

Resumo

Este documento é uma nota para interessados em improvisação de códigos. Talvez mais para aqueles interessados em procedimentos de pedagogia musical do código do que uma análise completa de uma ou duas obras. Indicamos eventos históricos da improvisação de códigos ligados à Música para situar a análise de um trecho exemplar (e pedagógico) desta história. Buscamos no Sistema Criativo de [McLean \(2006\)](#), [Wiggins \(2006\)](#) analisar o primeiro algoritmo gerador de uma sonoridade tonal em *A Study in Keith* de Andrew Sorensen e Swift ([2009](#)). Buscamos oferecer um texto em língua portuguesa sobre o *live coding*, através de um exemplo que consideramos representativo por: i) uma manutenção de conhecimentos musicais tonais; ii) sua pedagogia de programação através da música; iii) uma possibilidade de transposição destes algoritmos para o piano.

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

Sumário

Leia me	vii
Introdução	ix
1 TRABALHOS RELACIONADOS	1
1.1 Tecelagem	1
1.1.1 Charles Babbage e Joseph-Marie Jacquard	2
1.1.2 Weavecoding	3
1.2 Dança	6
1.2.1 Algorave	6
1.2.2 Coreografia	12
1.3 Música	13
1.3.1 Telepresença e espaços virtuais	14
1.3.2 Proto-história	15
1.3.3 Pietro Grossi	16
1.3.3.1 Just In Time Library (JITLib)	17
1.3.4 Baía de São Franscisco	18
1.3.5 Ron Kuivila	20
1.3.5.1 Quebra de sistemas	21
1.3.6 LAPTOP	22
1.3.6.1 TOPLAP	23
1.3.7 <i>Show us your screens</i>	24
1.3.7.1 Algorithms are Thoughts, Chainsaws are Tools	26
1.3.7.2 Obscurantismo é perigoso, mostre-nos suas telas	28
2 METODOLOGIA	29
2.1 Imagem Mental do improvisador-programador	29
2.1.1 Tidal	32
2.2 Sistemas criativos	34
2.3 Espaços conceituais e o processo de bricolagem	35
2.3.1 Quadro Conceitual de sistemas criativos	36
2.4 O modelo de improvisação	38
2.5 Diagramação dos espaços conceituais	39
2.6 Formalização	42
2.7 Discussão	43

3	ESTUDO DE CASO	45
3.1	Objetivo	46
3.2	Referentes Opcionais	46
3.2.1	Concertos Sun Bear	46
3.2.2	NI-Akoustik Piano	49
3.2.3	Ambiente e Linguagem: Impromptu	49
3.2.4	Extempore	51
3.2.5	Scheme	51
3.3	Blocos de Eventos	53
3.3.1	Primeiro evento sonoro: a sensação de quietude	53
3.3.2	Segundo Evento Sonoro	55
3.3.3	Terceiro Evento Sonoro	57
	Conclusão	59
	Referências	63
	APÊNDICE A – CÓDIGO FONTE DE UM UNIVERSO CONCEITUAL COMO NUVEM DE PALAVRAS SOBRE O IMPROVISO DE CÓDIGOS	69
A.1	Classes qualitativas de um Universo de conceitos do <i>live coding</i>	71
	APÊNDICE B – GROOVE	75

Lista de ilustrações

Figura 1 – Cartões perfurados da máquina de tear Jacquard. Fonte: wikimedia.org	3
Figura 2 – Tecido resultante da prática <i>Weaving code</i> . Fonte: Griffths (2015a)	5
Figura 3 – Performance no Foam Kernow. Fonte: Griffths (2015b)	6
Figura 4 – Alex McLean manipulando uma Matriz de botões para tecelagem, conectado a um Raspberry Pi. Fonte: Griffths (2015b)	7
Figura 5 – Performance do duo Canute (Karlesruhe, 2015) Fonte: < https://www.youtube.com/watch?v=uAq4BAbvRS4 >	9
Figura 6 – Performance do duo Mico Rex (Londres, 2013) Fonte: < http://www.pawfal.org/dave/blog/tag/algorave/ >	10
Figura 7 – Performance do duo Mico Rex (Londres, 2013) Fonte: < http://www.pawfal.org/dave/blog/tag/algorave/ >	11
Figura 8 – Ambiente de programação e composição <i>SuperCollider</i> . Fonte: < https://superollider.github.io/ >	11
Figura 9 – Dançarina (anônima) controlada por Kate Sicchio através de uma codificação improvisada. Fonte: < https://www.youtube.com/watch?v=uAq4BAbvRS4 >	13
Figura 10 – Performance de <i>screenBashing</i> . Fonte: < https://vimeo.com/148626379 >	14
Figura 11 – Definição do significado de TOPLAP. Fonte: Ramsay (2010)	23
Figura 12 – Modelo de bricolagem para o processo criativo realizado por um artista-programador. Fonte: McLean (2011, p. 122)	30
Figura 13 – Exemplo de uma característica de viscosidade e notação secundária no PureData. Fonte: autor	32
Figura 14 – 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	39
Figura 15 – Transcrição do motivo gerador do disco Kyoto, parte 1. Fonte: autor	48
Figura 16 – Sistema de Eixos - Rotacão entre primário e secundário. Fonte: Soares (2015-03-13)	48
Figura 17 – Sistema de tomada de som para produção de um VSTi. Fonte: < http://www.native-instruments.com/en/products/komplete/keys/definitive-piano-collection/ >	49
Figura 18 – Piano Disklavier de armário, com a parte interna exposta para exibir a placa-mãe. Fonte: wikimedia.org	50
Figura 19 – Primeiros eventos musicais gerados a partir das primeiras estruturas válidas de código. Fonte: autor	54
Figura 20 – Definição do tempo base. Fonte: autor	54

