b = rotate opposite direction
}

```

O *Hacking The Body 2.0*, ou *HTB2.0* (2015)³³ é uma performance mais formal. Se caracteriza por uma apresentação em um evento internacional, com um o palco-arena que reforça a hierarquia ator-público (o palco fica aproximadamente na linha da cabeça do espectador). A coreógrafa está sentada ao lado direito do palco, visível, mas debaixo de uma penumbra. Já a dançarina, cuja vestimenta branca equilibra com uma iluminação frontal e estática, expressa uma face de seriedade (ver Figura 7, p. 16). A performance, segundo A coreografia segue uma partitura de movimentos corporais. Mas esta partitura não é visual (o registro não é feito em papel), o resultado da apresentação não são movimentos corporais pré-definidos, e tão pouco existe um acompanhamento musical. Esta partitura é um correlato acústico do tato, codificada por uma coreógrafa. Afastada de uma bailarina, controla a direção dos movimentos corpóreos da performance:

“Esta peça é uma exploração de eletrônica codificada ao vivo e movimentos improvisados. Uma dançarina veste uma peça de atuadores hapticos. Estes atuadores são programados em tempo-real via OSC³⁴ para ‘zunir’ sobre os lados direito e esquerdo da dançarina para indicar qual lado do corpo a dançarina deve mover. A partitura é codificada ao vivo pela coreógrafa enquanto a dançarina responde por uma retroalimentação haptica. Esta peça explora o *live coding* de corpos, e movimento como saída, ao invés de saídas sonoras ou visuais como encontrado em muitas execuções de *live coding* ”³⁵

³³ Disponível em <<https://www.youtube.com/watch?v=iOAffWTBVE0>>

³⁴ N.A.: “Open Sound Control é um protocolo de comunicação entre computadores, sintetizadores sonoros e outros dispositivos multimídia que são otimizados para as modernas tecnologias de rede”. Disponível em <<http://opensoundcontrol.org/introduction-osc>>

³⁵ Tradução nossa de *This dance piece is an exploration of live coded electronics and improvisational movement. A dancer wears a custom garment of haptic actuators. These actuators are programmed real-time via OSC to ‘buzz’ on the right and left sides of the dancer to indicate which side of the body the dancer will move. The score is being live coded by choreographer while the dancer is responding to the haptic feedback. This piece explores live coding of bodies and movement as output rather than a sonic or visual output as found in many live coding performances.* Disponível em <<http://iclc.livedcodenetwork.org/performances.html>> .



Figura 7 – Dançarina (anônima) controlada por Kate Sicchio (2015) através de uma codificação improvisada. **Fonte:** <<https://www.youtube.com/watch?v=uAq4BAbvRS4>>.

É interessante notar uma sensação de quietude sonora³⁶ na performance, mas pouco discutida do ponto de vista musical. Como aponta a própria coreógrafa, a maioria das performances de improviso de códigos segue o seguinte procedimento: o código é criado, e um som, uma nota, uma imagem ou um vídeo são gerados, combinados, transformados de maneira contínua. Mas o padrão é a realização audiovisual. Mesmo em algumas performances de dança pesquisadas (e que não foram mencionadas neste documento), a dança e a projeção audiovisual se suportam. A criatividade deste trabalho toca um conceito técnico fundamental da computação: qual é o *dispositivo de entrada e de saída* praticado nas improvisações de códigos? Sicchio responde que o corpo já é um dispositivo de entrada e saída de interações sociais e pode ser controlado por outro humano através de comandos de rede. A sensação de quietude é presente não por questões musicais, mas é definido por uma prática de dança.

1.3 Música computacional

Apresentamos dois interessantes exemplos realizados com o *SuperCollider*. O primeiro é um ambiente de programação recorrente no ambiente acadêmico (ver [subseção 1.3.1](#), p. 17). A segunda será uma performance telepresencial (ver [subseção 1.3.2](#), p. 21). Por último, apresentamos uma perspectiva histórica da improvisação de códigos (ver [subse-](#)

³⁶ Cf. KOELLREUTTER; HAOULI; MANNIS, ,

ção 1.3.3, p. 22). Isto é, Ward et al. (2004) mencionam uma proto-história da improvisação de códigos, ou eventos artísticos relacionados à *Live Computer Music*.

1.3.1 Música de concerto e audiovisual

screenBashing de Magno Caliman (ver Figura 8) foi realizado durante o XIII ENCUN³⁷. Será útil para exemplificar uma “(...) performance de *livecoding* arquetípica [que] envolve programadores escrevendo códigos no palco, com suas telas projetadas para a audiência.”³⁸ (McLean; WIGGINS, 2010b, p. 1). No entanto, a tradução “improviso de códigos” como correlato de *livecoding*, bem como a própria definição de ?? (ver Capítulo 1, p. 1), se tornam problemáticas em *screenBashing*. Segundo o próprio compositor³⁹:

Não acho totalmente adequado considerar como puramente um improviso. Existe, desde a concepção da peça, uma intenção de composição ali. Composição no sentido de agenciamento de materiais *a priori*, entende? Claro, existe um nível de abertura ali com relação a, por exemplo, os caracteres que eu uso no loop, ou nos valores dos parâmetros do *Supercollider* (...). Existem sessões formais ali que, se eu tocar a peça novamente, vão se manter. Daí vc já percebe uma determinação no âmbito macro da coisa também. Em uma situação limite, eu consigo imaginar até uma partitura (de execução) para essa peça, que permitiria que outra pessoa tocasse, sem ver o vídeo da minha performance, e chegasse em resultados bastante similares. Então eu vejo uma série de características aí que diferem de um improviso, onde se espera que uma parte grande do processo de tomada de decisão seja feita no momento da performance. Claro que essa distinção entre composição/peça/determinação e performance/improviso/indeterminação é complexa e a discussão foge totalmente do escopo do seu trabalho. Mas só para atentar para esse detalhe, acredito que se imaginarmos um *continuum*, uma linha onde de um lado vc tem uma improvisação totalmente livre (impossível de se alcançar, claro) e do outro uma composição 100% determinada (tão impossível quanto), acredito que *screenBashing* está posicionada mais à direita...

A performance consiste no seguinte: Caliman senta-se ao computador, lateralmente à tela de projeção, com uma iluminação de penumbra. O projetor expõe o estado atual de seu *laptop*, que apresenta um editor de texto. O executante começa a programar em linguagem C (ver exemplo abaixo). Um pequeno laço iterativo (*for loop*) repete caracteres diversos, improvisados. Assim que termina de escrever, abre um console (ou terminal nos sistemas operacionais Unix) e compila o programa – a compilação é um processo no qual a representação textual humana, definida pela ANSI⁴⁰, é convertida códigos binários. Este processo não demora, e o programa é executado. O resultado é uma sequência de caracteres de texto como texturas visuais. É importante lembrar que Caliman utiliza um recurso

³⁷ Encontro Nacional de Compositores Universitários em Campinas-SP no ano de 2015.

³⁸ Tradução nossa de *The archetypal live coding performance involves programmers writing code on stage, with their screens projected for an audience..*

³⁹ Comunicação pessoal

⁴⁰ Disponível em <<http://www.ansi.org/>>

técnico do sistema operacional, que torna o terminal semi-transparente, o que permite sobreposições de texturas. A transformação da textura visual é levada a cabo através da modificação do argumento de entrada da função `printf` (caracteres entre aspas).



Figura 8 – Performance de *screenBashing*. Fonte: <<https://vimeo.com/148626379>>.

Exemplo 1.4 (Algoritmo inicial de Magno calimann)

```
# include <stdio.h>

int main()
{
    double i;

    for (int i=0; i<; i < 6666; i+=0.999)
    {
        printf("\\\\\" // \\\"");
        fflush(stdout)
        system("sleep 0.0006")
    }
}
```

Execução:

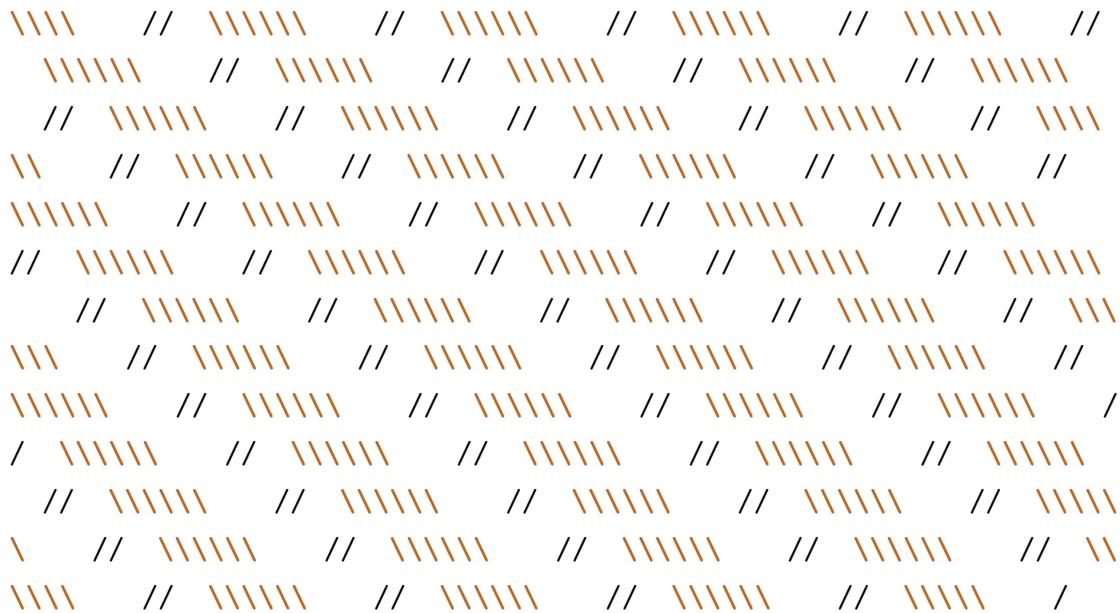
```
# gcc - compilador
# a.c - arquivo em linguagem c
```

```
# -o - escreve o resultado em um outro arquivo
# a - arquivo binario alvo
gcc a.c -o a
```

Este código gera algumas mensagens classificadas como *warnings*, ou erros de lógica, que não atrapalham a execução do código em pequenos programas.

```
magno.c: In function 'main':
magno.c:7:12: error: conflicting types for 'i'
```

Resultado:



Uma análise atenta do algoritmo visual pode ultrapassar os aspectos técnicos e apontar ideologias. Este é um campo com poucos estudos e não será feito neste trabalho. Limitamo-nos a comentar a experiência auditiva, e visual, contrapondo-a com a declaração do compositor. Para o autor deste texto, a repetição dos algarismos seis e nove, seria uma menção direta (e espelhada) de signos do *black metal* abstrato. Ou seja, um processo sincrético entre a imagem, o código (que de certa forma, faz parte da imagem) e o som (explicado adiante) é notável. No entanto o compositor declara que esta foi uma decisão tomada na hora, como uma piada, embora tenha alguma relação. Mas isso não quer dizer que, para o compositor, a sonoridade se configura como *black metal*. Objetamos que a questão do gênero ultrapassa questões de gosto ou prática sonora, se baseando mais em costumes sociais, como o que se veste, o que se fala, o que se bebe, etc⁴¹. Por outro lado,

⁴¹ Cf. JR.; Sá, 2003, 2009

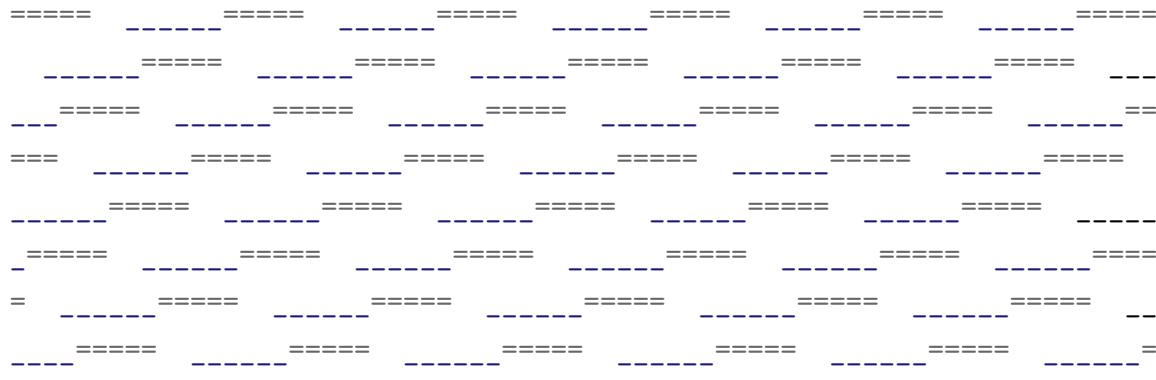
uma segunda notação mais enxuta e sem erros, permite apontar a questão:

Exemplo 1.5 (Correção ao algoritmo de Magno)

O início do código anterior pode ser corrigido. Os novos caracteres foram inseridos por Caliman em 2'18":