Figura 21 – Criação da rotina que irá executar acordes. Fonte: autor.	55
Figura 22 – Definição da função <i>chord</i> . Fonte: autor.	55
Figura 23 – Primeiros eventos musicais gerados a partir das primeiras estruturas válidas de código. Fonte: autor.	56
Figura 24 – Primeiros perturbações sistêmicas. Fonte: autor.	56
Figura 25 – Definição de <i>live coding</i> : “Insira a definição aqui”. Fonte: Collins (2014).	60
Figura 26 – 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.	70
Figura 27 – Esquema de concepção do projeto GROOVE. Fonte: (MATHEWS; MOORE, 1970).	76
Figura 28 – Laurie Spiegel configurando a saída analógica do GROOVE, durante a produção de <i>The Expanding Universe</i> . Fonte: (SPIEGEL, 1975).	77

Leia me

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

Introdução

Em seu artigo “A filosofia à venda, a dourada ignorância e a aposta de pascal”, Santos (2008, p. 11–12) elabora 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 supõe 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 (o que não exclui monopolizações).

É possível pensar que Santos realiza uma metáfora de um Mercado contemporâneo do conhecimento. Mas Santos esclarece que este tema 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, sugerimos espelhar a metáfora para o assunto específico deste documento. As filosofias vendidas estão em uma feira chamada *live coding*, que traduzimos por improvisação de códigos. Ali vendem o tonalismo, o pós-tonalismo, o jazz, a música algorítmica, o minimalismo, *live computer music*, música ambiental, música rave, música-ruído. Além disso são vendidas teorias da tecelagem, do audiovisual, da dança, e do lado científico, as ciências históricas e ciências cognitivas. Compramos uma amostra, cujo exemplares foram divididos em três grupos (ver Objetivos, p. x).

Conceito de Pensamento Ortopédico

Alex McLean defende que a metáfora é de importância central para a linguagem, e o faz através da *teoria da Metáfora Conceitual*(LAKOFF; JOHNSON, 1980 apud MCLEAN, 2011, p. 32). Utilizamos a metáfora de Santos para exemplificar o tipo de pensamento analítico empregado por improvisadores-programadores, quando estes se posicionam em um ambiente acadêmico. Para explicar o sentido de uma metáfora, McLean sugere utilizar texto entre aspas, seguido de outro em caixa alta.

Santos utiliza três metáforas. A primeira metáfora é o PENSAMENTO ORTOPÉDICO (isto é, “o mesmo processo especialista utilizado pelo médico responsável em corrigir deformidades do corpo”)¹. A segunda é a 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). O terceiro 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.²

Objetivos

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

¹ Disponível em <<http://www.priberam.pt/dlpo/ortopedia>>

² Grifo nosso.

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 Alex [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 desta técnica em qualquer outra área, onde o fluxo criativo é essencial:

“*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

Neste trabalho vamos 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. A discussão sobre tecelagem (ver seção 1.1, p. 1) conecta uma história da computação com a prática investigada. Os exemplos de dança (ver seção 1.2, p. 6) ilustram duas situações. O primeiro é a Música Eletrônica para Dançar² (ver subseção 1.2.1, p. 6), e o segundo nega o som como resultado da improvisação de códigos (ver subseção 1.2.2, p. 12). Por último, a característica musical da improvisações de códigos será explorada do ponto de vista histórico (ver seção 1.3, p. 13).

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 (PPG-ACL UFJF). 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.”³

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 Cf. RIETVELD, 2013.

³ 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..*

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) foram escritos ambos por Ada e Babbage. O primeiro programa escrito era dedicado ao cálculo de uma sequência numérica conhecida como *Números de Bernoulli*⁴. Apenas uma parte da máquina foi construída antes da morte de Babbage⁵.

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 Figura 1, p. 3). 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,

⁴ Allan G. Broomley, *Charles Babbage's Analytical Engine*, 1838. Disponível em <<http://athena.union.edu/~hemmend/Courses/cs80/an-engine.pdf>>

⁵ Disponível em <http://www.sciencemuseum.org.uk/objects/computing_and_data_processing/1878-3.aspx>

mas com padrões perfurados descrevendo cálculos abstratos ao invés de fios têxteis.”⁶

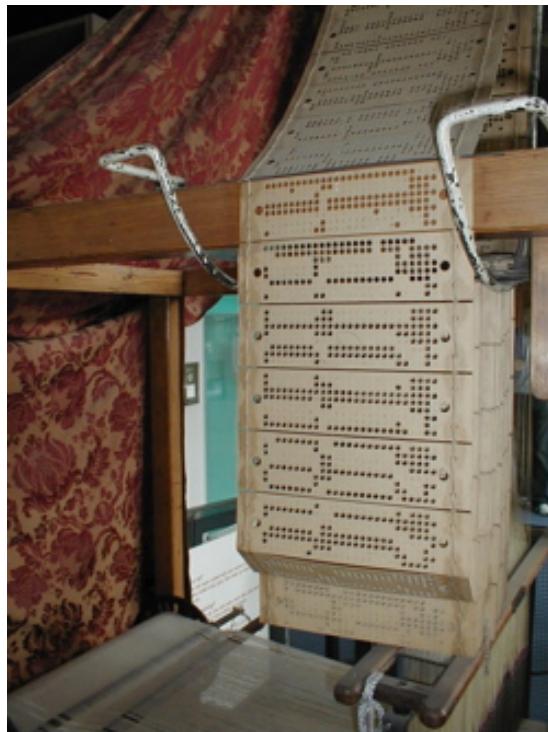


Figura 1 – Cartões perfurados da máquina de tear Jacquard. **Fonte:** wikimedia.org

1.1.2 Weavecoding

Se até hoje o mesmo sistema de Jacquard é utilizado, e considerado que a indústria têxtil possui suas aplicações computacionais, como improvisadores de códigos realizam uma bricolagem de improvisação audiovisual, computação e atividade têxtil? Exemplificamos o caso com um grupo que criou o conceito *weavecoding*. Para definí-lo, ilustramos uma investigação informal (um encontro de acadêmicos e não-acadêmicos no Foam Kernow⁷).