```
# include <stdio.h>
int main(){
    int i=0;
    for (; ; i++){
        printf("===== -----");
        fflush(stdout);
        // 30 frames por segundo (1/30)
        system("sleep 0.033");
        // 3 frames por segundo (1/3)
        // system("sleep 0.33")
    }
}
```

Resultado:



Os caracteres \\\\" // \\ e ===== -----, assim como outros, são sobrepostos. Esse processo ocorre com uma sensação de quietude sonora (ainda assim, este aspecto não é explorado musicalmente, mas essa não é a intenção da música). O improvisador abre códigos preparados no ambiente de programação *SuperCollider*⁴², que realiza a síntese de texturas sonoras ruidosas. Segundo o compositor, foram *tweets* encontrado na *internet*⁴³. Um novo som ruidoso é adicionado para cada nova textura visual. Este processo, de

⁴² Disponível em <<https://supercollider.github.io>>

⁴³ Disponível em <<https://twitter.com/sc140tweets>>

alternar a codificação da textura visual, e depois executar um programa no *SuperCollider*, é repetido até o fim da peça. A acumulação de sons e caracteres em movimento, cria um contínuo audiovisual que em um momento, as texturas visuais e sonoras entram se tornam um mesmo objeto audiovisual. O processo de compilações e execuções sucessivas exigem cada vez mais e mais do processamento do computador. No auge de saturação visual, sonora e de memória do computador, a improvisação finaliza com um silêncio brusco, decorrente do congelamento do sistema operacional. Essa característica, de pane sistêmica será revista (ver subseção 1.3.3.3, p. 26).

1.3.2 Telepresença e espaços virtuais

Uma performance virtual de *live coding* é aquela em que dois ou mais executantes, em endereços diferentes de uma rede de computadores. Isso situa três casos, do qual especificaremos um: i) uma rede local, com computadores diferentes, mas com os improvisadores fisicamente próximos; ii) uma rede remota, privada, que comunica um conjunto de pessoas fisicamente distantes; iii) a rede mundial de computadores, onde o navegador se torna o ambiente virtual de criação musical (ROBERTS; WAKEFIELD; WRIGHT, 2013). Em ambos, a premissa é compartilhar o mesmo código. Podem também compartilhar do mesmo som, mas isso depende da implementação técnica.

O primeiro caso é bastante documentado nas *live coding sessions*, sessões de improvisações que ocorrem em encontros, simpósios e *workshops*, ricamente documentadas⁴⁴.

Um exemplo do segundo caso é descrito por Junior, Lee e Essl (2015, p. 152–153). Uma performance telepresencial em 2014, por Ben Swift, Henry Gardner e Andrew Sorensen, realizada “entre dois intérpretes-programadores localizados na Alemanha e Estados Unidos usando um servidor SSH localizado na Austrália”⁴⁵. Junior, Lee e Essl ainda descrevem o desenvolvimento de um ambiente chamado *Supercopair*⁴⁶, um ambiente cooperativo de improvisação de códigos, possível através de serviços de computação em nuvem.

Execuções remotas assíncronas (isto é, entre duas ou mais pessoas, em tempos diferentes) estão sendo exploradas a partir do uso musical de um navegador de *internet*. São realizadas em *apps* como *Gibber* (ROBERTS; KUCHERA-MORIN, 2012)⁴⁷, *Wavepot*⁴⁸ e *Vivace* (VIEIRA et al., 2015), através da biblioteca *WebAudio API*⁴⁹. Um *software* na mesma direção foi desenvolvido, uma colaboração entre o autor deste documento e um

⁴⁴ Disponível em <<https://supercollider.github.io/archive>>

⁴⁵ Tradução nossa de *between two live coders located in Germany and United States using an SSH server located in Australia..*

⁴⁶ Disponível em <<https://github.com/deusanyjunior/atom-supercopair>>

⁴⁷ Disponível em <<http://gibber.mat.ucsb.edu/>>

⁴⁸ Disponível em <<http://www.wavepot.com>>

⁴⁹ Disponível em <<https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html>>

membro da banca⁵⁰.

1.3.3 Proto-história musical

Esta seção será dedicada à construção de um espaço conceitual histórico, do ponto de vista musical. Isto é, aqueles exemplos citados como “proto-históricos” que possuem alguma similaridade com o conjunto de regras práticas publicadas por Ward et al. (ver seção 1.4, p. 27), cujo objetivo é a criação sonora.

Mori (2015b) descreve um caso prematuro de *live coding* na Itália, com o compositor Pietro Grossi (ver subseção 1.3.3.1, p. 22). Divergente em algumas das propostas de Max Mathews, sacrificou a questão timbrística para trabalhar na questão performática. As atividades de grupos como *The Hub*, e compositores como Ron Kuivila, no final da década de setenta e anos oitenta, são descritos como fundamentais para o entendimento histórico do *live coding* em seus aspectos técnicos (ver subseção 1.3.3.2, p. 24). Um recorte do documento-manifesto “*Live Algorithm Programming and Temporary Organization for its Promotion*”, de McLean e Wiggins, será feita na seção 1.4 para discutir identidade cultural da organização TOPLAP. Na subseção 1.4.1.1, “Show us your screens”, é revisto como o manifesto que define as regras práticas do *live coding*, e a ideologia de projeção de telas.

1.3.3.1 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 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, p. 126), ao usar este programa,

“(…) o intérprete era capaz de produzir e reproduzir música em tempo real, digitando alguns comandos específicos e os parâmetros composticionais desejados. O som resultante vinha imediatamente depois da operação de decisão, sem qualquer atraso causado por cálculos. Havia muitas escolhas de reprodução no programa: era possível salvar na memória do computador peças de músicas pré-existentes, para elaborar qualquer material sonoro no disco rígido, para administrar arquivos musicais e iniciar um processo de composição automática, baseado em algoritmos que trabalham com procedimentos “pseudo-casuais”. Existia também uma abundância de escolhas para mudanças na estrutura da peça. Um dos mais importantes aspectos do trabalho de Grossi foi que todas intervenções eram instantâneas: o operador 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

⁵⁰ Cf. LUNHANI; SCHIAVONI, 2015

desta forma, perdendo muito no lado da qualidade sonora. Seu desejo era poder escutar os sons resultantes imediatamente.”⁵¹

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.”⁵². Neste sentido, defendia que o desenvolvimento de novos timbres gerados por computador deveria esperar por melhores implementações de *hardware*.

O disco “*GE-115 - Computer Concerto*” (1967)⁵³, contém gravações de peças executadas em um computador fabricado pela *General Eletrics*. Utilizam apenas uma forma de onda quadrada (pulsos) como timbre. Todas transformações sonoras derivam da soma instantânea das notas tocadas. Enquanto algumas peças são transcrições da “*Oferenda Musical*” de J.S.Bach, três peças originais estão incluídas: “*Mixed Paganini*”⁵⁴, “*Permutations Of Five Sounds*”⁵⁵ e “*Continuous*”⁵⁶.

Por outro lado, também é importante lembrar que Grossi também contribui para performances remotas. Mori descreve uma transmissão por telefone, subsidiada por uma companhia italiana, entre duas cidades, durante uma conferência de tecnologia. Esta transmissão foi o estímulo inicial para o desenvolvimento dos softwares *TAU2* e *TELETAU*. O software era executado em um “um computador conectado à rede de computadores BITNET, para explorar remotamente os recursos de cálculo do computador CNR e imediatamente escutar os resultados sonoros produzidos pelo TAU2.”⁵⁷

“[Pietro] Grossi fez sua primeira experiência do tipo durante uma conferência de tecnologia em Rimini em 1970, onde o músico reproduzia algumas de suas composições, bem como sons randômicos, empregando um terminal de vídeo conectado pelo telefone para o computador da

⁵¹ 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..*

⁵² 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.*

⁵³ Disponível em <<http://www.discogs.com/Studio-Di-Fonologia-Musicale-Di-Firenze-GE-115-Computer-Concerto/release/575632>>.

⁵⁴ Disponível em <https://www.youtube.com/watch?v=ZQSP_wF7wSY>.

⁵⁵ Disponível em <<https://www.youtube.com/watch?v=m0WVLJ2LxeY>>.

⁵⁶ Disponível em <https://www.youtube.com/watch?v=bf8jMA_zizc>.

⁵⁷ Tradução nossa de *a computer connected to the BITNET computer network, to exploit remotely the CNR computer calculation resources and to immediately listen the sound results produced by TAU2..*

CNR em Pisa. A RAI, empresa de radiodifusão italiana, emprestou suas pontes de rádio [Comunicação entre duas antenas] para enviar sinais sonoros entre Pisa e Rimini. É como se fosse o primeiro experimento de telemática musical no mundo.”⁵⁸

A partir destas abordagens de Grossi, pontuamos o conceito de *reflexividade*, ou a “habilidade de um programa manipular como dados algo que representa o estado do programa durante sua própria execução, o mecanismo para codificação de estados de execução é chamado *reificação*. (MALENFANT; JACQUES; DEMERS, 1996, p. 1)”⁵⁹.

1.3.3.2 Baía de São Francisco

Com o florescimento da indústria de computadores pessoais na Baía de São Francisco, 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-restricção, barulho, e improvisações com bordas que sobraram das revoluções contra-culturais dos anos 60(BROWN; BISCHOF, 2002, online)⁶⁰.

Na segunda metade da década de setenta, Jim Horton começou a adquirir micro-controladores KIM-1⁶¹. Segundo Brown e Bischof, não demorou para que outros interessados comprassem. Discussões informais posteriores, que incluiam, além de Horton, David

⁵⁸ Tradução nossa de *Grossi made his first experience of this kind during a conference on technology in Rimini in 1970, where the musician reproduced many of his compositions and random sounds as well, by employing a video terminal connected via telephone to the CNR's computer in Pisa. RAI, the Italian public broadcasting company, lent its powerful FM radio bridges to send back sound signals from Pisa to Rimini. It is likely to be the first official experiment of musical telematics in the world.* in Giomi, Francesco, Ligabue, Marco 1999. *L'istante zero. Conversazioni e riflessioni con Pietro Grossi, Firenze, Sismel Edizioni del Galluzzo*.

⁵⁹ 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..*

⁶⁰ 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.*

⁶¹ Disponível em <<http://www.6502.org/trainers/buildkim/kim.htm>>.

(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, 2002, online)”⁶².

Em 1978, Bischof, Behrman, Gold e Horton gravaram um *Extended Play* (EP)⁶³ a partir de uma performance. O disco foi lançado 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, 2002, online)⁶⁴

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

⁶² Tradução nossa de *the occasional tones of my machine caused Jim's machine to transpose its melodic activity according to my "key" note.* .

⁶³ Gravação muito longa para um *demo* e insuficiente para um disco, de vinil na época.

⁶⁴ 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 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.*

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 interagiam, cada uma enviando e recebendo dados pertinentes para o funcionamento musical(BROWN; BISCHOF, 2002, online)⁶⁵.

1.3.3.3 Ron Kuivila

McLean e Wiggins (2009) comentam a performance *Water Surfaces*, realizada na edição de 1985 da STEIM⁶⁶, em Amsterdã, como significativa para a concepção de uma improvisação de códigos (excluindo a tecnologia de projeção visual). A performance chamou a atenção, e foi incuída na primeira faixa do disco “*TOPLAP001 - A prehistory of live coding*”, como uma reconstrução da peça, 2007⁶⁷; uma nota sobre a performance descreve o seguinte: “ Esta obra usou programação FORTH ao vivo; Curtis Roads (1986) testemunhou e relatou a performance de Ron Kuivila feita na STEIM em Amsterdã, em 1985; a performance original termina com a quebra do sistema... ”⁶⁸

Ronald Kuivila programou um computador Apple II no palco para cirar sons densos, rodopiantes e métricos, disposto em camadas e dobravam sobre si. Considerando o equipamento usado, os sons eram surpreendentemente grandes em escala. Kuivila teve problemas em controlar a peça devido q problemas sistêmicos. Ele finalmente entrou em dificuldades técnicas e finalizou a performance(ROADS, 1986, p. 47)⁶⁹.

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

⁶⁵ 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.*

⁶⁶ *SStudio for Electro-Instrumental Music*, disponível em <<http://stein.org/about/>>.

⁶⁷ Disponível em <http://toplaph.org/wiki/TOPLAP_CDs>.

⁶⁸ 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....*

⁶⁹ Tradução de *Ronald Kuivila programmed an Apple II computeronstage to create dense, whirling, metric sounds that layered in and folded over each other. Considering the equipment used, the sounds were often surprisingly gigantic in scale. Kuivila had trouble controlling the piece due to system problems. He finally gave in to technical difficulties and ended the performance*

Amsterdã em 1985, mas antes disso, não fez uma música muito interessante. A performance consistiu de digitação”⁷⁰

Nenhuma fonte sonora foi encontrada disponível online.

1.4 LAPTOP

“*Live Algorithm Programming and Temporary Organization for its Promotion*” (WARD et al., 2004; BLACKWELL; COLLINS, 2005) é um primeiro documento-manifesto sobre o *live coding* como modalidade artística, e de suas regras práticas. O seu acrônimo LAPTOP representa o principal equipamento técnico utilizado. Este manifesto expõe o ambiente de performance característico do *algorave* e um suporte ideológico para o *Code DJing*. Ritos técnicos do improvisador, como por exemplo, a projeção do código, são justificados através do discurso de transparência e provável colaboração entre intérprete e público:

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 do algoritmo é o que importa. Outro experimento mental pode ser encarado com um DJ ao vivo codificando e escrevendo uma lista de instruções para o seu *set* (feito com o iTunes, mas aparelhos reais funcionam igualmente bem). Eles passam ao HDJ [*Headphone Disk Jockey*] de acordo com este conjunto de instruções, mas no meio do caminho modificam a lista. A lista está em um retroprojetor para que o público possa acompanhar a tomada de decisão e tentar obter um melhor acesso ao processo de pensamento do compositor. (WARD et al., 2004, p. 245) ⁷¹

Adiante podemos ver outros dois conceitos aglutinados: a Música de Processos., e a Música Generativa:

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

⁷⁰ 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..*

⁷¹ Tradução nossa de: *Live coding allows the exploration of abstract algorithm spaces as an intellectual improvisation. As an intellectual activity it may be collaborative. Coding and theorising may be a social act. If there is an audience, revealing, provoking and challenging them with the bare bone mathematics can hopefully make them follow along or even take part in the expedition. These issues are in some ways independent of the computer, when it is the appreciation and exploration of algorithm that matters. Another thought experiment can be envisaged in which a live coding DJ writes down an instruction list for their set (performed with iTunes, but real decks would do equally well). They proceed to HDJ according to this instruction set, but halfway through they modify the list. The list is on an overhead projector so the audience can follow the decision making and try to get better access to the composer's thought process.*

Neste ponto, os músicos podem ser pensados como programadores explorando seu código manifestado como som. Isso não reduz seu papel principal como um músico, mas complementa, com a perspectiva única na composição de sua música. Termos como “música generativa” e “música de processos” tem sido inventados e apropriados para descrever esta nova perspectiva de composição. Muita coisa é feita das supostas propriedades da chamada “música generativa” que separa o compositor do resultado do seu trabalho. Brian Eno compara o fazer da música generativa com o semear de sementes que são deixadas para crescer, e sugere abrir mão do controle dos nossos processos, deixando eles “brincarem ao vento”.⁷²

Se por um lado, a Música como um Processo Gradual⁷³ e a Música Generativa são referenciais possíveis na improvisação de códigos, essa não é a questão inicial. A ligação conceitual do *live coding* com a Música de Processos, e Música Generativa é relativa ao uso de algoritmos, mas não ao resultado sonoro como processo de escuta. Por exemplo, uma abordagem sobre a Música de Processos é apresentada por Mailman (2013, p. 128), e descreve a Música Minimalista de Processos como uma Música de Algoritmos Simples, um processo determinístico que age sobre focos de quadros temporais. Já a “Música Generativa é sensitiva às circunstâncias, isso quer dizer que irá reagir diferentemente dependendo das suas condições iniciais, onde ocorre e assim por diante.”⁷⁴(ENO, 1996). McLean (2011, p. 130) problematiza o processo na improvisação de códigos da seguinte forma:

“Na codificação ao vivo a performance é o processo de desenvolvimento de *software*, em vez de seu resultado. O trabalho não é gerado por um programa acabado, mas através de sua jornada de desenvolvimento do nada para um algoritmo complexo, gerando mudanças contínuas da forma musical ou visual ao longo do caminho. Isto contrasta com a arte generativa popularizada pela música geradora de Brian Eno (1996). (...) O resultado segue mais ou menos o mesmo estilo, com apenas algumas permutações, dando uma idéia das qualidades da peça. Isto é bem ilustrado pelo nosso estudo de caso de um artista-programador, que executa seu programa poucas vezes não para produzir novas obras, mas para obter diferentes perspectivas sobre o mesmo trabalho.”⁷⁵

⁷² 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”.*

⁷³ Cf. REICH, 1968

⁷⁴ Tradução nossa de *Generative music is sensitive to circumstances, that is to say it will react differently depending on its initial condition, on where it's happening and so on..*

⁷⁵ Tradução nossa de *In live coding the performance is the process of software development, rather than its outcome. The work is not generated by a finished program, but through its journey of development from nothing to a complex algorithm, generating continuously changing musical or visual form along the way. This is by contrast to generative art popularised by the generative music of Brian Eno (1996) (...) Output more or less follows the same style, with only a few permutations giving an idea of the*

1.4.1 TOPLAP

Uma permutação na ordem das letras do acrônimo LAPTOP dá origem ao acrônimo TOPLAP. Ward et al. (2004, p. 246) e Ramsay (2010) apontam que este acrônimo dinâmico; isto quer dizer que as primeira, terceira e quinta letras possuem diversos significados (ver Figura 9):

“A organização TOPLAP (www.toplap.org), cuja sigla possui diversas interpretações, uma sendo *Organização Temporária para a Proliferação da Programação de Algoritmos Ao Vivo*, foi criada para promover e explorar o *live coding*. TOPLAP nasceu em um bar enfumaçada em Hamburgo à uma da manhã em 15 de Fevereiro de 2004.”⁷⁶

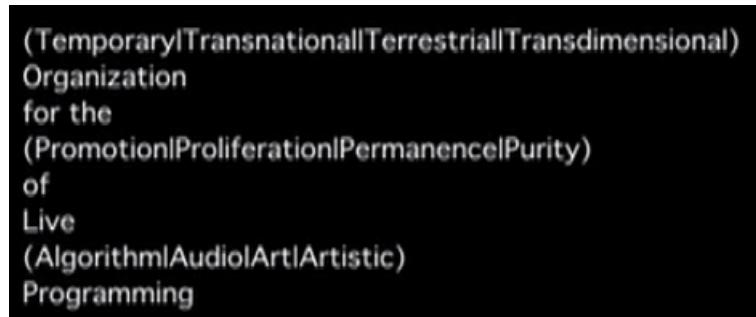


Figura 9 – Definição do significado de TOPLAP. Fonte: Ramsay (2010).

O símbolo “|” é uma representação gráfica do operador lógico *OR* (OU), bastante utilizado em algoritmos condicionais. Isto é, *Temporary | Transnational | Terrestrial | Transdimensional* significa que as letras ímpares “T”, e “P” e “A”, podem significar um ou outro termo indicado pelo algoritmo.

Este comportamento, de permutar ordem das letras é praticado por Nick Collins (1975-); a permutação de suas letras é utilizada pelo pesquisador para gerar pseudônimos como Click Nilson, ou Sick Lincoln. Isso transparece uma técnica de uso frequente na improvisação de códigos, provavelmente pela facilidade de sua implementação computacional em amb. Por exemplo, o *SuperCollider* oferece um método chamado *scramble*, que embaralha a ordem de um conjunto (de caracteres). Mais especificamente, a permutação de letras transparece uma reorganização gramática, mas que também reflete em técnicas de reorganização algorítmica da gramática musical.

qualities of the piece. This is well illustrated by our case study of an artist-programmer, who ran their program a few time not to produce new works, but to get different perspectives on the same work.

⁷⁶ Tradução nossa de *The organisation TOPLAP (www.toplap.org), whose acronym has a number of interpretations, one being the Temporary Organisation for the Proliferation for Live Algorithm Programming, has been set up to promote and explore live coding. TOPLAP was born in a smoky Hamburg bar at 1am on Sunday 15th February 2004.*

1.4.1.1 Show us your screens

Além das performances inaugurais nos festivais Europeus, o manifesto Lubeck04, “iniciado em um ônibus de trânsito Ryanair⁷⁷, em Hamburgo, para o aeroporto Lübeck(WARD et al., 2004, p. 247)”⁷⁸, mais conhecido como “*Show us your screens*”, prescreve algumas regras práticas 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.
- 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/impresiva 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.⁷⁹

⁷⁷ Disponível em <<https://www.ryanair.com/pt/pt/>>

⁷⁸ Tradução nossa de *begun on a Ryanair transit bus from Hamburg to Lubeck airport*.

⁷⁹ WARD et al., 2004, loc. cit.. 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,

Escolhemos dois pontos de interesse ; as frase “Obscurantismo é perigoso. Mostre-nos suas telas” e “Algoritmos são pensamentos, motosserras são ferramentas”, foi muito discutida no processo de qualificação desta tese. O primeiro questiona: é realmente necessário a projeção dos códigos para a questão da performance, do ponto de vista musical/cênico?

1.4.1.2 Obscurantismo é perigoso, mostre-nos suas telas

O manifesto acima surgiu, entre outros motivos, como uma resposta ao artigo “*Using Contemporary Technology in Live Performance; the Dilemma of the Performer*” (SCHLOSS, 2003). A crítica principal de Ward et al. refere-se ao sétimo dos questionamentos sugeridos para uma performance de improvisação ao vivo com computadores. Isto é, em um contexto de embate acadêmico, o desafio colocado por Schloss (2003, p. 241) foi um estímulo considerável para emancipação da improvisação de códigos. É curioso notar que o problema e a intenção de Schloss eram opostas ao que foi proposto por Ward et al.:

“Para reiterar, agora que nós temos computadores rápidos o suficiente para execução ao vivo, nós temos novas possibilidades, e um novo problema. Do começo da evidência arqueológica da música até agora, música era tocada acusticamente, e sempre foi fisicamente evidente como o som era produzido; ali existia uma relação de proximidade entre gesto e resultado. Agora nós não temos mais que seguir as leis da física (ultimamente temos, mas não nos termos do que o observador vê), uma vez que nós temos completo poder do computador como intérprete e intermediário entre nosso corpo físico e o som produzido. **Por esta causa, a ligação entre gesto e resultado foi completamente perdido, se é que existe ligação. Isto significa que nós podemos ir além da relação de causa-e-efeito entre executante e instrumento que faz a mágica.** Mágica é bom; muita mágica é fatal.”⁸⁰

A crítica de Schloss (2003, p. 239): “considerar a visão do observador sobre os modos de performance das interações físicas e mapeamentos de gestos em som, para fazer uma performance convincente e efetiva”⁸¹ era especificamente direcionada aos compositores que improvisam música computacional no palco com foco apenas no aspecto sonoro ou tecnológico. Sua questão tange a ausência de gestos referenciais, esforço físico, no caso

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.

⁸⁰ Tradução nossa de *To reiterate, now that we have fast enough computers to perform live, we have new possibilities, and a new problem. From the beginning of the archeological evidence of music until now, music was played acoustically, and thus it was always physically evident how the sound was produced; there was a nearly one-to-one relationship between gesture and result. Now we don't have to follow the laws of physics anymore (ultimately we do, but not in terms of what the observer observes), because we have the full power of computers as interpreter and intermediary between our physical body and the sound production. Because of this, the link between gesture and result can be completely lost, if indeed there is a link at all. This means that we can go so far beyond the usual cause-and-effect relationship between performer and instrument that it seems like magic. Magic is great; too much magic is fatal .*