O grupo *Weaving codes*⁸ foi formado para investigar “padrões a partir das perspectivas de tecelagem e música, e através do desenvolvimento de uma linguagem de

⁶ 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.* .

⁷ Disponível em <<http://fo.am/kernow/>>

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

computador e código para descrever a construção de tecidos”⁹. 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 Griffths, são praticantes e organizadores de improvisações de código como artistas-programadores. A principal contribuição dos autores provavelmente foi a divulgação de uma heurística para uma improvisação com computadores, *Lubeck04*, mais conhecido como “Mostre-nos suas telas”¹⁰, dentro de um manifesto publicado como “Programação de Algoritmo Ao vivo e Organização Temporária para sua Promoção”¹¹ (WARD et al., 2004). Este tema será discutido adiante (ver [subseção 1.3.7](#), p. 24).

Do manifesto à codificação têxtil, Griffths (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 2](#). 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 2](#).)

```
(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)
  (twist 9 10))
```

⁹ 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.*

¹⁰ Tradução nossa de *Show Us Your Screens*.

¹¹ Tradução nossa de *Live Algorithm Programming and Temporary Organization for its Promotion*.

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

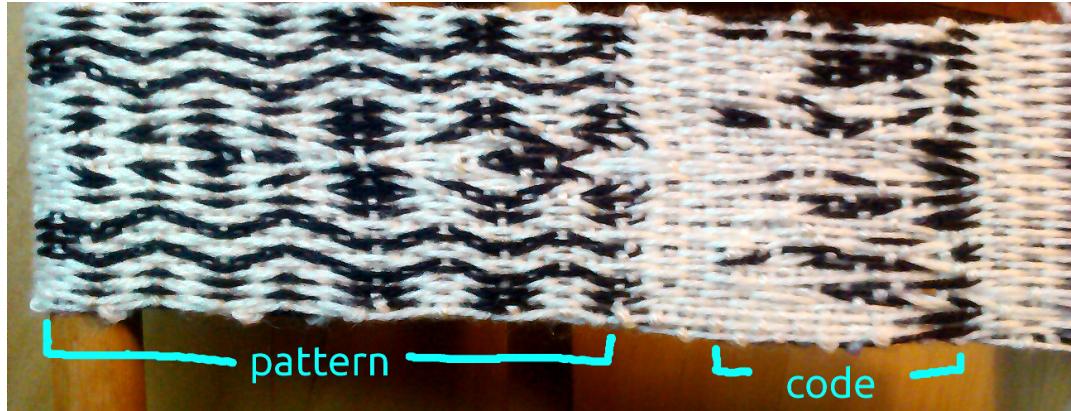


Figura 2 – Tecido resultante da prática *Weaving code*. **Fonte:** Griffiths (2015a).

Esta experiência de *weavecoding* pode ser aplicada em uma performance (ver Figura 3, p. 6). Griffiths ilustra uma performance arquetípica da improvisação de códigos: programadores escrevendo enquanto os resultados são projetados em superfícies planas. 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 4, p. 7), uma matriz de botões acopláveis, desenvolvida por Ellen Harlizius-Klück (investigadora da história da matemática, filosofia e tecelagem da Grécia Antiga na Universidade de Copenhague¹²) e Alex McLean (ver Figura 4, p. 7). Imagens em movimento foram projetadas como capturas das atividades têxteis e processadas por Griffiths.

“ 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. ”¹³

¹² Disponível em <<http://www.saumweberei.de/>>

¹³ 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. .*

está fal-
tando
ligaçao
entre
este pa-
rágrafo
e o ante-
rior



Figura 3 – Performance no Foam Kernow. **Fonte:** Griffiths (2015b).

Esta descrição de Griffiths possui uma documentação audiovisual¹⁴. É interessante notar que a menção ao grupo *Slub* auxilia a discussão musical. Por hora apenas mencionamos sua inserção no cenário da Música Eletrônica de Dança¹⁵. Subgêneros como *rave*, *gabba* e *rotterdam*, são algumas das principais categorizações musicais¹⁶ aplicadas à improvisação de códigos (ver subseção 1.2.1, p. 6).

1.2 Dança

A Dança é ilustrada de duas maneiras. A primeira é dança como fim de uma improvisação musical. 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 Cheshire (2013), cita Nick Collins e Alex McLean como os criadores do termo *algorave*. Surgiu durante uma