⁸¹ Tradução nossa de *Its now necessary, (...) ;to consider the observer's view of the performer's modes of physical interactions and mappings from gesture to sound, in order to make the performance convincing and effective..*

de performances com dispositivos extendidos, o problema do movimento exagerado, e a expectativa cênica na performance musical:

- “1. Causa-e-efeito é importante, pelo menos para o observador/audiência em uma sala de concerto.
- 2. Corolário: Mágica na performance é bom. Muita mágica é fatal! (chato).
- 3. Um componente visual é essencial para a audiência, tal como existe um aparato visual de entrada para parâmetros e gestos.
- 4. Sutileza é importante. Grandes gestos são facilmente visíveis de longe, o que é bom, mas eles são movimentos de desenho animado se comparados à execução de um instrumento musical.
- 5. Esforço é importante. Neste sentido, nós estamos em desvantagem de desempenho na performance musical com o computador.
- 6. Improvisação no palco é bom, mas “mimar” o aparato no palco não é improvisação, é edição. É provavelmente mais apropriado fazer isso no estúdio antes do concerto, ou se durante o concerto, com o console no meio ou atrás da sala de concerto.
- 7. Pessoas que representam devem representar. Um concerto de música de computador não é uma desculpa/oportunidade para um programador(a) se sentar no palco. Sua presença melhora ou impede o desempenho da representação? ”⁸²

Duas opiniões divergentes resolvem seus problemas de maneiras divergentes sem considerarem como uma pode auxiliar a outra. No item 3, é apontado uma questão: para a audiência, e não para o improvisador, o componente visual é essencial (substantificação provável da prática). Ward et al. vão no caminho oposto ao de Schloss, e exageram este item, ao projetar códigos. Mas para Schloss, realizar isso é mimar o aparato (e o público), e tornar a apresentação pedante. É curioso notar, que Schloss faz um apontamento importante, no item 5, sobre a ausência de esforço. Não que ela seja premissa para o resultado sonoro. Mas para o público, e Schloss trata exatamente deste ponto na performance musical (item 1). Mas a crítica mais ácida é o item 7, cujo pensamento não difere de uma lógica fordiana: as atividades de artista e de programador devem ser bem definidas, e separadas. Para Schloss, são duas atividades que não se complementam. Para McLean, são interdisciplinares.

1.4.1.3 Algorithms are Thoughts, Chainsaws are Tools

“Algorithms are Thoughts, Chainsaws are Tools” é o nome dado ao vídeo de Stephen Ramsay (2010), publicado no Vimeo, em 27 de fevereiro de 2010, como um *Coffee-Table*

⁸² Tradução nossa de *1. Cause-and-effect is important, at least for the observer/audience in a live concert venue. 2. Corollary: Magic in a performance is good. Too much magic is fatal! (Boring). 3. A visual component is essential to the audience, such that there is a visual display of input parameters/gestures. The gestural aspect of the sound becomes easier to experience. 4. Subtlety is important. Huge gestures are easily visible from far away, which is nice, but they are cartoon- movements compared to playing a musical instrument. 5. Effort is important. In this regard, we are handicapped in computer music performance. 6. Improvisation on stage is good, but “baby-sitting” the apparatus on stage is not improvisation, it is editing. It is probably more appropriate to do this either in the studio before the concert, or if at the concert, then at the console in the middle or back of the concert hall. 7. People who perform should be performers. A computer music concert is not an excuse/opportunity for a computer programmer to finally be on stage. Does his/her presence enhance the performance or hinder it?*

Movie. É uma análise pessoal da performance de *Strange Places* de Andrew Sorensen. O nome do vídeo é derivado de uma das regras práticas apresentadas na [subseção 1.4.1.1](#), p. 30; mais especificamente, o sexto item.

O algoritmo como pensamento é um espaço conceitual abstrato (ver [Capítulo 2](#), p. 37); pode conter qualquer fundamento teórico pertinente para uma improvisação específica. O dispositivo usado (motoserra, máquina de tecelagem ou o computador) é um meio pelo qual uma estratégia transversal (ver [subseção 2.1.1](#), p. 38) toma sua forma sonora. É interessante aqui notar que este vídeo contém uma descrição e comentários que podem elucidar a frase-alvo sob o prisma da partitura musical. Abaixo realizei uma compilação de fragmentos de alguns dos comentários que considerei pertinentes. Ramsey apresenta a seguinte descrição do vídeo:

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 tímpanos⁸³.

Sem entrar em méritos críticos do registro, limitamo-nos a descrever como um VLog, uma variante no formato audiovisual de *weblogs*⁸⁴. Se caracteriza por ser um vídeo de curta duração, com opiniões pessoais de quem fez, geralmente no quarto da pessoa, com *headsets* (microfone+headphones). A prática de inserir comentários em um é bastante útil para levantar outras opiniões. Realizamos a tradução de alguns comentários. Não são nossas opiniões, mas podem oferecer ao leitor uma abrangência sobre o que pensam um público entusiasta.

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.

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*, constitui simplesmente 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?⁸⁵

⁸³ [RAMSAY, 2010](#), loc. cit. 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..*

⁸⁴ Cf. [BAKER](#).

⁸⁵ [RAMSAY, 2010](#), loc. cit. 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*

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

O que torna o *livecoding* diferente, e pode a performance de música tradicional imitar isso? Para responder esta questão, parece importante notar que as formas nas quais a música improvisada muitas vezes apela para alguma noção de autenticidade ou gênio. Enquanto o *livecoding* ele mesmo à noção de virtuosismo de código, “autenticidade” parece fora de lugar aqui. Se música improvisada sugere expressão, o *livecoding* sugere um conjunto de restrições na expressão, descrevendo os parâmetros através dos quais a máquina [midi] ganha expressão⁸⁶

Michel Pasin defende que o ato de improvisação musical requer conhecimentos técnicos prévios, mas não necessariamente correlacionados ao conhecimento do que é uma partitura: “Em geral, é somente dominando um instrumento que você pode esquecer sobre a técnica e concentrar em ‘dizer’ coisas com o instrumento.”⁸⁷. Este caso é bastante específico de performances com linguagens de baixo e alto-nível. Porém seria possível objetar que a prática de construção de linguagens artificiais, no topo de outras linguagens artificiais, possibilita um praticante não-familiarizado com a programação elaborar rotinas computacionais. Porém aí caímos em um problema: estaria o praticante realizando uma improvisação de códigos? ou melhor, isso importa, se o objetivo é a criação musical? Para delinear estas questões buscamos definir no próximo capítulo o que consideramos por objetivo de um agenciamento sonoro improvisado.

1.5 Discussão

Neste capítulo delimitamos a discussão musical afim de ilustrar a improvisação de códigos do ponto de vista de uma bricolagem conceitual. Exemplificamos a prática de improvisar códigos em situações onde é possível seguir dois caminhos para criar artefatos artísticos: o primeiro segue a trilha do formalismo, com concertos, espetáculos em teatros. O segundo ocorre informalmente, e geralmente está ligado à diversão, ou um encontro como uma oficina. Por outro lado, esta divisão binária pode expandir para áreas como tecelagem, dança e música. Não discutimos a situação audiovisual em seus detalhes, pois é um campo proeminente. Existem ainda as abordagens de instalações artísticas, jogos, e aplicações pedagógicas. Mas deixaremos a tarefa para quem se dispor. Ademais, oferecemos um cenário

its points of interest, actually. Or what would the livecoding analog be to a non-improvisational live performance of music?

⁸⁶ Tradução nossa de (...) *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.*

⁸⁷ Tradução nossa de *In general, it is only by mastering an instrument that you can forget about the technique and concentrate on 'saying' things with the instrument..*

proto-histórico do ponto de vista, na Itália com o compositor Pietro Grossi, nos EUA com Jim Horton, John Bischoff, Tim Perkis, e na Holanda com Ron Kuivila (residente nos EUA), que deram suporte ao pensamento promovido na Inglaterra e Alemanha. Sugestões para essa proto-história foram colocados no [Apêndice B](#). Na

2 Metodologia

Este capítulo busca investigar um método analítico de improvisações que possibilita estruturar *imagens mentais* e *espaços conceituais* do que está sendo observado como atividade criativa (ver [seção 2.1](#), p. 37). O “Quadro Conceitual de Sistemas Criativos”¹ (ver [subseção 2.2.1](#), p. 45) será utilizado como método de análise para uma improvisação de códigos (ver [Capítulo 3](#), p. 53).

2.1 Criatividade e Sistemas Criativos

Esta teoria supõe uma diferença entre criatividade, Computação criativa, Sistemas criativos e Comportamentos criativos. Sumarizamos estas diferenças na [Tabela 1](#).

Tabela 1 – Definições formais de criatividade por [Wiggins \(2006, p. 451\)](#)

Criatividade	“A performance de tarefas que, quando executados por um humano, são consideradas criativas” ²
Computação criativa	“O estudo e suporte, através de meios e métodos computacionais, do comportamento exibido por sistemas naturais e artificiais, que são considerados criativos”. ³
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” ⁴

Por exemplo, a criatividade como uma *prática reflexiva*, descrita por [McLean \(2011, p. 119\)](#). Ilustra o pintor Paul Klee. Supondo que o artista cria uma imagem mental do que irá fazer (de onde vem essa imagem mental, pode ser uma experiência real, uma lembrança, uma exploração intelectual, uma aula). Decide então esboçar em um desenho. O próximo passo será visualizar o que fez, escolher os gestos apropriados, e readequar outros. Como parte do esboço, pode elaborar seu espectro de cores, e como se organizam no espaço, se são pontos, linhas (com cores quentes ou frias), planos estruturais e questões de luz. Com uma parte decidida, pode ir para o quadro e começar a materializar o esboço. No meio do caminho, percebe que aquilo que foi considerado adequado no esboço, é inadequado para a situação prática, e readequa, diretamente no quadro ou pode voltar ao esboço e redefinir os elementos e depois voltar ao quadro. Após observar o que fez, reage ao resultado dentro de seu próprio sistema estético. Essa reação é acompanhada de uma re-elaboração da imagem mental feita anteriormente, e por novas estratégias de conversão entre a imagem

¹ Tradução nossa de *Creative System Frameworks*.

² Tradução de *The performance of tasks which, if performed by a human, would be deemed creative..*

³ 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..*

⁴ Tradução de *One or more of the behaviours exhibited by a creative system.*

mental e a imagem física. O processo continua até que o ofício seja considerado completo (*obra*).

Este não é o mesmo processo utilizado pelo(a) improvisador(a)-programador(a). O material, e o objetivo são diversos do(a) artista plástico(a). Seu material é o texto, mas não o texto discursivo, e sim o texto que descreve uma rotina de tarefas computacionais. Seu objetivo não é chegar a uma obra finalizada, mas sim não descontinuar o processo. McLean chama esse processo de programação por bricolagem (ver [Figura 10](#), p. 39). Em ambos os casos, na prática reflexiva e na bricolagem, são necessários conceitos que suportem a intenção (e vontade) do ofício. Conceitos elaborados através de restrições de idéias podem ser sistematizados, com o auxílio de esquemas visuais. Esta abordagem é muito semelhante à posição defendida por McLean, onde conceitos podem ser representados geometricamente. No caso do(a) improvisador(a)-programador(a), *espaços conceituais* são reelaborados indefinidamente, começando por uma *estratégia transversal*, passando para a observação, reação, e reformulação da imagem mental.

“A Figura 6.2 [Figura 10] caracteriza a programação por bricolagem como um laço retroalimentado envolvendo o algoritmo escrito, sua interpretação, e a percepção do programador e sua reação ao resultado ou comportamento [do algoritmo]. (...). No começo o programador tem um conceito meio-formado que só atinge consistência interna através do processo de ser expresso como um algoritmo. O laço interno é onde o programador elabora o objetivo de suas imaginações, e o laço externo é onde essa trajetória está fundamentada na pragmática do que elas realmente têm que fazer. Através deste processo ambos algoritmos e conceitos são desenvolvidos até que o programador sinta que um se aplica com o outro, ou de outra forma julga o processo criativo finalizado.”⁵

2.1.1 O programa será transcendido - Língua Artificial é o caminho

Definir verbalmente, “imagem mental”, em termos visuais, estabelece uma relação entre a observação e a verbalização. Por observação não queremos dizer apenas a visão, mas estados conduzidos pela comunicação. Para [McLean \(2011, p. 24\)](#), imagens mentais se relacionam com quaisquer estados quase-perceptuais: “Por exemplo o uso de entonação prosódica na fala é *paralinguística*, não inteiramente notada em texto escrito, mas ainda assim simboliza um conteúdo significativo.”⁶. Isto é, na teoria da *Codificação Dual* ([PAIVIO](#),

⁵ Tradução nossa de *Figure 6.2 characterises bricolage programming as a creative feedback loop encompassing the written algorithm, its interpretation, and the programmer's perception and reaction to its output or behaviour. (...). At the beginning, the programmer may have a half-formed concept, which only reaches internal consistency through the process of being expressed as an algorithm. The inner loop is where the programmer elaborates upon their imagination of what might be, and the outer where this trajectory is grounded in the pragmatics of what they have actually made. Through this process both algorithm and concept are developed, until the programmer feels they accord with one another, or otherwise judges the creative process to be finished.*

⁶ Tradução nossa de *For example the use of prosodic intonation in speech is paralinguistic, not entirely notated in written text, yet symbolising meaningful content.*

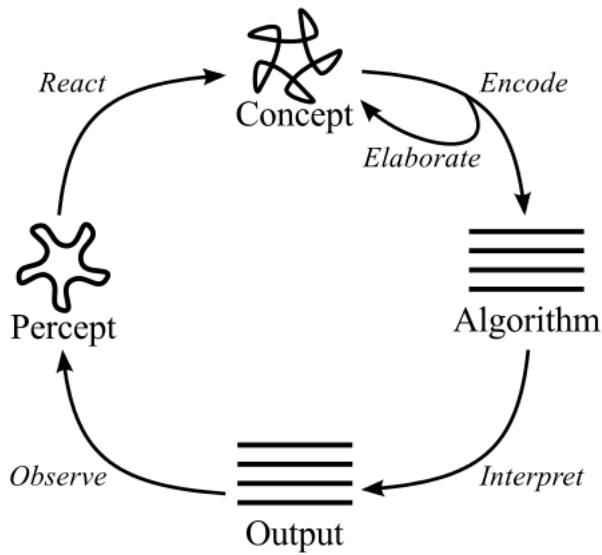


Figure 6.2: The process of action and reaction in bricolage programming

Figura 10 – Modelo de bricolagem para o processo criativo realizado por um artista-programador. **Fonte:** McLean (2011, p. 122).

1990 apud MCLEAN, 2011, p. 25–29), é estabelecida uma relação de colaboração entre hierarquias de linguagem e códigos de percepção. Esta colaboração estrutura a informação comunicada:

“Seu [Paivio] argumento não é que existem dois códigos, mas sim que existe uma hierarquia de códigos, que se ramificam no topo em códigos lingüísticos discretos e códigos de percepção contínua, que Paivio nomeia como *logogens* e *imagens* respectivamente. (...) A explicação oferecida pela teoria da Codificação Dual é que existem sistemas de símbolos distintos, mas integrados, para linguagens e figuras.”⁷

Para Paivio e McLean, “humanos são capazes de compreender linguagem enquanto simultaneamente atendem às figuras [de linguagem].”⁸ Por exemplo, quando falamos em **código de computador**, podemos imaginar um computador, e provavelmente uma pessoa escrevendo um texto. Pode ser também que apenas o computador e o texto escrito apareçam na mente. Pode ser também que apenas o texto escrito, em uma linguagem que não sabemos, seja a imagem mental. Dependendo da experiência acumulada com este objeto **código**, podemos visualizar outras possibilidades. Mas e o que vêm à mente quando imaginamos algo anterior ao código de computador, sendo que este último materializa a imaginação? ou melhor, quais são os mecanismos que o improvisador utiliza para estimular sua imaginação?

⁷ Tradução nossa de *His contention is not that there are two codes, but rather that there is a hierarchy of code, which branch at the top into discrete linguistic codes and continuous perceptual codes, which Paivio names logogens and images respectively (...) The explanation offered by Dual Coding Theory is that there are distinct, yet integrated symbol systems for imagery and language..*

⁸ Tradução nossa de *humans are able to comprehend language while simultaneously attending to imagery..*

Um sistema integrado entre códigos linguísticos e perceptuais é contextualizado por McLean no caso da improvisação de códigos (*live coding*): se atentarmos para uma partitura-programação, veremos que o programador lida com duas hierarquias. A primeira delas inclue a execução de uma *estratégia transversal*, uma formalização da imagem criada para a notação do código. Isso inclue o comentário, que explica textualmente a imagem e seu resultado; a disposição espacial do código; o destacamento de sintaxe (cores); nomes de variáveis; e nomes de métodos. A segunda hierarquia é descrita através do *Quadro de estruturação das Dimensões Cognitivas da Notação* (CHURCH; GREEN, 2008 apud MCLEAN, 2011, p. 95–97). No caso, é um referencial para a elaboração de novas linguagens, cuja sintaxe pode ser moldada para atividades específicas (“Linguagens de Domínio Específico”⁹ ou DSL). Para McLean, DSLs “provêm termos padronizados para descrever demandas particulares em um domínio de uma tarefa” (ver Tabela 2, p. 40). No capítulo anterior, (ver subseção 1.4.1.1, p. 30), o item “O programa será transcendido - Língua Artificial é o caminho.” trata justamente desta questão, de uma língua que traduza, a partir de algumas restrições, a imagem mental:

Tabela 2 – Dimensões cognitivas da Notação para linguagens de programação. **Fonte:** (CHURCH; GREEN, 2008 apud MCLEAN, 2011).

Dimensão	Significado
Abstração	“Disponibilidade de mecanismos de abstração” ¹⁰
Dependências escondidas	“Invisibilidade de ligações importantes entre entidades.” ¹¹
Compromisso prematuro	“Restrição na ordem de execução das coisas.” ¹²
Notação secundária	“Notação diversa da sintaxe formal.” ¹³
Viscosidade	“Resistência à mudança.” ¹⁴
Proximidade de mapeamento	“Proximidade de representação para o domínio-alvo.” ¹⁵
Consistência	“Semânticas similares são expressadas em formas sintáticas similares.” ¹⁶
Dispersividade	“Prolixidade da linguagem.” ¹⁷
Tendência ao erro	“Probabilidade de erros.” ¹⁸
Operações mentais difíceis	“Demanda de recursos cognitivos.” ¹⁹
Provisoriedade	“Grau de compromisso com ações e marcos.” ²⁰
Função de expressividade	“medida em que o efeito de um componente pode ser inferida.” ²¹

⁹ Tradução nossa de *Domain Specific Languages*.

Não explicaremos todas, pois não são dimensões autônomas, o que diverge do foco de nosso trabalho. McLean exemplifica a interdependência entre viscosidade e notação secundária: a possibilidade de diferentes soluções para o mesmo resultado. Isto é, se uma Notação secundária é possível. Por exemplo, em um *patch* de PD, cuja imagem mental é uma onda quadrada. Podemos utilizar objetos nativos ou objetos extendidos (ver Figura 11, p. 41). Uma outra interdependência de dimensões, entre as Operações Mentais Difíceis e a não-Invisibilidade de Dependências Escondidas, é notória do ponto de vista pedagógico. Podem afastar o compositor de seu objetivo musical, mais ou menos estruturado em uma teoria. DSLs como PureData, Max/MSP, Csound, SuperCollider, Tidal, praticam a invisibilidade de dependências em diferentes graus. Porém, entre os improvisadores de código, é considerado virtuosismo o praticante recorrer às Operações Mentais difíceis com o mínimo possível de Invisibilidade de Dependências. Por exemplo, utilizar linguagens de baixo nível como C (ver subseção 1.3.1, p. 17), ou de alto nível como Perl (McLean, 2004) e ainda assim, elaborar sons e imagens de maneira criativa.

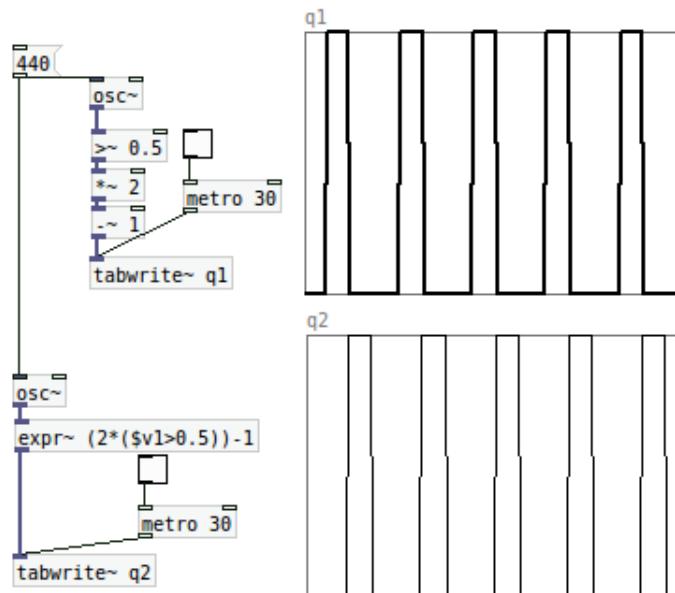


Figura 11 – Exemplo de uma característica de viscosidade e notação secundária no Pure-Data. **Fonte:** autor.

¹⁰ Tradução nossa de *Avaliability of abstraction mechanisms*.

¹¹ Tradução nossa de *Invisibility of important links between entities..*

¹² Tradução nossa de *Constraints on the order of doing things..*

¹³ Tradução nossa de *Notation other than formal syntax..*

¹⁴ Tradução nossa de *Resistance to change..*

¹⁵ Tradução nossa de *Closeness of representation to target domain..*

¹⁶ Tradução nossa de *Similar semantics are expressed in similar syntactic forms.*

¹⁷ Tradução nossa de *Verbosity of language..*

¹⁸ Tradução nossa de *Likelihood of mistakes..*

¹⁹ Tradução nossa de *Demand on cognitive resources..*

²⁰ Tradução nossa de *Degree of commitment to actions or marks..*

²¹ Tradução nossa de *Extent to which the purpose of a component may be inferred..*

2.1.2 Bricolagem como estratégia transversal

Vamos ilustrar um pequeno processo da *estratégia transversal* com o *software* Tidal. Segundo [McLean e Wiggins \(2010b, p. 2\)](#), *Tidal* é uma linguagem de composição generativa, onde “padrões podem ser compostos de numerosos subpadrões em uma variedade de maneiras e para uma profundidade arbitrária, para produzir [partes] inteiras complexas de partes simples”²². Amostras sonoras representam imagens mentais de suas fontes (por exemplo “sn” para *snare*, caixa-clara), com ritmos organizados com o auxílio de símbolos delimitadores de tempo (como espaço, “ “, e colchetes, “[”, “]”, { e }). Ritmos podem ser revertidos (*rev*), diminuídos e aumentados (*slow*, *density*), iterados (*every*) para recombinação permutação, padrões mais complexos (*can*), como o algoritmo bjorklund que simula ritmos tradicionais²³. Efeitos de panoramização, atraso (*delay*), filtros e comunicação de rede. No Exemplo 2.1. a imagem mental é a demanda da linguagem, que é produzir Música Eletrônica para Dançar.

“Tidal é uma linguagem de padrões imbebida em uma linguagem de programação Haskell, consistindo de representação de padrão, uma biblioteca de padrões geradores e combinadores, um [mecanismo] de agendamento de eventos e uma interface para programar ao vivo. Esta é uma extensa re-escrita de um trabalho anterior introduzido sobre o título *Petrol* [[McLean e Wiggins \(2010a\)](#)]. Extensões incluem melhoramentos de representação de padrão e uma integração totalmente configurável do protocolo Open Sound Control [[opensoundcontrol.org \(2002\)](#)] ([McLean; WIGGINS, 2010b](#)) ”²⁴

Exemplo 2.1 (Exemplo de Estratégia Transversal)

Imagen mental: um *loop* sincopado, mas bastante regular, descrito em um compasso. Em uma “partitura-mental”, estruturamos o primeiro tempo com um baixo, que volta a tocar na segunda semicolcheia do terceiro tempo. No Segundo tempo, silêncio. No quarto tempo uma caixa aberta:

O padrão acima pode ser elaborado em uma voz (*d1*), que redireciona (\$) a função que toca amostras sonoras (*sound*). Esta função lê uma corrente de caracteres

²² Tradução nossa de *patterns may be composed of numerous subpatterns in a variety of ways and to arbitrary depth, to produce complex wholes from simple parts.*

²³ Cf. [TOUSSAINT, 2005](#)

²⁴ Tradução nossa de *Tidal is a pattern language embedded in the Haskell programming language, consisting of pattern representation, a library of pattern generators and combinators, an event scheduler and programmer's live coding interface. This is an extensive re-write of earlier work introduced under the working title of Petrol [15]. Extensions include improved pattern representation and fully configurable integration with the Open Sound Control (OSC) protocol [16].*

(string) separados por um espaço em branco. Espaços em branco são delimitadores temporais. Cada subdivisão temporal é representada por delimitadores como [e].