¹⁴ Disponível em <<https://www.youtube.com/watch?v=XrnIVUp9QgM>>

¹⁵ Cf. RIETVELD, 2013

¹⁶ Cf. JR.; Sá, 2003, 2006, 2009

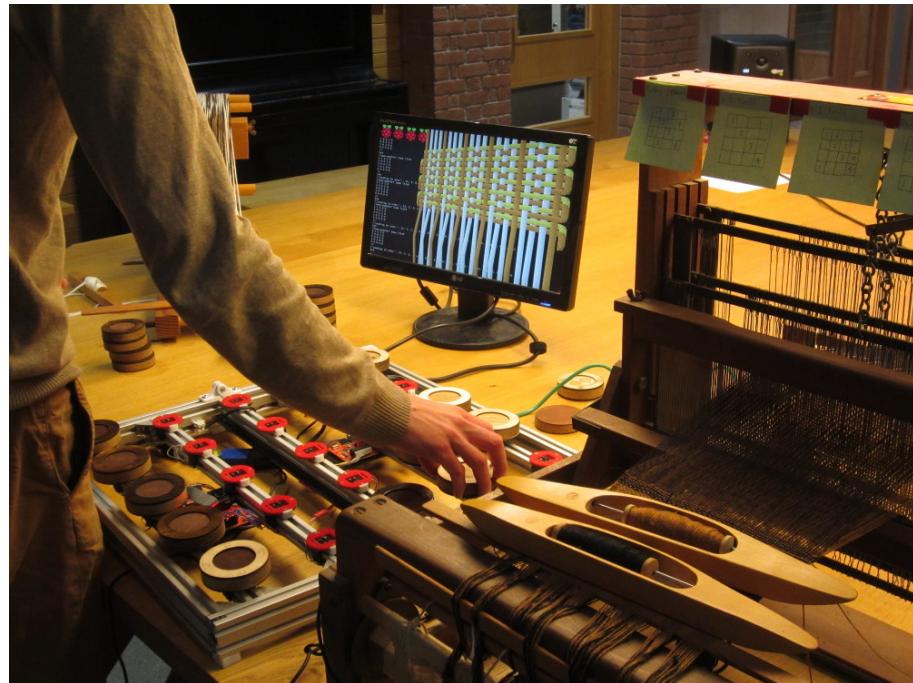


Figura 4 – Alex McLean manipulando uma Matriz de botões para tecelagem, conectado a um Raspberry Pi. **Fonte:** Griffiths (2015b).

gig (um termo utilizado no início do *jazz* para caracterizar um 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. ”¹⁷

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 subseção 1.3.6, p. 22).

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

¹⁷ 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.* .

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.”¹⁸

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.”¹⁹ 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*²⁰, e do *Supercollider*. Em 2000 o então duo *Slub*, realizam performances, autodenominadas *generative techno*, com abordagem *gabba*. Em 2001 é identificado a utilização de redes neurais para composição de padrões semelhantes ao *drum'n'bass*. Em 2004 é fundado o TOPLAP 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.

O registro audiovisual do duo Canute, Matthew Yee-King e Alex McLean, reforça improvisos de códigos arquetípicos (ver Figura 5, p. 9). Aqui a projeção segue à risca duas recomendações, “Mostre-nos suas telas” e “Obscurantismo é perigoso”(ver subseção 1.3.7.2, p. 28). Os instrumentos utilizados são um *laptop* e uma bateria eletrônica. McLean improvisa códigos, escritos no ambiente de programação *Tidal* (ver subseção 2.1.1, p. 32).

O registro audiovisual cita termos como *club* e *chordpunch*. Estes descrevem gêneros musicais para o código improvisado. É importante lembrar que McLean também utiliza outras categorizações musicais em outras performances. É curioso notar que em alguns momentos do vídeo, certas modificações nos códigos causam uma perturbação brusca em sistema de ritmos (percebido pelo 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. Sugerimos que este não é *a priori* um erro do instrumentista, mas sim de questões de performance que serão mencionados na subseção 1.3.7.

¹⁸ 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..*

¹⁹ 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>>.

²⁰ Disponível em <<https://csound.github.io/>>.



Figura 5 – Performance do duo Canute (Karlesruhe, 2015) **Fonte:** <<https://www.youtube.com/watch?v=uAq4BAbvRS4>>.

Griffiths registra uma *rave* na embarcação MS Stubnitz, em Canary Wharf, Londres, em 2013. Cárdenas e Ernesto Romero/Jorge Ramírez (Mico Rex, Figura 7) tocam neste evento. Encontramos um registro audiovisual de curta duração da apresentação do duo Mico Rex²¹, mas não de Cárdenas. Exemplos sonoros específicos estão disponíveis nas redes sociais *SoundCloud* e *Vimeo*²².

Notas descritivas especificam instrumentos e linguagens de programação. MicoRex utiliza voz e *programação* ao vivo com o ambiente de programação SuperCollider. Realiza performances de *electro-pop*, *8bits/glitch*, *electro*, *punk* e *breakz*²³. Cárdenas realiza suas performances com guitarra e programação ao vivo com o SuperCollider. É interessante lembrar que Cárdenas possui uma formação em Composição e Violão clássico na Universidade de Los Andes²⁴ e que além de recorrer ao *techno* e *dubstep*, se apropria do *noise* através de distorções, e composição mista de retroalimentação de sinais em instrumentos expandidos (como a utilização de objetos diversos na guitarra em conjunto com Processamento Digital de Sinais).

O SuperCollider²⁵ é um ambiente de programação desenvolvido por James McCartney. Realiza síntese sonora em *tempo real*(ver subseção 1.3.3.1, p. 17) e permite a programação de rotinas musicais(ver Figura 8, p. 11). O exemplo 1.2.1 um código de

²¹ Disponível em <<https://vimeo.com/65309754>>

²² <<https://soundcloud.com/tiemposdelruido>> e <<https://soundcloud.com/micorex/>>.