```
-- Electronic Dance Music, BPM = 120
-- tempo 1 - baixo          (bass)
-- tempo 2 - silencio       (silence)
-- tempo 3 - silencio + baixo
-- tempo 4 - caixa          (sn e sn:4)
d1 \$ (sound "bass3 silence [silence bass3] sn:4")
```

Sonoramente, é útil para começar. Mas uma Música Eletrônica para Dançar requer mais elementos. Seguiremos com mais dois passos. Podemos complementar os ritmos com uma caixa e um baixos mais secos no segundo e terceiro tempo.

The musical score consists of three staves: House bass, Snare drum, and bass3. The time signature is common time (C). The House bass staff has a continuous eighth-note pattern. The Snare drum staff has a sixteenth-note pattern with accents. The bass3 staff has an eighth-note pattern. All staves begin with a note on the first beat of each measure.

```
-- Electronic Dance Music, BPM = 120
-- tempo 1 - baixo          (bass)
-- tempo 2 - silencio       (silence)
-- tempo 3 - silencio + house
-- tempo 4 - caixa          (sn e sn:4)
d1 \$ (sound "bass3 sn [silence house] sn:4")
```

É possível também fazer com que este padrão reduza seu tempo pela metade a cada quatro tempos, :

The musical score consists of three staves: House bass, Snare drum, and bass3. The time signature is common time (C). The House bass staff has an eighth-note pattern. The Snare drum staff has a sixteenth-note pattern with accents. The bass3 staff has an eighth-note pattern. The pattern repeats every two measures, effectively halving the tempo of the previous score.

```
-- Electronic Dance Music, BPM = 120
-- com uma caixa seca no segundo tempo
```