²³ Disponível em <<http://algorave.com/micorex/>>

²⁴ Disponível em <<http://cargocollective.com/tiemposdelruido/Alexandra-Cardenas>>

²⁵ Disponível em <<https://supercollider.github.io/>>

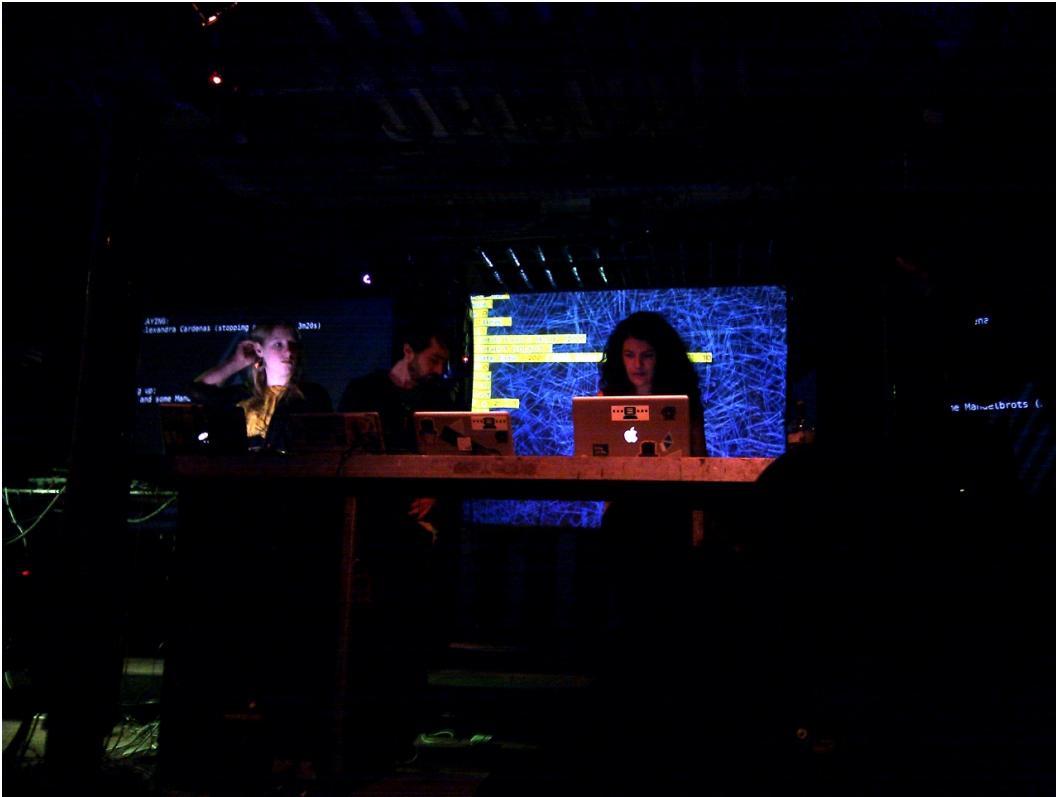


Figura 6 – Performance do duo Mico Rex (Londres, 2013) **Fonte:** <<http://www.pawfal.org/dave/blog/tag/algorave/>>.

Frederick Olofson que ilustra uma codificação de uma sonoridade de *games*, recorrente no *algorave*. 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 videogame²⁶. O exemplo é relativamente simples; a frequência portadora do sintetizador 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 1.2 (Definição de uma unidade sonora e sua execução)

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

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

²⁶ Disponível em <https://www.atariage.com/2600/archives/schematics_tia/index.html>

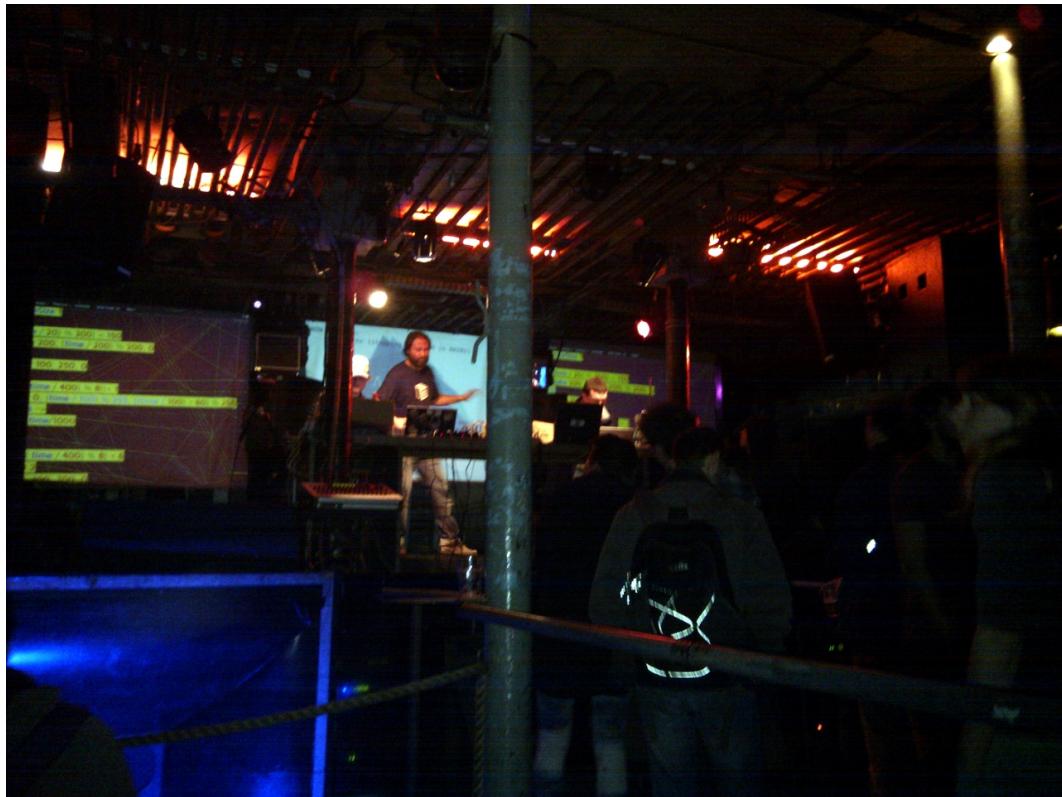


Figura 7 – Performance do duo Mico Rex (Londres, 2013) **Fonte:** <<http://www.pawfal.org/dave/blog/tag/algorave/>>.

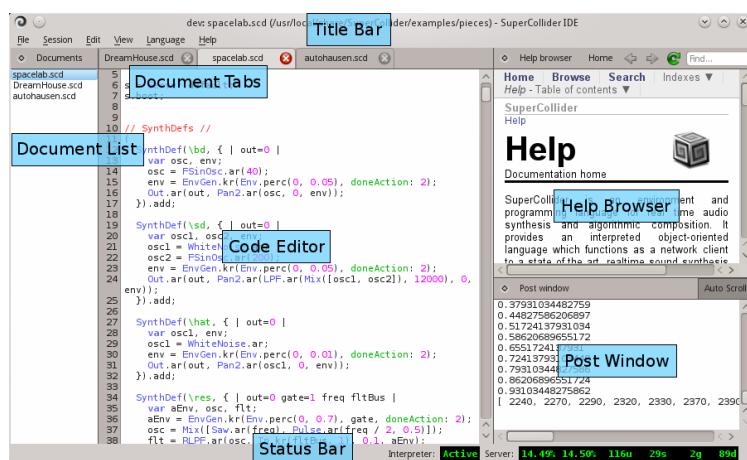


Figura 8 – Ambiente de programação e composição *SuperCollider*. **Fonte:** <<https://superollider.github.io/>>.

```

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
)

```

1.2.2 Coreografia

A segunda situação de “dança” é ilustrada na ??, com uma performance coreografada por Kate Sicchio (2015). Envolve ambientes de apresentação formal (teatros, auditórios), como na performance *HTB2.0* de Kate Sicchio²⁷. A coreografia segue uma partitura, que é codificada durante a performance. Uma bailarina improvisa movimentos conforme o que está indicado na partitura. No entanto sua conexão com a partitura não é visual ou sonora, e sim tátil:

“Esta peça é uma exploração de eletrônica codificada ao vivo e movimentos improvisados. Uma dançarina veste uma peça de atuadores táteis. Estes atuadores são programados em tempo-real via OSC²⁸ 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* ”²⁹

²⁷ Disponível em <<https://www.youtube.com/watch?v=iOAffWTBVE0>>

²⁸ 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>>

²⁹ 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.* .



Figura 9 – Dançarina (anônima) controlada por Kate Sicchio através de uma codificação improvisada. **Fonte:** <<https://www.youtube.com/watch?v=uAq4BAbvRS4>>.

É interessante notar uma sensação de quietude sonora na performance. 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. Mesmo em algumas performances de daça pesquisadas, a dança aparecia como um ornamento ou um suporte para a projeção audiovisual. 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 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.

1.3 Música

A situação “musical” será exemplificada de duas maneiras. A primeira é a performance de concerto. A segunda será exemplificada como performances telepresenciais e virtuais.

Um exemplo recente do primeiro caso é o improviso *screenBashing* de Magno Caliman (ver Figura 10), realizado durante o XIII ENCUN³⁰. Será útil para ilustração de uma “(...) performance de *livecoding* arquetípica [que] envolve programadores escrevendo

³⁰ Encontro Nacional de Compositores Universitários em Campinas-SP no ano de 2015.

códigos no palco, com suas telas projetadas para a audiência”³¹(McLean; WIGGINS, 2010b, p. 1)

O executante senta-se ao computador como um pianista, com uma penumbra usual dos sistemas de iluminação das casas de concerto. Uma tela de projeção expõe o estado atual de seu *laptop*: um editor de texto é aberto e o executante começa a programar em linguagem C, uma pequena iteração (*for loop*) que repete caracteres diversos, que serão improvisados ao longo do concerto. Assim que termina de escrever um pequeno programa, o executante abre um console (ou terminal nos sistemas operacionais Unix) e solicita para o sistema operacional (no caso um Mac OsX) compilar o programa. A compilação é um processo no qual a representação textual humana é convertida em uma linguagem apropriada para o computador. Este processo não demora, e o programa é executado. O resultado é uma sequência de caracteres de texto como texturas visuais, semi-transparentes. E tudo isso com uma sensação de quietude sonora. O executante então abre códigos preparados no ambiente de programação *SuperCollider*³² para síntese de texturas sonoras ruidosas. Um novo programa visual, na linguagem C, é improvisado e sobreposto à imagem anterior. Um novo som de textura ruidosa é adicionado. Este processo é repetido até o fim da peça. A acumulação dos sons às imagens cria um contínuo sonoro sincrético, que exige cada vez mais e mais do processamento do computador. A improvisação finaliza com um congelamento do sistema operacional, em razão deste acúmulo de memória.