```
-- e uma caixa aberta no quarto tempo
-- A cada 4 tempos, o ritmo diminui pela metade
-- e depois volta ao normal.
d1 \$ every 4 (density 0.5) (sound "bass3 sn [silence house] sn:4")
```

Para McLean (2011, p. 130), esta estratégia criativa, de programar “no momento”, a partir de um arquivo de texto em branco, com uma imagem mental do resultado sonoro (ou visual), é caracterizada pela bricolagem. No início do exemplo acima, o programador elabora um meio-conceito do que quer fazer, cuja expressão apenas ganha existência através da codificação (ver Figura 10, p. 39). As fases de observação, e reação levam o improvisador programador à reconceitualização, e um novo código é escrito. No entanto, ao invés de finalizar, o improvisador segue desenvolvendo.

2.2 Sistemas criativos

Para Wiggins (2006, p. 450), criatividade pode ser discutida do ponto de vista lógico, orientado geometricamente. A Equação 2.2 considera a imagem mental de um Universo vazio, e com o máximo potencial de criação.

$$\mathcal{U} = \emptyset \quad (2.1)$$

Wiggins define o Universo de Conceitos (\mathcal{U}_x) como um conjunto não estrito dos *Espaços Conceituais* (\mathcal{C}_x) de Margaret Boden (1990). Isto é, um Universo de Conceitos a respeito de alguma coisa, no nosso caso da improvisação de códigos (ver Equação 2.2, p. 44).

$$\mathcal{U}_{livecoding} = [\mathcal{C}_{Tecelagem}, \mathcal{C}_{Audiovisual}, \mathcal{C}_{Dança}, \mathcal{C}_{Música}, \dots, ?] \quad (2.2)$$

“Boden concebe o processo de criatividade como uma identificação e/ou localização de novos objetos conceituais em um espaço conceitual.”²⁵. Uma definição semelhante de Thornton (2007, p. 2) reafirma a criação de uma imagem mental de uma *metáfora* (que pode situar sons, palavras, imagens, cheiros,tato,gostos):“Qualquer ato criativo é fundado na conceitualização ou realização de um ponto dentro de um espaço conceitual particular”²⁶.

²⁵ Tradução nossa de *Boden conceives the process of creativity as the identification and/or location of new conceptual objects in a conceptual space..*

²⁶ Tradução nossa de *Any creative act is thus founded on conceptualisation or the realisation of a point within a particular ‘conceptual space’.*

Neste sentido, selecionamos a imagem mental da metáfora sonora, a partir de um espaço conceitual da Música. Dentro espaço conceitual da Música, selecionamos um específico que vem do *jazz*. Nossa escolha não foi por gosto (ainda que o autor e o orientador gostem), mas sim por ilustrar a intenção de readequação da figura do intérprete concertista para os propósitos do programador (ver subseção 1.4.1.2, p. 31).

2.2.1 Quadro Conceitual de sistemas criativos

Uma maneira adequada de descrever um sistema criativo (ou parte dele) considera um *Universo de Conceitos*:

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).²⁷

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, representações, e conceitos:

“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).”²⁸

²⁷ Tradução de *The universe, \mathcal{U} , is a multidimensional space, whose dimensions are capable of representing anything, and all possible distinct concepts correspond with distinct points in \mathcal{U} . (...) To make the proposal as state-spacelike as possible, I allow that \mathcal{U} contains all abstract concepts as well as all concrete ones, and that it is therefore possible to represent both complete and incomplete artefacts*

²⁸ 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) ²⁹
\mathcal{L}	Linguagem	Linguagem utilizada para expressar regras.
\mathcal{A}	Alfabeto	Alfabeto da linguagem que contém caracteres apropriados para 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 significa uma falsidade booleana; a necessidade disso para valores reais se tornará clara abaixo” (WIGGINS, 2006, p. 452) ³⁰
$[[\mathcal{R}]]$	Regras de validação	“Uma função que interpreta \mathcal{R} , resultando em uma função indicando aderência ao conceito em \mathcal{R} ” ³¹
$\mathcal{C} = [[\mathcal{R}]](\mathcal{U})$	Espaço Conceitual	“Todos espaços conceituais são um subconjunto não-estrito de \mathcal{U} ” ³² . 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} ³³
\mathcal{T}	Regras de detecção	“Regras definidas dentro de \mathcal{L} para definir estratégias transversais para localizar conceitos dentro de \mathcal{U} ” (McLean, 2006) ³⁴
\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) ³⁵ “Regras definidas dentro de \mathcal{L} para avaliar a qualidade ou a desejabilidade do conceito c ” (McLean, 2006) ³⁶
$<<< \mathcal{R}, \mathcal{T}, \mathcal{E} >>>$	Função de interpretação	Uma regra necessária para definir o espaço conceitual, “independentemente da ordem, mas também, ficionalmente, enumerá-los em uma ordem particular, sob o controle de \mathcal{T} – isto é crucial para a simulação de um comportamento criativo de um \mathcal{T} particular (WIGGINS, 2006) ³⁷ . “Uma função que interpreta a estratégia transversal \mathcal{T} , informada por \mathcal{R} e \mathcal{E} . Opera sobre um subconjunto ordenado de $mathcal{U}$ (do qual tem acesso randômico) e resulta em outro subconjunto ordenado de \mathcal{U} .” ³⁸

²⁹ Tradução de *A universe of all possible concepts*.

³⁰ 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.

³¹ Tradução de A function interpreting \mathcal{R} , resulting in a function indicating adherence of a concept to \mathcal{R}

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). McLean realiza uma comparação entre o *Universo de possibilidades* de Wiggins com o *Modelo de Improvisação* de Pressing (ver seção 2.3, p. 47). No entanto, McLean argumenta que:

Pressing discute comportamento criativo no contexto do Modelo de Improvisação, e de fato é parte do Quadro conceitual 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 linguagem comum na qual comparar com outros modelos³⁹.

2.3 O modelo de improvisação

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 elementos de uma performance exemplar, como o caso investigado neste trabalho. 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. Este nos pareceu um modelo mais transparente para o compositor, músico e intérprete. O que não quer dizer que é possível readequar ambos para nosso interesse. Um sumário sobre o modelo de improvisação é apresentado na Tabela 4. Por seu caráter lógico, parece ser uma possibilidade interessante, e assumiremos como tal.

³² Tradução de *All conceptual spaces are non-strict subset*.

³³ Tradução de *A function interpreting \mathcal{R} , resulting in a function indicating adherence of a concept to \mathcal{R}* .

³⁴ Tradução de *Rules defined within \mathcal{L} to define a traversal strategy to locate concepts within \mathcal{U}*

³⁵ Tradução de *(...) set of rules which allows us to evaluate any concept we find in \mathcal{C} and determine its quality, according to whatever criteria we may consider appropriate*.

³⁶ Tradução de *Rules defined within \mathcal{L} which evaluate the quality or desirability of a concept c* .

³⁷ 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}* .

³⁸ Tradução de *A function interpreting the traversal strategy \mathcal{T} , informed by \mathcal{R} and \mathcal{E} . It operates upon an ordered subset of \mathcal{U} (of which it has random access) and results in another ordered subset of \mathcal{U}* .

³⁹ 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*.

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 ⁴⁰
K'	Uma seqüência de blocos de eventos E, onde um bloco de eventos não se sobrepõe com o seguinte ⁴¹
I'	Uma improvisação, particionada por interrupções em um número de K sequências ⁴²
R'	Um referente opcional, tal como uma partitura ou uma norma estilística ⁴³
G'	Um conjunto de objetivos ⁴⁴
M'	Uma memória de longo prazo ⁴⁵
O'	Um conjunto de objetos ⁴⁶
F'	Um conjunto de características dos objetos ⁴⁷
P'	Um conjunto de processos ⁴⁸

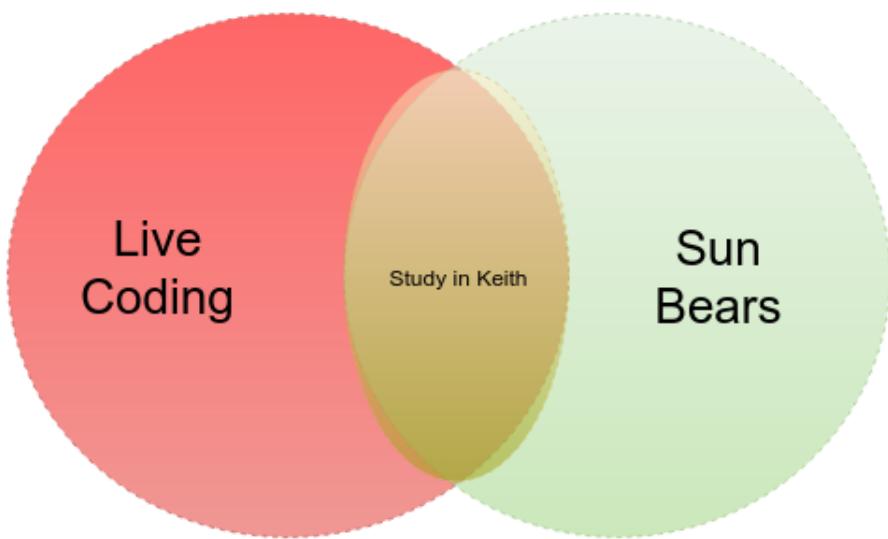


Figura 12 – Representação da justaposição entre dois espaços conceituais. A região em marrom representa um grupo de conceitos transitórios, bem como os limites desta transição. **Fonte:** autor.

⁴⁰ A cluster of sound events.

⁴¹ A sequence of E event clusters, where event cluster onsets do not overlap with those of a following one

⁴² An improvisation, partitioned by interrupts into a number of K sequences

⁴³ An optional referent, such as a score or stylistic norm

⁴⁴ A set of current goals.

⁴⁵ Long term memory.

⁴⁶ An array of objects.

⁴⁷ An array of objects Features.

⁴⁸ An array of Process

2.4 Diagramação dos espaços conceituais

Formalmente, a figura acima pode ser representada como na [Equação 2.4](#), se desconsiderarmos qualquer outros espaços conceituais.

Exemplo 2.2 (Representação formal da [Figura 12](#))

$$\mathcal{C}_{Study \text{ in } Keith} = \mathcal{C}_{live \text{ coding}} \cup \mathcal{C}_{Sun \text{ Bears}} \quad (2.3)$$

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

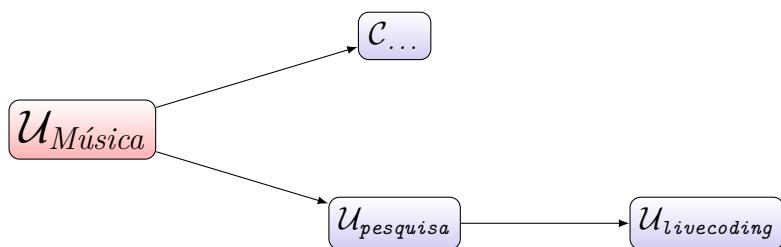
Exemplo 2.3 (Representação formal das propriedades da [Figura 12](#))

$$\mathcal{C}_{SK} = [\mathcal{E}'_{SK}, \mathcal{K}'_{SK}, \mathcal{I}'_{SK}, \mathcal{R}'_{SK}, \mathcal{G}'_{SK}, \mathcal{M}'_{SK}, \mathcal{O}'_{SKF'}, \mathcal{P}'_{SK}] \quad (2.4)$$

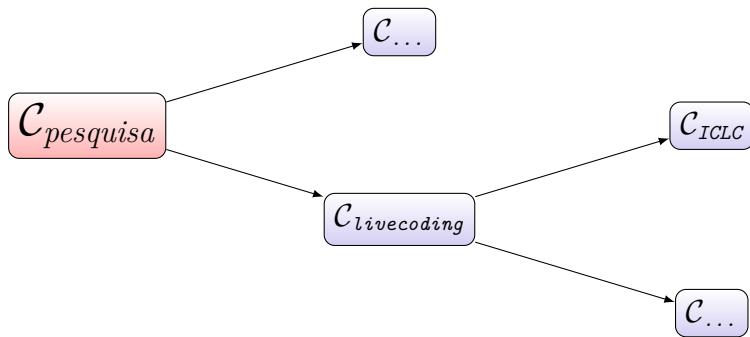
Nos diagramas abaixo, $C\dots$ representa qualquer espaço conceitual abstrato (que pode incluir outro previamente apresentado). Entre os elementos iniciais (raízes, vermelho) e transitórios (nós, azul), ocorrem as ramificações (ramos, linhas pretas), isto é, a exploração de conceitos dentro de outros 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, $\mathcal{C}_{pesquisa} = [[\mathcal{R}_{pesquisa}]](\mathcal{U}_{pesquisa})$.

Exemplo 2.4 (Representação do universo conceitual da *pesquisa*)

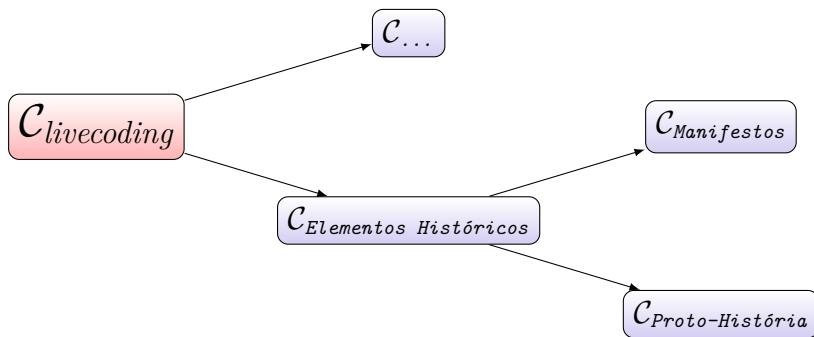
O Universo de Conceitos da pesquisa, $\mathcal{U}_{pesquisa}$, é um recorte do universo conceitual da música, $\mathcal{U}_{música}$:



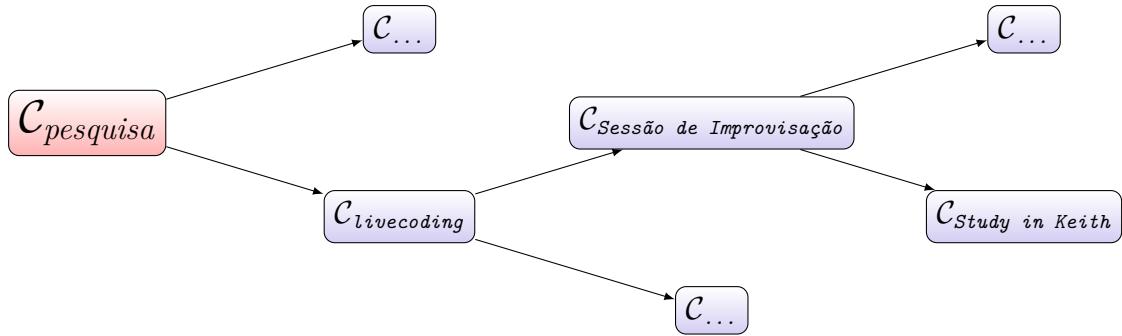
No primeiro capítulo, incluímos um subjconjunto neste Espaço Conceitual da Pesquisa (ver [Apêndice A](#), p. [77](#)).



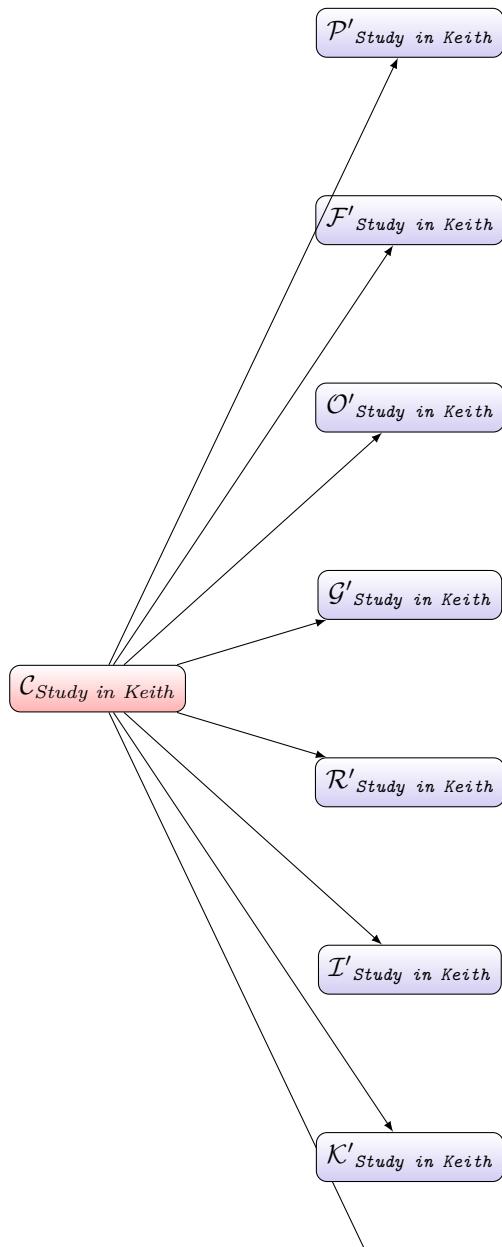
Podemos incluir elementos históricos, o período transitório entre 1970 e 2000 (*circa*), onde emanciparam as práticas e as regras heurísticas.



Por último, $C_{pesquisa}$ investiga o *live coding* a partir de um caso específico:



Por outro lado $C_{Study in Keith}$ pode ser definido pelo modelo de improvisação de Pressing (Tabela 4, 48).



Exemplo 2.5 (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). 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. 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, misturados. No entanto o espaço conceitual não será investigado em sua totalidade, e sim apenas uma sonoridade.

3 Estudo de caso

“Para fundamentar a discussão em música, considere uma peça de *jazz*, onde *jazz* é um conceito e uma composição particular é uma instância de um conceito. O musicista, explorando os limites do *jazz*, encontra então uma peça para além das regras usuais do *jazz*. Através deste processo, os limites do gênero musica podem ser redefinidos em algum grau, ou se a peça está em um novo terreno particularmente fértil, um novo sub-gênero de *jazz* emerge. Contudo uma peça de música que não quebra limites, de alguma forma pode ser considerada não-criativa.”¹

O objetivo deste capítulo será detalhado na [seção 3.1](#). Seu contexto inicia com dois registros audiovisuais de *A Study in Keith*, publicados por [Sorensen \(2015\)](#) e [Sorensen e Swift \(2009\)](#). São o mesmo registro, mas com duas descrições diversas de um mesmo espaço conceitual \mathcal{E}_{ask} . Existe uma semelhança entre as explicações: um referente opcional \mathcal{R}_{ask}^0 são os Concertos *Sun Bear* (ver [seção 3.2, p. 54](#)). [Sorensen \(2015\)](#) indica outros referentes, \mathcal{R}_{ask}^1 como o ambiente de programação *Impromptu* e \mathcal{R}_{ask}^2 a linguagem de programação *Scheme* (ver [subseção 3.2.3, p. 58](#)):

“*A Study in Keith* é uma performance de programação ao vivo por Andrew Sorensen, inspirado nos concertos *Sun Bear* de Keith Jarret. Toda a música que você ouve é gerada a partir do código do programa que é escrito e manipulado em *tempo-real* durante a performance. O trabalho foi executado usando o ambiente de desenvolvimento [em linguagem] Scheme [chamado] Impromptu ([<http://impromptu.moso.com.au>](http://impromptu.moso.com.au)). Não é Keith, mas inspirado por Keith ([SORENSEN, 2015](#)). ”²

Uma breve análise de uma sonoridade cadencial de um dos concertos será investigada para encontrar outros possíveis referentes opcionais (ver [subseção 3.2.1, p. 54](#)). As mensagens codificadas no ambiente *Impromptu* são enviadas para um segundo *software*, um instrumento virtual de estúdio (VSTi), que atua como um simulador de teste para a utilização em um instrumento acústico em uma vinheta nomeada *Disklavier Sessions* (ver [subseção 3.2.2, p. 56](#)). É indicado também um evento sonoro muito recorrente na improvisação de códigos. O silêncio é, com suas variações temporais, uma constante em outros trabalhos de Sorensen (ver [subseção 3.3.1, p. 61](#)):

¹ Tradução nossa de *To ground the discussion in music, consider a piece of jazz, where jazz is the concept and the particular composition is an instance of that concept. The musician, in exploring the boundaries of jazz, then finds a piece beyond the usual rules of jazz. Through this process, the boundaries of a music genre may be redefined to some degree, or if the piece is in particularly fertile new ground, a new sub-genre of jazz may emerge. Indeed a piece of music which does not break boundaries in some way could be considered uncreative..*

² Tradução nossa de “*A Study In Keith*” is a live programming performance by Andrew Sorensen inspired by Keith Jarrett’s *Sun Bear* concerts. All of the music you hear is generated from the program code that is written and mani[p]ulated in real-time during the performance. The work was performed using the Impromptu Scheme software development environment ([<http://impromptu.moso.com.au>](http://impromptu.moso.com.au)). Not Keith, but inspired by Keith. .

“A Study In Keith é um trabalho para piano solo (NI’s Akoustik Piano), inspirado nos concertos *Sun Bear* de Keith Jarrett. Note que não existe som durante os dois primeiros 2 minutos da performance, enquanto estruturas iniciais são construídas. Não é bem Keith, mas inspirado por Keith (SORENSEN; SWIFT, 2009)”³

3.1 Objetivo

Como pontua McLean (2011, p. 121) “Nosso estudo de caso é de alguma forma simplista e não é intenção ilustrar uma grande arte ou um grande código. Contudo delineia um processo criativo de classes, como efetuado pelo presente autor.”⁴

Este capítulo tratará sobre um grupo finito de blocos de eventos, $[\mathcal{E}'_0, \dots, \mathcal{E}'_2]$, para investigar qual é um primeiro objetivo \mathcal{G}'_0 (ver seção 3.3, p. 61). Será apresentado também uma classe objetos \mathcal{O}'_{ask} (uma figura de notas) pertinente para entender como este início de improviso inclue uma característica \mathcal{F}'_{ask} N semelhante à *Illiad Suite* de Hiller e Isaacson (1959), mas aplicado ao piano solo.

3.2 Referentes Opcionais

3.2.1 Concertos Sun Bear

Os concertos *Sun Bear* são originalmente dez LPs de improvisações de Keith Jarret no Japão, produzidos pela *ECM Records*⁵ entre 1976 e 1978. É o terceiro dos concertos de improvisação que incluem o *Solo Concerts: Bremen/Lausanne* (1973) e *The Köln Concert* (1975).

Foram realizados e gravados como sessões de improvisação contínua, variando entre 31 a 43 minutos cada. Para cada dia, duas sessões de improvisação, em cidades diferentes. Kyoto, 5 de novembro⁶; Osaka, 8 de novembro⁷; Nagoya, 12 de novembro⁸. Tokyo, 14 de novembro⁹; Sapporo, 18 de Novembro¹⁰.

Um documento crítico impresso é mencionado na *internet* como um antigo documento contendo notas discográficas (SWENSON, 1985). Seu acesso foi restrito durante a

³ Tradução nossa de "A Study In Keith" is a work for solo piano (NI's Akoustik Piano) by Andrew Sorensen inspired by Keith Jarrett's Sun Bear concerts. Note that there is no sound for the first 2 minutes of the performance while initial structures are built. Not quite Keith, but inspired by Keith..

⁴ Tradução nossa de Our case study is somewhat simplistic, and is not intended to illustrate either great art or great code. However it does trace a creative process of sorts, as carried out by the present author..

⁵ <http://www.ecmrecords.com/>

⁶ Disponível em <<https://www.youtube.com/watch?v=T2TfIQNxhjc>>.

⁷ Disponível em <<https://www.youtube.com/watch?v=FC4iZ1wMoU8>>

⁸ <<https://www.youtube.com/watch?v=3a7ezm3D1jA>>.

⁹ Disponível em <<https://www.youtube.com/watch?v=ZH8VIjjhPQ4>>

¹⁰ Disponível em <https://www.youtube.com/watch?v=BqYBT_HoG4M>

pesquisa, e não foi possível incluir alguma citação. Da mesma forma, documentos analíticos sobre a peça não foram encontrados em alguma base de dados. Existem algumas notas discográficas compiladas por uma comunidade de fãs e críticos musicais estadounidenses. Duas notas sugerem uma descrição da forma musical aplicada por Keith Jarret: “O tema de *Kyoto Parte 1* é repetido por Keith Jarret no fim de *Kyoto Parte 2*. Então podemos considerar o todo deste concerto como uma grande Suíte.”¹¹ (GARBOLINO, 2014, p. 129).

Revisto por Richard S. Ginnel¹²: [-] Este pacote gigantesco – um conjunto de dez LPs agora comprimidos em uma caixa robusta de seis [embalagens de] CDs – foi ridicularizado uma vez como uma última viagem de ego, provavelmente por muitos que não tomaram um tempo para ouvir tudo. (...) Ainda assim, o milagre é como esta caixa é consistentemente muito boa. **Na abertura de Kyoto, a meditação direcionada para o gospel** está em plena atuação, ao nível de suas melhores performances solo em Bremen e Koln, e os concertos Osaka e Nagoya possuem citações de primeira linha, geralmente do tipo *folk*, mesmo profundas, idéias líricas (GARBOLINO, 2014, p. 130)¹³.

O *gospel* e o *folk* são citados como gêneros musicais inclusivos nesta suíte. Porém esta Suíte não possui pausas entre as partes (o improviso é contínuo, mas seccionado por transições). Uma transcrição do motivo gerador deste *gospel* no concerto de Kyoto buscou encontrar referentes opcionais adicionais para \mathcal{R}'_{ask} (isto é, informações de harmonia e ritmo), mesmo com a afirmação anterior que nenhuma relação pode ser encontrada (ver Figura 13, p. 56).

O bloco de sonoridades iniciais \mathcal{E}'^0_{ask} sugere alguma referência à forma Fantasia Cromática ou um Prelúdio, no sentido de um jogo livre que prepara o improvisador para a sonoridade do instrumento no ambiente tocado, com uma progressão bastante simples. Uma sonoridade oscilante é colocada em movimento num âmbito de terça menor nos compassos 1 e 2 entre Si bemol, Dó e Ré Bemol, repetindo 3 vezes (\mathcal{P}'^0_{KJ}) dois objetos \mathcal{O}'^0_{KJ} e \mathcal{O}'^1_{KJ} : um ostinato na mão esquerda e uma bordadura na mão direita. Nos compassos 3 a 5 um terceiro objeto \mathcal{O}'^2_{KJ} , um acorde de Sol bemol Maior (transcrito assim para facilitar a leitura), dividido em uma quarta justa entre a mão esquerda e direita, com terças alternadas em colheias na mão direita. Este evento é então expandido nos compassos 6 a 10, gerando uma figura cromática proto-melódica, cujo acompanhamento harmônico

¹¹ Tradução nossa de *The theme of Kyoto Part 1 is repeated By Kj at the end of Kyoto Part 2. So we can consider the whole of this concert as one big Suite.*

¹² Disponível em <<http://www.mcana.org/formembersatlarge.html>>.

¹³ Tradução de *Review by Richard S. Ginell*: [-] This gargantuan package – a ten-LP set now compressed into a chunky six-CD box – once was derided as the ultimate ego trip, probably by many who didn't take the time to hear it all. You have to go back to Art Tatum's solo records for Norman Granz in the '50s to find another large single outpouring of solo jazz piano like this, all of it improvised on the wing before five Japanese audiences in Kyoto, Osaka, Nagoya, Tokyo, and Sapporo. Yet the miracle is how consistently good much of this giant box is. In the opening Kyoto concert, Jarrett's gospel-driven muse is in full play, up to the level of his peak solo performances in Bremen and Koln, and the Osaka and Nagoya concerts have pockets of first-rate, often folk-like, even profound, lyrical ideas.



Figura 13 – Transcrição do motivo gerador do disco Kyoto, parte 1. **Fonte:** autor.

segue uma progressão de substituição por tritono: Sol Bemol Maior, Fá Maior com sexta adicionada (ou Ré menor com sétima) e Dó Maior com sétima menor. Limitamo-nos a considerar esta progressão do ponto de vista do *blues*, o que relacionaria com o *gospel*. Por exemplo, a progressão padrão $V^7 \Rightarrow IV^7 \Rightarrow I^7$ pode ser transformada com substituições de dominantes e subdominantes. Uma operação comum é realizar uma substituição do grau $V^7 \Rightarrow bV^7$. Porém Jarret suprime esta sétima, resultando em um bV. Já o quarto grau pode substituir $IV^7 \Rightarrow IV_5^6$. Neste sentido, com as devidas mudanças, este *blues* é estruturado em uma sonoridade de tensão progressiva (tríade, tétrade de segundo grau invertido, e 'dominante'): $bV \Rightarrow IV_5^6 \Rightarrow I^7$ ou $1bV \Rightarrow {}_3II^7 \Rightarrow I^7$.

3.2.2 NI-Akoustik Piano

O NI, é uma abreviação para *Native Instruments*, uma empresa de tecnologias para áudio¹⁴. O *Akoustic Piano* é uma extensão (*plugin*) VST, que emula diferentes pianos acústicos. Entre os instrumentos utilizados, incluem o *Bösendorfer 290 Imperial*, *Steinway D*, e *Bechstein D 280*¹⁵. Os instrumentos são gravados nota a nota por um complexo

¹⁴ Disponível em <<http://www.native-instruments.com/en/company/>>.

¹⁵ Disponível em <<http://www.kvraudio.com/product/akoustik-piano-by-native-instruments>>.

sistema de tomada de som. Na [Figura 14](#), a gravação de um outro VSTi ilustra como é realizado o processo (no caso, por Uli Baronowsky)

Figura 14 – Sistema de tomada de som para produção de um VSTi. **Fonte:** <<http://www.native-instruments.com/en/products/komplete/keys/definitive-piano-collection/>>

ou em um instrumento acústico modificado, como o *Disklavier* da Yamaha (ver [Figura 15](#), p. 57). Este último caso não é citado por Sorensen (2015) e Sorensen e Swift (2009), masem *Disklavier Sessions* (SORENSEN, 2013) é semelhante à atividade descrita em *A Study in Keith*:

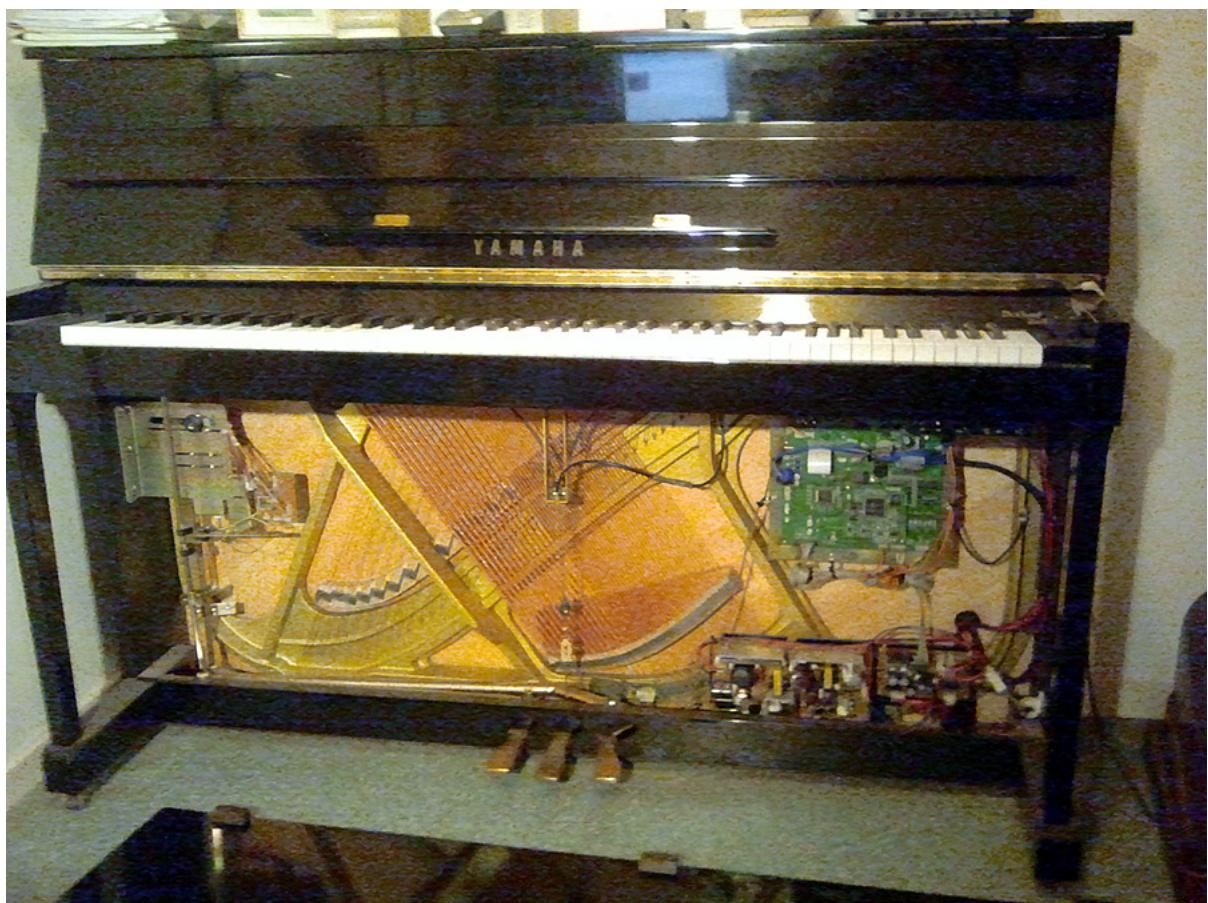


Figura 15 – Piano Disklavier de armário, com a parte interna exposta para exibir a placa-mãe. **Fonte:** [wikimedia.org](https://commons.wikimedia.org)

“Em *Disklavier Sessions* os programas escritos em tempo-real por Ben e Andrew geram um fluxo de dados de notas que são enviados para ser executado em um piano disklavier mecanizado. Assim como as alturas das notas, toda a performance do piano deve ser codificada na informação gerada pelo programa e enviada para o piano disklavier.”¹⁶

¹⁶ Tradução nossa de *In the Disklavier Sessions the programs beign written in real-time by Ben and*

3.2.3 Ambiente e Linguagem: Impromptu

“Impromptu é uma linguagem e um ambiente de programação OSX¹⁷ para compositores, artistas sonoros, VJ's e artistas gráficos com um interesse em programação ao vivo ou interativa. Impromptu é um ambiente de linguagem Scheme, um membro da família das languages Lisp. Impromptu é usado por artistas-programadores em performances de *livecoding* em torno do mundo.”¹⁸

Existe uma restrição quanto ao nicho de usuários do *software*. É indicado que este grupo é restrito aos usuários de computadores Apple; no entanto uma investigação indica que o código-base é desenvolvido em um nível acima de outro, *Extempore*.

Sorensen e Gardner (2010, p. 823) que o *ambiente de programação ciberfísico* é análogo à *partitura* tradicional; sua diferença está ligada à sua execução imediata. Nos termos de Mathews e Moore (1970), uma *engenharia humana*. Nos termos de Fenerich, Obici e Schiavoni (2014, p. 5), uma programação-partitura:

“Considere a analogia da partitura musical tradicional. A partitura provê uma especificação estática da intenção – um programa de domínio estático. Musicistas, representam o domínio do processo, executam ações requeridas para realizar ou reificar a partitura. Finalmente, as ações no domínio do processo resultam em ondas sonoras que são percebidas por uma audiência humana como música. Este estágio final é o nosso domínio real de trabalho. Agora considere um domínio de programação dinâmica no qual o compositor concebe e descreve uma partitura em *tempo-real*. Nós geralmente chamamos este tipo de composição de improvisação. Na improvisação o(a) musicista é envolvido em um circuito-fechado retroalimentado que envolve premeditação, movendo para ação casual e finalmente para reação, refinamento e reflexão.”¹⁹

3.2.4 Extempore

O *Extempore* possui uma filosofia próxima àquela descrita por Mathews e Moore (1970), isto é, de um sistema humano-máquina reflexivo. Um nome específico é dado para a atividade de escrita do código, ou *programação ciberfísica*:

Andrew are generating a live stream of note data which is sent to a mechanized diskavier piano to be performed. As well the individual note pitches all of the piano performance must be encoded into the information being generated by the program and sent to diskavier piano.

¹⁷ Sistema Operacional Mac OSX.

¹⁸ Tradução nossa de *Impromptu is an OSX programming language and environment for composers, sound artists, VJ's and graphic artists with an interest in live or interactive programming. Impromptu is a Scheme language environment, a member of the Lisp family of languages. Impromptu is used by artist-programmers in livecoding performances around the globe.* ..

¹⁹ Tradução nossa de *Consider the analogy of a traditional musical score. The score provides a static specification of intention – a static program domain. Musicians, representing the process domain, perform the actions required to realise or reify the score. Finally, the actions in the process domain result in sound waves which are perceived by a human audience as music. This final stage is our real-world task domain. Now consider a dynamic program domain in which a composer conceives of and describes a musical score in real-time. We commonly call this type of composition improvisation. In it, the improvising musician is involved in a feedback loop involving forethought, moving to causal action and finally to reaction, refinement and reflection..*

“*Extempore* é projetado para suportar um estilo de programação apelidado de [”]programação ciberfísica”. Programação ciberfísica suporta a noção de um programador humano operando como um agente ativo em uma rede de *tempo-real* distribuída de sistemas ambientalmente conscientes”²⁰

Entre suas características incluem²²:

- Processamento de Sinais Digitais (DSP)²³ em tempo-real;
- Sequenciamento de áudio (baseado em notas) de alto-nível²⁴;
- Processamento de gráficos;

A segunda característica será explorada neste capítulo como base técnica para o processo criativo em *Study in Keith*

3.2.5 Scheme

Scheme (1970) é citado na *internet* como uma definição de linguagem, ou dialeto, da linguagem Lisp (1958). Como uma linguagem imperativa orientada à expressões, utiliza *representações simbólicas* de listas, para definir rotinas que podem definir outras rotinas. Esta característica habilitou programadores a realizarem uma atividade conhecida como *meta-programação*.

Exemplo 3.1 (Notação Scheme)

Neste tipo de notação (*prefix notation*, ou notação prefixada), são listados *átomos*, que podem ser números, operadores, funções ou outras listas. Por exemplo. Uma divisão de dois números, pode ser escrita de maneira bastante simples:

```
;; uma operacao de divisao
(/ 1 2)

;; pode ser mais descritiva
(define divide (lambda (a b) (/ a b)))

;; execucao
(divide 1 2)
```

²⁰ Tradução nossa de *Extempore is designed to support a style of programming dubbed ‘cyberphysical’ programming. Cyberphysical programming supports the notion of a human programmer operating as an active agent in a real-time distributed network of environmentally aware systems.* ²¹ .

²² Disponível em <<http://benswift.me/2012/08/07/extempore-philosophy/>>

²³ Sobre DSP, Cf. SMITH, 2012-06.

²⁴ Como o disparo de sons baseado em parâmetros como altura, intensidade e duração. Disponível em <<http://benswift.me/2012/10/15/playing-an-instrument-part-i/>>.

Um exemplo musical é sugerido por Sorensen e Gardner (2010, p. 823-824) como fonte para um pseudo-código documentado. É possível notar uma ênfase em um discurso camerístico, na improvisação com instrumentos acústicos e sons eletrônicos, e uma base harmônica tonal comum:

Exemplo 3.2 (Exemplo imaginário)

NOTA: o processo será realizado de maneira semelhante àquele descrito por Mathews e Moore (1970) (??, p. ??), isto é, a partir de um circuito-fechado entre executante, máquina e projetores (visuais e acústicos).

“ Dois performers se apresentam no palco. Um violinista, em pé e parado, com seu arco preparado. Outro senta-se atrás do brilho da tela do *laptop*. Uma projeção da tela do *laptop* é projetada acima do palco, e mostra uma página em branco, com um simples cursor piscando. O musicista-programador começa a digitar ... ”²⁵

```
( play-sound ( now ) synth c3 soft minute)
```

“ ... a expressão é avaliada, e lampeja no retroprojector para exibir a ação do executante. Um som estéreo sintetizado entra imediatamente no espaço e o violinista começa a improvisar em simpatia com a novidade da textura. O músico-programador, ouve o material temático fornecido pelo violinista e começa a delinear um processo gerativo Markoviano para acompanhar o violino: ”²⁶

```
( define chords
  ( lambda ( beat chord duration )
    ( for-each ( lambda ( pitch )
      ( play synthj pitch soft duration ))
      chord )
    ( schedule (* metro * ( + beat duration )) chords
      (+ beat duration )
      ( random ( assoc chord (( Cmin7 Dmin7 )
        ( Dmin7 Cmin7 )))))
      duration )))
  ( chords (* metro * get-beat 4) Cmin7 4)
```

“... A função *chords* é chamada no primeiro tempo de um novo xxxxxxxx e uma simples progressão recursiva de acordes come a suportar a performance melódica do violino. A função *chords*

cria um laço temporal, gerando uma sequência interminável de acordes de quatro tempos. Depois de poucos momentos de reflexão, o musicista-programador começa a modificar a função *chords* para suportar uma progressão de acordes mais variada, com uma razão aleatória [em função] da recursão temporal... ”²⁷

```
( define chords
  ( lambda ( beat chord duration )
    ( for-each ( lambda ( pitch )
      ( play dls (+ 60 pitch) soft duration ))
      chord )
    ( schedule (* metro * (+ beat duration)) chords
      (+ beat duration)
      ( random ( assoc chord (( Cmin7 Dmin7 Bbmaj )
        ( Bbmaj Cmin7 )
        ( Dmin7 Cmin7 ))))
      ( random (3 6))))))

( chords (* metro * get-beat 4) Cmin7 4)
```

3.3 Blocos de Eventos

3.3.1 Primeiro evento sonoro: a sensação de quietude

Uma nota pertinente sobre esta improvisação é feita pelo próprio Sorensen: nos primeiros dois minutos do vídeo (aproximadamente 1'53''). existe um silêncio. Este silêncio é característico daquele momento em que os primeiros códigos são escritos.

Este comportamento, do tempo de codificação, ao tempo de ação musical, é similar em outros dois vídeos, de Sorensen: An evening of livecoding at 53 Rusden Street²⁸, Just for Fun²⁹, A Study in Part³⁰, Stained³¹, Transmissions in Sound³², Antiphony³³,

²⁸ Disponível em <<https://vimeo.com/2433303>>

²⁹ Disponível em <<https://vimeo.com/2433971>>

³⁰ Disponível em <<https://vimeo.com/2434054>>

³¹ Disponível em <<https://vimeo.com/2502546>>

³² Disponível em <[TransmissionsinSound](#)>

³³ Disponível em <<https://vimeo.com/2503188>>

Strange Places³⁴, Orchestral³⁵, UMDT³⁶, Day of Triffords³⁷, Face to Face³⁸, BM&E³⁹, A Christmas Carol⁴⁰ Dancing Phalanges⁴¹, Livecoding Audio DSP⁴², Jazz Ensemble Study⁴³, Variations on a Christmas Theme⁴⁴. Esta característica também foi observada em uma outra performance (ver subseção 1.3.1, p. 17).

Definição do instrumento e do tempo

Seu início é um pequeno comentário que contém o nome do executante e seu email para contato (primeiros sete segundos), bem como a escrita de um código que inicializa o NI-Akoustik (até 0'43", ver Exemplo imaginário 3.3).

Exemplo 3.3 (Definição de instrumento)

Primeiros eventos musicais gerados a partir das primeiras estruturas válidas de código.

Fonte: (SORENSEN, 2015).

```
;;;;;;;;;;;;;;;;;;;;
;; Andrew Sorensen andrew@moso.com.au
(define piano (au:make-node "aumu" "NaDd" "-NI-"))
(au:connect-node piano 0 *au:output-node* 0)
(au:update-graph)

(au:load-preset piano "/tmp/convert_grand.aupreset")
```

Em 0'52" Sorensen define um tempo base (ver Definição de instrumento 3.4). Em seguida, Sorensen apaga o código para então iniciar definições de notas (0'54").

Exemplo 3.4 (Definição de tempo)

Definição do tempo base. **Fonte:** (SORENSEN, 2015).

```
(define *metro* (make-metro 110))
```

³⁴ Disponível em <<https://vimeo.com/2503257>>

³⁵ Disponível em <<https://vimeo.com/2579694>>

³⁶ Disponível em <<https://vimeo.com/2579880>>

³⁷ Disponível em <<https://vimeo.com/2735394>>

³⁸ Disponível em <<https://vimeo.com/5690854>>

³⁹ Disponível em <<https://vimeo.com/7339135>>

⁴⁰ Disponível em <<https://vimeo.com/8364077>>

⁴¹ Disponível em <<https://vimeo.com/8732631>>

⁴² Disponível em <<https://vimeo.com/15585520>>

⁴³ Disponível em <<https://vimeo.com/15679078>>

⁴⁴ Disponível em <<https://vimeo.com/18008372>>

Alternativas à definição de um instrumento

Basicamente, o que foi desenvolvido no primeiro exemplo da seção anterior (ver Definição de tempo 3.3, p. 62), recorre à uma definição customizada de amostras sonoras de um *plugin* VST. Ben Swift, co-desenvolvedor do Extempore, sugere outras maneiras de definir um instrumentos. Duas diferentes maneiras incluem a a utilização de amostras sonoras prédefinidas e organizadas⁴⁵, ou o processamento de sinal digital (DSP), através de definições de cada parâmetro do instrumento criado⁴⁶.

Definição de uma sequência de blocos

Até 1'07", uma sequência de blocos \mathcal{K}'_{ASK} é definida, sem definir quais são os \mathcal{E}'_{ASK} que constituem. Isto é, definições de funções podem ser alocadas hierarquicamente, sem definir o que está na estrutura mais interna. No caso o laço iterativo (

`pc:cb-for-each-p`

) chama por acordes (

`chords`

), cujos parâmetros são três: i)

`piano (pc:make-chord 50 70 2 (pc:diatonic 0 '- degree))`

; ii). (ainda não sabemos quais são, o que caracteriza uma memória particular já interiorizada por Sorensen; isto é, ele ensaiou essa improvisação). (que (ver Definição de tempo 3.5, p. 63)). Para cada acorde, é chamada uma função que ainda não existe, `chords`. Esta função é acompanhada de uma lista de 4 parâmetro (50, 70, 2). O último parâmetro é uma função de *callback*, ou uma função que será executada

Exemplo 3.5 (Definição da sequência de blocos)

Criação da rotina que irá executar acordes. Fonte: (SORENSEN, 2015).

```
(pc:cb-for-each-p chords piano
  (pc:make-chord 50 70 2 (pc:diatonic 0 '- degree))
  dur)
```

⁴⁵ Disponível em <<http://benswift.me/2012/10/17/loading-and-using-a-sampler/>>

⁴⁶ Disponível em <<http://benswift.me/2012/09/28/making-an-instrument/>>

Definição de blocos

Em 1'53" a função *chords* é definida de maneira válida, e os primeiros eventos sonoros são criados (ver [Definição da sequência de blocos 3.6](#)):

Exemplo 3.6 (Algoritmo que define os acordes)

Definição da função *chord*. Fonte: ([SORENSEN, 2015](#)).

```
(define chords
  (lambda (time degree dur)
    (if (member degree '(i)) (set! dur 3.0))
        (for-each (lambda (p)
                     (play-note (*metro* time) piano p
                               (+ 50 (* 20 (cos (* pi time))))
                               (*metro* 'dur dur)))
                  (pc:make-chord 50 70 2 (pc:diatonic 0 (quote -) degree)))
        (callback (*metro*) (+ time (* .5 dur))) chords (+ time dur)
        (random (assoc degree '((i vii
                                   (vii i)))) dur)))
    (chords (*metro* 'get-beat 4.0) 'i 3.0))
```

O restante da improvisação será conduzida a partir da modificação desta função, que a cada tempo, executa um conjunto de notas (inicialmente blocos musicais, que desenvolve para contrapontos floridos, ostinatos e gestos arpejo/escala), conforme é indicado pelo programador.

3.3.2 Segundo Evento Sonoro

Uma escuta investigativa desta performance considerou o resultado sonoro gerador como replicante de uma escrita contrapontística modal, não-estrita, inicialmente à duas vozes em primeira espécie, que lembra um canto religioso.

Sorensen define um tempo regular de 110 BPM durante o momento de silêncio. Os primeiros eventos sonoros que ocorrem após o momento de silêncio foram transcritos considerando a percepção dos tempos fortes e fracos. Isto é, enquanto Sorensen define um tempo regular de 110 BPM, o processo de transcrição de maneira aproximada em um tempo regular de aproximadamente 40 BPM. A partir disso, cada compasso desta transcrição são dois compassos do vídeo. Segundo o próprio vídeo, o tempo é definido por 110 BPM e cada nota inicial é de quatro tempos, mas que irá sofrer alterações.

A [Algoritmo que define os acordes 3.6](#) indica que um jogo entre primeiro grau menor, e sétimo grau diminuto será conduzido (ver [Figura 17](#)).



Figura 16 – Primeiros eventos musicais gerados a partir das primeiras estruturas válidas de código. **Fonte:** autor.

Uma pequena transcrição de uma primeira seção da improvisação é apresentada na ??: uma primeira seção da peça, que vai do 1'53" até 4'55", pode ser descrita como: um contraponto, inicialmente à duas vozes em primeira espécie (dó eólio), que sofre perturbações sistêmicas. Tais perturbações parecem ser derivadas das intervenções de Sorensen no código do programa.

Tais perturbações criam variações na acentuação, bem como adicionam novas figuras de ritmo, de maneira bastante gradual (ver ??).



Figura 17 – Primeiros perturbações sistêmicas. **Fonte:** autor.

No ponto culminante da peça podemos escutar um contraponto florido, adicionado de um ostinato com uma única nota, e algumas intervenções de gestos, arpejos ou escalas em alta velocidade.

3.3.3 Terceiro Evento Sonoro

Este conceito nos pareceu bastante similar àquele apresentado por Hiller e

Conclusão

Estimulada por artistas-programadores(as), a interiorização de um processo criativo (bricolagem) pode ter conduzido à formação de programas de pesquisas interdisciplinares no Norte global. Isto é, teorias das Artes Músicais, Audiovisuais, Corporais, Têxteis, Filosofia, e se a criatividade permitir, da Engenharia Crítica, são coisificadas para elaboração de novos conceitos. Uma classe particular de artistas-programadores(as) (MCLEAN, 2011, p. 16), o(a) improvisador(a) de códigos (*live coder*), admite conhecimentos da engenharia de computação, revisada sob o prisma de uma ou mais teorias como extensão para improvisações ou análise de registros.

O *live coder* é aquele que pratica o *live coding* como divulgado em uma página da Internet⁴⁷ e uma publicação (WARD et al., 2004); também se mantém informado das publicações que investigam o *live coding*, como em revistas acadêmicas como *Computer Music Journal*, *Leonardo*, listas de email de *softwares*, páginas do Google, e improvisa programas com fins pedagógicos, musicais, audiovisuais, têxteis e filosóficos. O improvisador de códigos pratica o *live coding*:

Programação imediata (ou: programação de conversa, programação no fluxo, programação interativa) é um paradigma que inclui a atividade de programação ela mesma como uma operação do programa. Isto significa um programa que não é tomado como ferramenta que cria primeiro, e depois é produtivo, mas um processo de construção dinâmica de descrição e conversação - escrever o código e então se tornar parte da prática musical ou experimental. (SuperCollider.ORG, 2014, Verbete JITLib)⁴⁸

Ward et al. (2004) definem a improvisação de códigos como “atividade da escrita integral (ou partes) de um programa enquanto ele é executado”⁴⁹. Blackwell e Collins (2005) enfatizam a definição do ponto de vista da linguagem de programação como instrumento musical. McLean (2006-07-30) relata o *live coding* como ferramenta para um *Disk Jockey codificado*. Sorensen e Swift (2009) pontuam a improvisação de códigos como “uma prática de performance para o qual linguagens de computador definem o meio primário de expressão artística.”⁵⁰. Para Sorensen e Gardner (2010), *live coding* (ou *livecoding*) envolve a premissa de uma programação-partitura audiovisual reativa:

⁴⁷ Disponível em <<http://www.toplap.org>>

⁴⁸ 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.*

⁴⁹ Tradução nossa de *Live coding is the activity of writing (parts of) a program while it runs.*

⁵⁰ Tradução nossa de *Live coding is a performance pratice for which computer languages define the primary means of expression..*

Livecoding é uma prática de arte computacional que envolve criação em tempo-real de programas de audiovisual generativo para performances multimídias interativas. Comumente as ações dos programadores são expostas para uma audiência por projeção do ambiente de edição. Performances de livecoding geralmente envolvem mais de um participante, e são geralmente iniciadas a partir de uma folha conceitual em branco (SORENSEN; GARDNER, 2010, p. 823)⁵¹.

Magnusson (2011), Collins (2014) sintetizam o *live coding* como improvisação audiovisual. Sorensen (2014) define como “programar sistemas de tempo-real em tempo real”⁵². Em um discussão entitulada “Wtf is livecoding”⁵³ diz que o “*Live coding* celebra a efemeridade da própria definição”⁵⁴ (ver Figura 18, p. 68).

Embora semelhantes, as definições mudam de detalhes de acordo com o contexto. Por exemplo, Ward et al. (2004) enfatizam que o código pode ser (re)composto de partes menores. McLean (2006-07-30) enfatiza algum onde um código é (re)programado. Sorensen (2014) enfatiza que modificar algo é próprio da técnica, em seus significados técnicos. O compositor Nick Collins situa que a definição nunca deve ser uma constante, e sim caracterizada em função do contexto.



Figura 18 – Definição de *live coding*: “Insira a definição aqui”. **Fonte:** Collins (2014).

Se por um lado a definição agrupa definições, o que dificulta a tarefa inicial de descrever os fundamentos objeto de pesquisa, por outro ilustra a improvisação de códigos como um *Universo de conceitos*. Neste trabalho consideramos que definições ou performances de improvisação de códigos estão contidas em diferentes *Espaços Conceituais*

⁵¹ Tradução de *Livecoding [10, 50] is a computational arts practice that involves the real-time creation of generative audiovisual software for interactive multimedia performance. Commonly the programmers' actions are exposed to the audience by projection of the editing environment. Livecoding performances often involve more than one participant, and are often commenced from a conceptual blank slate*

⁵² Tradução nossa de *programming real-time systems in real-time*.

⁵³ Disponível em <<http://lurk.org/groups/livecode/messages/topic/ofAxZpxsKFpDRLnoA48Bh>>

⁵⁴ Tradução nossa de *Live coding celebrates the ephemerality of definition itself*.

(WIGGINS, 2006; McLean, 2006). Artistas-programadores (*live coders*) transitam entre os Espaços Conceituais para criação de Sistemas Criativos (códigos, programas). Estes Sistemas Criativos são representados em diferentes Linguagens de Programação. Regras práticas conduzem o processo de escrita e exposição desta linguagem; mas não restringem o resultado (no caso da pesquisa, musical). Mas algumas categorizações musicais se destacam. Neste sentido, selecionamos um exemplo simbólico, *A Study in Keith* de Andrew Sorensen e Swift (2009)⁵⁵. Representa um caso particular que foge dos exemplos citados anteriormente, mas envolve a manutenção de uma tradição musical tonal através de um interessante esforço de *replicação do estilo*.

No Capítulo 1 selecionamos algumas abordagens de uma grande quantidade de exemplos possíveis. Foram escolhidos por manterem alguma conexão com a improvisação de códigos no contexto sonoro. De certa forma, induzimos conceitos ligados à Música. No Capítulo 2 apresentamos um modelo de formalização dEspaços conceituais observados pelo prisma do Modelo de Improvisação discutido por Alex McLean (2006). No Capítulo 3, organizamos conceitos de uma sonoridade-algoritmo inicial de *A Study in Keith* segundo este modelo. Por último uma conclusão, mais uma reflexão sobre o processo de pesquisa do que obtenção de um resultado final. Dois apêndices foram adicionados para exposição do material que estimulou o interesse pelo tema discutido.

⁵⁵ Disponível em <<https://vimeo.com/2433947>>.

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 81, 83 e 89.
- BAKER, J. E-mail, *Origins of "Blog" and "Blogger"*. Disponível em: <<http://listserv.linguistlist.org/cgi-bin/wa?A2=ind0804C&L=ADS-L&P=R16795&I=-3>>. Citado na página 33.
- 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 27 e 67.
- 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 44.
- BROWN, C.; BISCHOF, J. *INDIGENOUS TO THE NET: Early Network Music Bands in the San Francisco Bay Area*. 2002. Disponível em: <<http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html>>. Citado 3 vezes nas páginas 24, 25 e 26.
- CANUTE live in Jubez Karlsruhe Algorave. 2015–27–01. Disponível em: <<https://www.youtube.com/watch?v=uAq4BAbvRS4>>. Citado 2 vezes nas páginas vii e 10.
- CHESIRE, T. *Hacking meets clubbing with the 'algorave'*. Wired Magazine, 2013. Disponível em: <<http://www.wired.co.uk/magazine/archive/2013/09/play/algorave>>. Citado na página 8.
- CHURCH, L.; GREEN, T. Cognitive dimensions - a short tutorial. In: *Proceedings of 20th Psychology of Programming Interest Group*. [S.l.: s.n.], 2008. Citado na página 40.
- COLLINS, N. Generative music and laptop performance. v. 22, n. 4, p. 67–79, 2003. Disponível em: <<http://portal.ku.edu.tr/~megunal/articles/Generative%20Music%20and%20Laptop%20Performance.pdf>>. Citado 2 vezes nas páginas 7 e 8.
- COLLINS, N. *Origins of live coding*. Durham University, 2014. Disponível em: <<http://www.livecodenetwork.org/files/2014/05/originsoflicoding.pdf>>. Citado 2 vezes nas páginas vii e 68.
- 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.n.], 2014. p. 355–358. Disponível em: <http://nime2014.org/proceedings/papers/426_paper.pdf>. Citado na página 9.
- 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 9.

DOWNIE, M. *Choreographing the Extended Agent: performance graphics for dance theater*. phdthesis — MIT, 2005. Disponível em: <<http://openendedgroup.com/writings/downieThesis.html>>. Citado na página 12.

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 9.

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 28.

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 58.

FORTH, J.; WIGGINS, G. A.; MCLEAN, A. Unifying conceptual spaces: Concept formation in musical creative systems. *Minds & Machines*, v. 20, p. 503—532, 2010. Disponível em: <http://www.retiary.org/ls/expanding_universe/>. Citado 3 vezes nas páginas xii, xiii e xiv.

GARBOLINO, M. *Keith jarrett Disco Version 19*. [s.n.], 2014. Disponível em: <http://www.keithjarrett.org/wp-content/uploads/Discographie_Jarrett-november-2014.pdf>. Citado na página 55.

GRIFFTHS, D. *Tag Archives: algorave*. Dave's blog of art and programming, 2013. Disponível em: <<http://www.pawfal.org/dave/blog/tag/algorave/>>. Citado 2 vezes nas páginas vii e 11.

GRIFFTHS, D. *A cryptoweaving experiment*. 2015. Disponível em: <<http://kairotic.org/a-cryptoweaving-experiment>>. Citado 4 vezes nas páginas vii, 2, 4 e 5.

GRIFFTHS, D. *Weavecoding performance experiments in Cornwall*. 2015. Disponível em: <<http://www.pawfal.org/dave/blog/tag/weavecoding/>>. Citado 2 vezes nas páginas vii e 6.

HAOULI, J. E.; MANNIS, J. A. Abertura para o silêncio. In: . [S.l.: s.n.]. p. 8. Citado na página 16.

HILLER, L. A.; ISAACSON, L. M. The experimental resultes: the illiac suite. In: *Experimental music: composition with an eletronic computer*. 1st. ed. McGRAW-HILL BOOK COMPANY, INC., 1959. p. 152–164. Disponível em: <<https://archive.org/details/experimentalmusi00hill>>. Citado 2 vezes nas páginas 54 e 65.

IAZZETTA, F. *Música e mediação tecnológica*. [S.l.]: Ed. Perspectiva-Fapesp, 2009. ISBN 9.788527308724E12. Citado na página 86.

JR., J. S. J. À procura da batida perfeita: a importância do gênero musical para a análise da música popular massiva. v. 6, n. 2, p. 31–46, 2003. Citado 2 vezes nas páginas 19 e 52.

Junior, A. D. d. C.; Lee, S. W.; Essl, G. Supercopair: Collaborative live coding on supercollider through the cloud. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 152–158. ISBN 978 0 85316 340 4. Citado 2 vezes nas páginas 21 e 90.

- KOELLREUTTER, H.-J. Wu-li: Um ensaio de música experimental. p. 203–208. Citado na página 16.
- LUNHANI, G.; SCHIAVONI, F. Termpot: criação, edição de funções no navegador em tempo de execução. p. 154–159, 2015. Disponível em: <<http://compmus.ime.usp.br/sbcm2015/files/proceedings-print.pdf>>. Citado 2 vezes nas páginas 22 e 90.
- MAGNUSSON, T. Algorithms as scores: Coding live music. v. 21, p. 19–23, 2011. Citado 2 vezes nas páginas xii e 68.
- 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 28.
- 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 24.
- MATHEWS, M. V. The digital computer as a musical instrument. v. 142, n. 3592, p. 553–557, 1963. Disponível em: <<http://www.jstor.org/stable/1712380>>. Citado na página 87.
- MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 6 vezes nas páginas vii, xiv, 58, 60, 83 e 84.
- MCCARTNEY, J. SuperCollider: a new real time synthesis language. p. 257–258, 1996. Disponível em: <https://www.academia.edu/7249277/Patterns_of_movement_in_live_languages>. Citado 2 vezes nas páginas 86 e 87.
- McLean, A. *Hacking Perl in Nightclubs*. 2004. Disponível em: <<http://www.perl.com/pub/2004/08/31/livecode.html>>. Citado na página 41.
- McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: <https://www.academia.edu/467101/Music_improvisation_and_creative_systems>. Citado 8 vezes nas páginas iv, xii, xiii, xiv, 46, 47, 48 e 69.
- McLean, A. *hacking perl music*. Youtube, 2006–07–30. Disponível em: <<https://www.youtube.com/watch?v=fbefIdbSmD4>>. Citado 2 vezes nas páginas 67 e 68.
- MCLEAN, A. *Artist-Programmers and Programming Languages for the Arts*. Tese (Doutorado) — Department of Computing, Goldsmiths, University of London, October 2011. Disponível em: <<http://slab.org/writing/thesis.pdf>>. Citado 15 vezes nas páginas iv, vii, xii, xiv, 3, 5, 7, 28, 37, 38, 39, 40, 44, 54 e 67.
- McLean, A.; WIGGINS, G. *Patterns of movement in live languages*. 2009. Disponível em: <https://www.academia.edu/7249277/Patterns_of_movement_in_live_languages>. Citado 2 vezes nas páginas 22 e 26.
- McLean, A.; WIGGINS, G. *Petrol: Reactive pattern language for improvised music*. Proceedings of the International Computer Music Conference, 2010. Disponível em: <https://www.academia.edu/467094/Petrol_Reactive_pattern_language_for_improvised_music>. Citado na página 42.

- McLean, A.; WIGGINS, G. *TIDAL – PATTERN LANGUAGE FOR LIVE CODING OF MUSIC*. Centre for Cognition, Computation and Culture; Department of Computing Goldsmiths, University of London, 2010. Disponível em: <<http://smcnetwork.org/files/proceedings/2010/39.pdf>>. Citado 2 vezes nas páginas 17 e 42.
- MORI, G. Analysing live coding with ethnographical approach. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 117–124. ISBN 978 0 85316 340 4. Citado na página 1.
- 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 2 vezes nas páginas 22 e 81.
- NUNZIO, A. D. *Genesi, sviluppo e diffusione del software "MUSIC N"nella storia della composizione informatica*. phdthesis — Facoltà di Lettere e Filosofia - Università degli Studi di Bologna, 2010. Citado na página 83.
- OPENSSOUNDCONTROL.ORG. *OSC: Open Sound Control*. opensoundcontrol.org, 2002. Disponível em: <<http://opensoundcontrol.org/about>>. Citado na página 42.
- PAIVIO, A. *Mental representations: A Dual Coding Approach*. Oxford Psychology Series. [S.l.]: Oxford University Press, 1990. Citado 2 vezes nas páginas 38 e 39.
- Pressing, J. Cognitive process in improvisation. In: *Cognitive Processes in the Perception of Art*. Elsevier Science Publishers B.V. (North-Holland), 1984. p. 345–363. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0166411508623584>>. Citado 2 vezes nas páginas xii e xiii.
- 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 4 vezes nas páginas xii, xiii, 47 e 48.
- RAMSAY, S. *Algorithms are Thoughts, Chainsaws are Tools*. Vimeo, 2010. Disponível em: <<https://vimeo.com/9790850>>. Citado 4 vezes nas páginas vii, 29, 32 e 33.
- REICH, S. Music as a gradual process. In: *Writings about Music, 1965–2000*. Oxford University Press, 1968. ISBN 978-0-19-511171-2. Disponível em: <<http://ccnmtl.columbia.edu/draft/ben/feld/mod1/readings/reich.html>>. Citado 2 vezes nas páginas 28 e 86.
- RIETVELD, H. C. Bloomsbury Publishing Inc., 2013. 1–14 p. Disponível em: <<http://file.ebook777.com/005/DjCulInTheMixPowTecAndSocChaInEleDanMus.pdf>>. Citado 2 vezes nas páginas 1 e 7.
- ROADS, C. The second steim symposium on interactive composition in live electronic music. p. 45–50, 1986. Disponível em: <<http://www.jstor.org/stable/3679484>>. Citado na página 26.
- 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 21, 81 e 90.
- ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado 2 vezes nas páginas 21 e 90.

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

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

SCHLOSS, A. Using contemporary technology in live performance; the dilemma of the performer. v. 32, p. 239–242, 2003. Disponível em: <https://people.finearts.uvic.ca/~aschloss/publications/JNMR02_Dilemma_of_the_Performer.pdf>. Citado 2 vezes nas páginas 1 e 31.

SICCHIO, K. Hacking choreography: Dance and live coding. p. 31–39, 2014. Disponível em: <http://muse.jhu.edu/journals/computer_music_journal/v038/38.1.sicchio.pdf>. Citado na página 12.

SMITH, S. W. *DSP Guide - The Scientist and Engineer's Guide to Digital Signal Processing*. 2012–06. Disponível em: <<http://dspguide.com/>>. Citado na página 59.

SOARES, G. R. *Luteria Composicional de algoritmos pós-tonais v1.1FINAL*. Prévia da dissertação para a banca de qualificação para o Mestrado em Arte, Cultura e Linguagens do IAD-UFJF. — UFJF, 2015–03–13. Disponível em: <https://github.com/glerm/luteria/raw/master/LUTERIA_2015janeiro.pdf>. Citado na página 86.

SORENSEN, A. *Disklavier sessions*. Vimeo, 2013. Disponível em: <<https://vimeo.com/50061269>>. Citado 2 vezes nas páginas xii e 57.

SORENSEN, A. *Programming in Time - Live Coding for Creative Performances*. 2014. Disponível em: <<https://www.youtube.com/watch?v=Sg2BjFQnr9s>>. Citado na página 68.

SORENSEN, A. *The Disklavier Sessions*. Youtube, 2015. Disponível em: <<https://www.youtube.com/watch?v=cFEadvBeBqw>>. Citado 5 vezes nas páginas 53, 57, 62, 63 e 64.

SORENSEN, A.; GARDNER, H. Programming with time: cyber-physical programming with impromptu. p. 822–834, 2010. Disponível em: <<http://diyhpl.us/~bryan/papers2/paperbot/67845a4fb5b009259c389f90ab02c1c0.pdf>>. Citado 4 vezes nas páginas 58, 60, 67 e 68.

SORENSEN, A.; SWIFT, B. *A Study in Keith*. Vimeo, 2009. Disponível em: <<https://vimeo.com/2433947>>. Citado 7 vezes nas páginas iv, xii, 53, 54, 57, 67 e 69.

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 4 vezes nas páginas viii, 83, 84 e 85.

SuperCollider.ORG. *SuperCollider Overviews: JITLib - An overview of the Just In Time library*. 2014. Citado na página 67.

SWENSON, J. *The Rolling stone jazz record guide*. Rolling Stone Press, 1985. 219 p. ISBN 039472643-X. Disponível em: <https://openlibrary.org/books/OL2867249M/The_Rolling_stone_jazz_record_guide>. Citado na página 54.

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 52.

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 2 vezes nas páginas 19 e 52.

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 2 vezes nas páginas 44 e 45.

TOUSSAINT, G. T. The euclidean algorithm generates traditional musical rhythms. p. 47–56, 2005. Disponível em: <<http://cgm.cs.mcgill.ca/~godfried/publications/banff.pdf>>. Citado na página 42.

VIEIRA, V. et al. Vivace: A collaborative live coding language. arXiv, n. 1502, p. 16, 2015. Disponível em: <<http://arxiv.org/abs/1502.01312>>. Citado na página 21.

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 26.

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 15 vezes nas páginas xii, 1, 4, 7, 17, 22, 27, 28, 29, 30, 31, 67, 68, 81 e 88.

WIGGINS, G. A. A preliminary framework for description, analysis and comparison of creative systems. v. 19, n. 3592, p. 449–458, 2006. Disponível em: <<http://axon.cs.byu.edu/Dan/673/papers/wiggins.pdf>>. Citado 6 vezes nas páginas iv, 37, 44, 45, 46 e 69.

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 81.

APÊNDICE A – Código fonte de um Universo Conceitual como nuvem de palavras sobre o improviso de códigos

Apresentamos no exemplo A.1 o código-fonte, em linguagem python, utilizado para extração parcial do espaço conceitual da pesquisa (??, p. ??). 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 19). 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 si não resolve todos os problemas, mas auxilia na elaboração da imagem.

Exemplo 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
# ICLC2015.txt:
#   - um arquivo em modo de texto
#     convertido de um conjunto de artigos
```

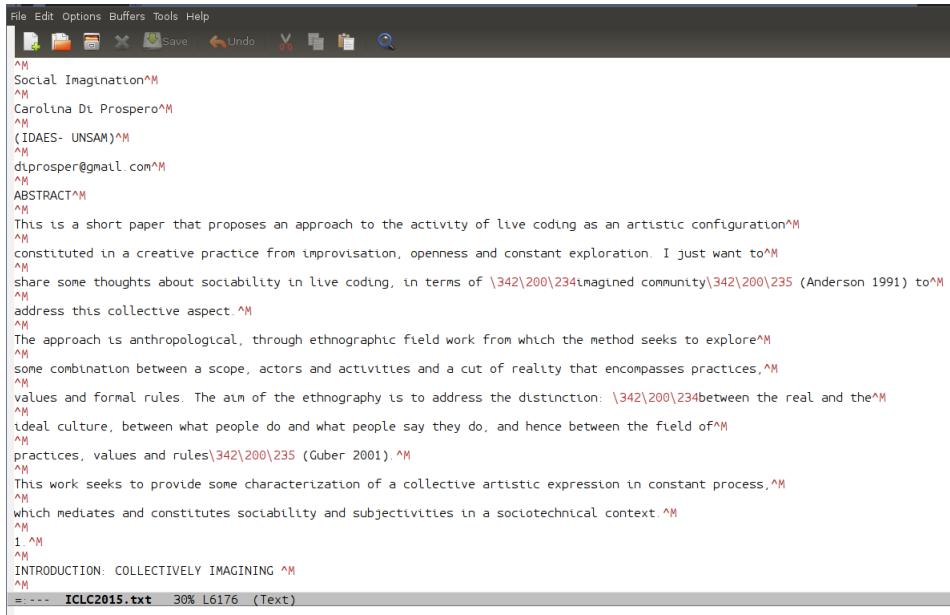


Figura 19 – Resultado da conversão do arquivo *pdf* para *txt* resulta em problemas de codificação que necessitaram ser corrigidos por comparação com o arquivo original. **Fonte:** autor.

```
# - Universo conceitual publicado
# em 2015 na Inglaterra
t = open("ICLC2015.txt", "r").read()

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

# Mostre a imagem gerada com o matplotlib,
# biblioteca para plotar imagens por dados numericos,
# levando em conta as frequencias das palavras
plt.figure()
plt.imshow(wc)
plt.axis("off")
plt.show()
```

A.1 Classes qualitativas de um Universo de conceitos do *live coding*

Com auxílio da biblioteca NLTK¹, categorizamos a nuvem de conceitos de acordo com sua função textual (ver exemplo B.1).