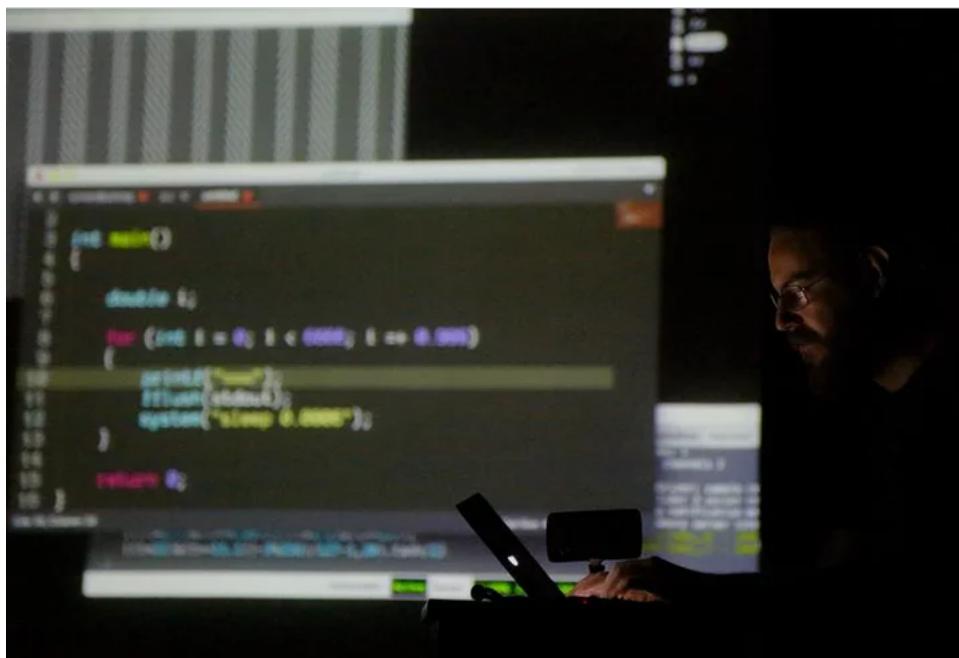


Figura 10 – Performance de *screenBashing*. Fonte: <<https://vimeo.com/148626379>>.

³¹ Tradução nossa de *The archetypal live coding performance involves programmers writing code on stage, with their screens projected for an audience.*

³² Disponível em <<https://supercollider.github.io>>

1.3.1 Telepresença e espaços virtuais

Até agora comentamos performances presenciais, mas também existem os exemplos virtuais. Podem usar imagens em movimento no caso de um *live coding* preocupado com texturas visuais. Mas no caso de uma performance “musical”, esta característica pode ser suprimida.

Uma performance virtual de *live coding* é aquela em que dois ou mais executantes, em locais diferentes mas conectados pela *internet*, compartilham do mesmo código. Por compartilharem do mesmo código podem também compartilhar do mesmo som. Um detalhe pode dividir esta performance entre telepresenciais e assíncronas. Enquanto a performance telepresencial necessita compartilhar o som digitalizado, na performance assíncrona esta característica é delegada individualmente para cada executante.

[Junior, Lee e Essl \(2015\)](#), p. 152–153) descrevem 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 Australia”³³. [Junior, Lee e Essl](#) ainda descrevem um esforço conjunto de performance remota entre O autor ainda descreve um esforço Por outro lado, execuções remotas assíncronas (isto é, entre duas ou mais pessoas, em tempos diferentes) são realizadas em softwares como *Gibber* ([ROBERTS; KUCHERA-MORIN, 2012](#))³⁴ e *Wavepot*³⁵.

1.3.2 Proto-história

O *Universo de Conceitos* do *live coding* contém numerosos exemplos. Expomos alguns *Espaços Conceituais* através de exemplos recentes. Se existem outras abordagens, musicalmente e visualmente diversas, a improvisação é o tema articulador. Seria possível dizer que cada uma cria seu próprio *Sistema Criativo* [McLean \(2006\)](#). Métodos de análise que expliquem cada uma dessas improvisações, como *Modelos de Improvisação* ainda é um campo em exploração, através de investigações de *Inteligência Artificial*. Neste sentido, este capítulo busca construir espaços conceituais, e especialmente aqueles citados como “proto-históricos”, isto é, cujas características são semelhantes (ou remeteram algo para o autor deste documento) ao conjunto de regras práticas publicadas por [Ward et al..](#)

Na [subseção 1.3.3, Mori \(2015b\)](#) descreve um caso prematuro de *live coding* na Itália, com o compositor Pietro Grossi (1917-2002). Divergente em algumas das propostas de Max Mathews, sacrificou a questão timbrística para trabalhar na questão performática.

Descrevemos, na seção [subseção 1.3.4](#), as atividades dos grupos *The Hub* e *The League of Automatic Composers* como fundamentais para o entendimento histórico do *live*

³³ Tradução nossa de *between two live coders located in Germany and United States using an SSH server located in Australia..*

³⁴ Disponível em <<http://gibber.mat.ucsb.edu/>>

³⁵ Disponível em <<http://www.wavepot.com>>

coding.

Sugerimos na subseção 1.3.3.1 apontar a tecnologia JIT (AYCOCK, 2003) como um sujeito sócio-técnico fundamental para que o *live coding* fosse possível.

Um recorte do documento-manifesto “*Live Algorithm Programming and Temporary Organization for its Promotion*”, de McLean e Wiggins, será feita na subseção 1.3.6 para discutir identidade cultural da organização TOPLAP.

Na subseção 1.3.7, “Show us your screens”, é revisto como o manifesto que define as regras práticas do *live coding*.

Na subseção 1.3.7.2, revemos a ideologia de projeção de telas.

1.3.3 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 compostionais 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 automático, baseado em algoritmos que trabalham com procedimentos “pseudo-casuais”. Existia também uma abundância de escolhas para mudanças na estrutura da peça. Um dos mais importantes aspectos do trabalho de Grossi foi que todas 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 desta forma, perdendo muito no lado da qualidade sonora. Seu desejo era poder escutar os sons resultantes imediatamente.³⁶

³⁶ Tradução nossa de (...) *the performer was able to produce and reproduce music in real time by typing some specific commands and the desired composition's parameters. The sound result came out immediately after the operator's decision, without any delay caused by calculations. There were many reproduction choices inscribed in this software: it was possible to save on the computer memory pieces of pre-existing music, to elaborate any sound material in the hard disk, to manage the music archive and to start an automated music composition process based on algorithms that worked with “pseudo-casual” procedures. There were also plenty of choices for piece structure modifications. One of the most important aspects of Grossi's work was that all the interventions were instantaneous: the operator had not to wait for the computer to finish all the requested operations and then hear the results. Data calculation and sound reproduction were simultaneous. This simultaneity was not common in the computer music field of that time and Grossi deliberately chose to work in this way, losing much on the sound quality's side. His will was to listen to the sound result immediately.*

Esta abordagem parte de uma abordagem “preguiçosa” (*lazy*). Grossi dizia sobre si mesmo, como “uma pessoa que está consciente de que o seu tempo é limitado e não quer perder tempo em fazer coisas inúteis ou na espera de alguma coisa quando não é necessário.”³⁷. 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)³⁸, 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*”³⁹, “*Permutations Of Five Sounds*”⁴⁰ e “*Continuous*”⁴¹.