Exemplo A.2 (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
```

O código acima gerou uma saída textual que foi reorganizada na tabela [Tabela 5](#).

Uma breve análise da nuvem de palavras (ver ??, p. ??), pode elucidar parte das questões-satélites do *live coding*. Na [Tabela 5](#) filtrei parte dos resultados por conjuntos de

¹ Disponível em <<http://nltk.org/>>.

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 num- meral, 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, internati- onal	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

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 anteriormente, 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>>

APÊNDICE B – Sugestões Proto-históricas

Descrevemos neste capítulo um trabalho de [Mathews e Moore \(1970\)](#), GROOVE, ainda pouco observado por improvisadores de códigos e a atuação de uma tecnologia, a compilação JIT ([AYCOCK, 2003](#)) como um sujeito sócio-técnico fundamental para que o *live coding* fosse possível.

B.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 da família de softwares MUSIC N¹. É o primeiro de trabalho 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 como exemplar por sua execução instrumental e disponibilidade *online* (ver a seguir). Seu desenvolvimento iniciou em 1968 na *Bell Labs*. Segundo o próprio Mathews, o funcionamento do sistema oferece algumas possibilidades a partir de três conceitos: *criação*, *retroalimentação* e *ciberficação*. O primeiro conceito foi implementado com um sistema de arquivos, onde as funções criadas no processo criativo são memorizadas, e podem ser editadas. O segundo conceito se relaciona com o terceiro:

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 20](#)]. Assim ele é capaz de modificar as funções instantaneamente como um resultado de suas observações daqueles efeitos ([MATHEWS; MOORE, 1970, p. 715](#))².

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.

¹ Desenvolvidos a partir de 1957. As versões softwares MUSIC I, II, III, IV, IV-B, IV-BF, V (que passou por modificações no IRCAM), MUSIC 360, MUSIC 11 acarretaram no desenvolvimento do software CSound, disponível em <<https://csound.github.io/>>.

² Tradução de *GROOVE provides opportunity for immediate feedback from observations of the effects of time functions to computer inputs which compose the function. In the compose mode of the GROOVE system, a human being is in the feedback loop (...) Thus he is able to modify the functions instantaneously as a result of his observations of their effects.*

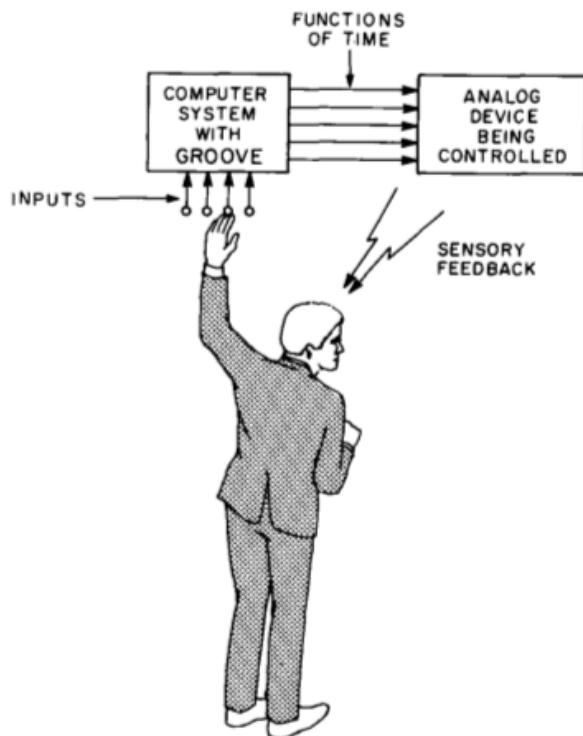


FIG. 1. Feedback loop for composing functions of time

Figura 20 – Esquema de concepção do projeto GROOVE. **Fonte:** (MATHEWS; MOORE, 1970).

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 partitura é tocada. Seus modos de influência podem ser mais variados do que aqueles que um regente convencional, que pode principalmente controlar o tempo, intensidade, e estilo (MATHEWS; MOORE, 1970, p. 715-716)³.

Como exemplo, selecionamos uma descrição da compositora Laurie Spiegel (1975) (ver Figura 21) para sumarizar as características do GROOVE, durante a produção de *The Expanding Universe*⁴, 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

³ Tradução de *The final concept is more nebulous. Since GROOVE is a man-computer system, the human engineering of the system is most important. For example, we discovered that the control of the program time needs to be quite different for composing than for editing, and the program was modified accordingly. (...) 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.*

⁴ Disponível em <<https://www.youtube.com/watch?v=dYUZmsfm4Ww>>.

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.⁵



Figura 21 – Laurie Spiegel configurando a saída analógica do GROOVE, durante a produção de *The Expanding Universe*. **Fonte:** ([SPIEGEL, 1975](#)).

Embora não declare ser uma peça minimalista, a descrição de *The Expanding Universe* considera, de maneira asséptica, os fenômenos psicoacústicos como elementos

⁵ 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.*

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 auditórios sutis dentro do som que estamos imersos. Em vez de sermos arrastados, como nas cascatas de muitas notas executadas em blocos de tempo que mudam repentinamente, tal como tantas vezes consiste a música "minimalista", abrimos nossos ouvidos mais e mais para os 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.⁶

Nesta citação podemos sumarizar um conceito para o *live coding*: Música como um Processo Gradual Cf. REICH, 1968. Porém, o significado de processo pode ser desenvolvido infinitamente. Isso não será realizado. O Processo para Spiegel é diverso daquele considerado no *live coding*, e uma digressão desta pode afastar demais o foco do trabalho principal. Para compreensão deste termo, será necessário explorar outros aspectos correlacionados no decorrer deste trabalho.

Para finalizar esta sessão, a figura Figura 21 sugere um conceito rotineiro para o *live coder*. Esta rotina é uma atividade constante de improvisação códigos para aquisição de destreza para uma performance. Iazzetta (2009), Soares (2015-03-13) lembram que esta atividade, de codificar como se construísse um instrumento musical, se caracteriza por sua conexão com a esfera composicional, nomeado *luteria composicional*.

B.2 SuperCollider

O SuperCollider⁷ é um ambiente de programação desenvolvido por James McCartney, lançado em 1996. Segundo McCartney (1996),

"SuperCollider começou como dois programas separados que eu escrevi. O primeiro foi um programa chamado *Synth-O-Matic* que era um sintetizador de tempo diferido, escrito de forma semelhante à linguagem C, para

⁶ 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.*

⁷ Disponível em <<https://superollider.github.io/>>

Machintosh em 1990 e foi abandonado. O segundo era um objeto *MAX* chamado *Pyrite* que continha um interpretador para a linguagem que se extendeu e foi usado no SuperCollider. Escrever o SuperCollider envolveu integrar uma linguagem interpretada, um coletor de lixo [administração automática da memória], e uma biblioteca de funções do Pyrite com uma máquina de síntese e funções do Synth-O-Matic. Eu gostaria de agradecer ao Curtis Roads por encorajar-me a reviver o programa do Synth-O-Matic, que levou ao SuperCollider.”⁸

Uma outra perspectiva é oferecida como uma possibilidade lógica e enxuta de outros paradigmas de programação musical. McCartney (1996, p. 1), descreve alguns problemas com o paradigma de programação musical desenvolvido a partir do Music N (MATHEWS, 1963), como por exemplo, a uma estrutura estática inerente à concepção de objetos conectados por cabos:

“As abstrações fornecidas pelas linguagens MUSIC N, incluindo o CSound, são abstrações de unidades geradoras, o laço de computação para a amostra de áudio, a representação de conexões entre unidades geradoras, e instanciamento e desalocação de instrumentos. Estas abstrações tornaram a escrita de algoritmos de processamento de sinais mais fácil, porque eles abstraem um número de detalhes incômodos. Contudo, a família Music N provê poucas estruturas de controle, nenhuma estrutura de dados reais, e nenhuma função de usuário. SAOL melhora o paradigma do Music N provendo tipos de abstrações encontradas na linguagem C, tais como estruturas de controle, funções e algumas estruturas de dados. Max, que é um tipo diferente de linguagem de programação, provê um conjunto interessante de abstrações que permite muitas pessoas usá-lo sem perceberem que estão programando acima de tudo. (...) A linguagem Max também é limitada em sua habilidade de tratar seus objetos como dados, o que torna uma estrutura de objetos estáticos. Evoluções posteriores do Max, como o jMax, e o Pd, fazem várias coisas para expandir as limitações de estrutura de dados do Max, mas ainda assim possuem uma estrutura estática de objeto.”⁹

⁸ Tradução nossa de *SuperCollider began as two separate programs that I wrote. The first was a program called Synth-O-Matic which was a non-real-time C-like synthesis programming language for the Macintosh written in 1990 and abandoned. The second was a MAX object called Pyrite which contained the interpreter for the language which was extended and used in SuperCollider. Writing SuperCollider involved integrating the language interpreter, garbage collector and function library of Pyrite with the synthesis engine and functions of Synth-O-Matic. I'd like to thank Curtis Roads for encouraging me to revive the Synth-O-Matic program which ultimately led to SuperCollider..*

⁹ Tradução nossa de *The abstractions provided by the Music N languages, including Csound (www.csounds.com), are the abstraction of a unit generator, the audio sample computation loop, the representation of the connections between unit generators, and instrument instantiation and deallocation. These abstractions make writing signal-processing algorithms easier, because they abstract a number of cumbersome details. However, the Music N family provides few control structures, no real data structures, and no user functions. SAOL (www.saol.net) improves the Music N paradigm by providing the kinds of abstractions found in the C language, such as control structures, functions, and some data structures. Max (www.cycling74.com/products/maxmsp.html), which is quite a different kind of programming language, provides an interesting set of abstractions that enable many people to use it without realizing they are programming at all. (...). The Max language is also limited in its ability to treat its own objects as data, which makes for a static object structure. Later evolutions of Max, such as jMax (www.ircam.fr/produits/logiciels/log-forum/jmax-e.html) and Pd (www.pure-data.org), do various things to expand the data structure limitations of Max but still have a generally static object structure. .*

McCartney discute adiante um modelo alternativo de notação musical, adaptado aos padrões de uma linguagem que expresse comportamentos musicais, ao invés da determinação de pontos fixos de parâmetros musicais:

“Uma lingagem musical de computador deve prover um conjunto de abstrações que expressam idéias compostoriais e de processamento de sinais da maneira mais fácil e direta possível. Os tipos de idéias que alguém pode expressar, contudo, podem ser diferentes e levar para diferentes ferramentas. Se alguém interessado em realizar uma partitura que represente uma peça musical como um artefato fixado, então o modelo de partitura/orquestra será suficiente. Motivações que levaram a projetar o SuperCollider estavam na habilidade de perceber processos sonoros que são diferentes, a cada vez que eles são tocados, para escrever peças que de alguma forma descrevem um campo de possibilidades ao invés de uma entidade fixa, e o que facilita a improvisação ao vivo por um compositor/executante.”¹⁰

Um exemplo de uso pode ilustrar o discurso do McCartney. O exemplo abaixo (p. 89) é um código de Fredrik Olofson, outro personagem importante para a improvisação de códigos (WARD et al., 2004). Este exemplo é peculiar, uma vez que expõe não somente estruturas dinâmicas e deterministas, mas possibilita criar uma conexão com dois assuntos discutidos anteriormente, algoraves (ver subseção 1.2.1, p. 8) e microchips (ver ??, p. ??). Um sintetizador recria o timbre do videogame *Atari2600* (laçado no EUA em 1977); mais especificamente é um simulador do *chip TIA* (*Television Interface Adapter*), responsável pela geração de gráficos e imagens no videogame Atari¹¹. O exemplo é relativamente simples. Um sintetizador (*SynthDef*) e um sequenciador (*Pbind*) são definidos. O sequenciador controla parâmetros como tons, frequências de modulação e panoramização do sintetizador. É interessante notar que padrões fixos (*Pseq* e *Pn*) se misturam com padrões variáveis (*Pbrown*) e segue padrões de movimentos brownianos diferentes que variam entre 28 e 31 Hz, alternados com 23 e 26 Hz. Enquanto isso a frequência moduladora segue um padrão repetitivo que alterna 10 e 16 Hz com 11 e 16 Hz.

Exemplo B.1 (Notação do SuperCollider)

Fonte: <<http://superollider.sourceforge.net/audiocode-examples/>>

```
// Simple synth definition using the Atari2600 UGen:  
(
```

¹⁰ Tradução nossa de *A computer music language should provide a set of abstractions that makes expressing compositional and signal processing ideas as easy and direct as possible. The kinds of ideas one wishes to express, however, can be quite different and lead to very different tools. If one is interested in realizing a score that represents a piece of music as a fixed artifact, then a traditional orchestra/score model will suffice. Motivations for the design of SuperCollider were the ability to realize sound processes that were different every time they are played, to write pieces in a way that describes a range of possibilities rather than a fixed entity, and to facilitate live improvisation by a composer/performer.*

¹¹ Disponível em <https://www.atariage.com/2600/archives/schematics_tia/index.html>

```

SynthDef(\atari2600, {|out= 0, gate= 1, tone0= 5,
tone1= 8, freq0= 10, freq1= 20, amp= 1, pan= 0|
  var e, z;
  e= EnvGen.kr(Env.asr(0.01, amp, 0.05), gate, doneAction:2)
  ;
  z= Atari2600.ar(tone0, tone1, freq0, freq1, 15, 15);
  Out.ar(out, Pan2.ar(z*e, pan));
}).store
)

// And a pattern to play it:
(
Pbind(
  \instrument, \atari2600,
  \dur, Pseq([0.25, 0.25, 0.25, 0.45], inf),
  \amp, 0.8,
  \tone0, Pseq([Pseq([2, 5], 32), Pseq([3, 5], 32)], inf),
  \tone1, 14,
  \freq0, Pseq([Pbrown(28, 31, 1, 32), Pbrown(23, 26, 3, 32)
    ], inf),
  \freq1, Pseq([Pn(10, 16), Pn(11, 16)], inf)
).play
)

```

B.2.1 Just In Time Library (JITLib)

A reflexividade (ver [subseção 1.3.3.1](#), p. 22) é uma característica de diversos ambientes de *live coding*. Segundo [Aycock \(2003\)](#), os primeiros programas JIT foram Genesis (com base no LISP, 1960), LC² (*Language for Conversational Computing*, 1968) e APL (1970). Este último deu origem ao conceito *lazy evaluation* (avaliação preguiçosa).

O *SuperCollider* foi o primeiro dos ambientes de programação musical a implementar esta característica. Com a divulgação da biblioteca JITLib¹², os primeiros Espaços Conceituais do *live coding* se estruturaram de maneira bastante formal na comunidade de músicos-programadores. Isto é, durante o ato de codificação podemos codificar a execução de um som antes mesmo de defini-lo (ver [Notação do SuperCollider B.2](#)).

¹² Disponível em <<http://doc.sccode.org/Overviews/JITLib.html>>.

Exemplo B.2 (Exemplo de avaliação preguiçosa no *Supercollider*.)

Um tipo de variável específica começa com o caractere ~ para indicar um ambiente propício para a avaliação preguiçosa. Mesmo antes de sua definição, podemos tocar um sintetizador:

```
// play some output to the hardware busses,
// this could be any audio rate key.

~out.play;
~out = { SinOsc.ar([400, 408] * 0.8, 0, 0.2) };
```

Depois que o código acima é escrito e executado, podemos escrever outros códigos para substituir o sintetizador durante sua execução (*runtime*):

```
// replacing the node.

// the crossfade envelope is created internally.

~out = { SinOsc.ar([443, 600 - Rand(0,200)], 0, 0.2) };
~out = { Resonz.ar(Saw.ar(40 + [0,0.2], 1), [1200, 1600], 0.1)
         + SinOsc.ar(60 * [1,1.1], 0, 0.2) };
~out = { Pan2.ar(PinkNoise.ar(0.1), LFClipNoise.kr(2)) };
```

Fonte: <http://doc.sccode.org/Tutorials/JITLib/proxyspace_examples.html>

Outros *softwares* e ambientes também merecem menção: *ixiLang*¹³ *ChuckK*¹⁴, *Extempore*¹⁵, *Impromptu*¹⁶, *SonicPi*¹⁷

Esta técnica têm sido largamente implementada em navegadores de internet (ROBERTS; WAKEFIELD; WRIGHT, 2013), ou remotamente (Junior; Lee; Essl, 2015). Isto é, entre duas pessoas distantes uma da outra, mas concetadas através da *internet* ou de redes privadas.

Trabalhos neste caminho incluem o *Gibber*¹⁸ e *Wavepot*¹⁹. Com base neste último, implementamos um ambiente chamado *Termpot*²⁰.

¹³ Disponível em <<http://www.ixi-audio.net/ixilang/>>

¹⁴ Disponível em <<http://chuck.cs.princeton.edu/>>.

¹⁵ Disponível em <<http://benswift.me/extempore-docs/>>.

¹⁶ Disponível em <<http://impromptu.moso.com.au/>>

¹⁷ Disponível em <<http://sonic-pi.net/>>

¹⁸ Disponível em <<http://gibber.mat.ucsb.edu/>>. Cf. ROBERTS; KUCHERA-MORIN, 2012

¹⁹ Disponível em <<https://www.wavepot.com>>.

²⁰ Disponível em <<https://jahpd.github.io/termpot>>. Cf. LUNHANI; SCHIAVONI, 2015.