A partir desta abordagem 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)”⁴².

1.3.3.1 Just In Time Library (JITLib)

A reflexividade 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 JITLib⁴³, 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 definí-lo (ver Definição de uma unidade sonora e sua execução 1.3).

³⁷ 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*.

³⁸ Disponível em <<http://www.discogs.com/Studio-Di-Fonologia-Musicale-Di-Firenze-GE-115-Computer-Concerto/release/575632>>.

³⁹ Disponível em <https://www.youtube.com/watch?v=ZQSP_wF7wSY>.

⁴⁰ Disponível em <<https://www.youtube.com/watch?v=m0WVLJ2LxeY>>.

⁴¹ Disponível em <https://www.youtube.com/watch?v=bf8jMA_zizc>.

⁴² 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..*

⁴³ Disponível em <<http://doc.sccode.org/Overviews/JITLib.html>>.

Exemplo 1.3 (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*⁴⁴ *ChuckK*⁴⁵, *Extempore*⁴⁶, *Impromptu*⁴⁷, *SonicPi*⁴⁸

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 Gibber⁴⁹ e *Wavepot*⁵⁰. Com base neste último, implementamos um ambiente chamado *Termpot*⁵¹.

⁴⁴ Disponível em <<http://www.ixi-audio.net/ixilang/>>

⁴⁵ Disponível em <<http://chuck.cs.princeton.edu/>>.

⁴⁶ Disponível em <<http://benswift.me/extempore-docs/>>.

⁴⁷ Disponível em <<http://impromptu.moso.com.au/>>

⁴⁸ Disponível em <<http://sonic-pi.net/>>

⁴⁹ Disponível em <<http://gibber.mat.ucsb.edu/>>. Cf. ROBERTS; KUCHERA-MORIN, 2012

⁵⁰ Disponível em <<https://www.wavepot.com>>.

⁵¹ Disponível em <<https://jahpd.github.io/termpot>>. Cf. LUNHANI; SCHIAVONI, 2015.

1.3.4 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-restrição, barulho, e improvisações com bordas que sobraram das revoluções contra-culturais dos anos 60(BROWN; BISCHOF, 2002, online)⁵².

Na segunda metade da década de setenta, Jim Horton começou a adquirir microcontroladores KIM-1⁵³. Segundo Brown e Bischof, não demorou para que outros interessados comprassem. Discussões informais posteriores, que incluiam, além de Horton, David (Behrman), John Bischoff, Tim Perkis, Rich Gold, Cathy Morton, Paul Robinson, e Paul Kalbach, sugeria a formação de uma “orquestra de silício” (*silicon orchestra*).

Ademais, em 1977, Horton colaborou com duas peças que interligavam estes microcontroladores. A primeira era construída sobre algoritmos inspirados nas teorias matemáticas de Leonard Euler (séc. XVIII). A segunda peça também explorava a comunicação entre os microcontroladores, de forma que “notas ocasionais da minha (Bischof) máquina faziam a máquina de Jim transpor atividades melódicas de acordo com minha nota base(BROWN; BISCHOF, 2002, online)”⁵⁴.

Em 1978, Bischof, Behrman, Gold e Horton gravaram um *Extended Play* (EP)⁵⁵ 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

⁵² 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.*

⁵³ Disponível em <<http://www.6502.org/trainers/buildkim/kim.htm>>.

⁵⁴ Tradução nossa de *the occasional tones of my machine caused Jim's machine to transpose its melodic activity according to my "key"note.* .

⁵⁵ Gravação muito longa para um demo e insuficiente para um disco, de vinil na época.

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)⁵⁶

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

Os membros da liga geralmente adaptavam composições solo para usar dentro da banda. Estes solos eram desenvolvidos independentemente por cada compositor, e eram tipicamente baseados em esquemas de algoritmos de um tipo ou outro. Existiam características de improvisação diferentes para muitas delas, como bem as músicas eram diferentes em detalhes. Teorias matemáticas, sistemas de afinação experimentais, algoritmos de inteligência artificial, projetos de instrumentos de improvisação, e performance interativa eram algumas das áreas exploradas nestes trabalhos (...) Os solos tocavam simultaneamente no cenário de grupo, se tornando "sub"-composições que interagiam, cada uma enviando e recebendo dados pertinentes para o funcionamento musical(BROWN; BISCHOF, 2002, online)⁵⁷.

⁵⁶ 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.*

⁵⁷ 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.*

1.3.5 Ron Kuivila

Segundo McLean e Wiggins (2009) comenta alguma importância a performance *Water Surfaces*, realizada na edição de 1985 da STEIM⁵⁸, em Amsterdã, como significativa para o *live coding*. A performance chamou a atenção, e foi incuída na primeira faixa do disco “*TOPLAP001 - A prehistory of live coding*”, como uma reconstrução da peça, 2007⁵⁹; 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... ”⁶⁰

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)⁶¹.

Ge Wang (2005), em uma comunicação pessoal com Curtis Roads, cita a seguinte declaração: “Eu vi o *software* FORTH de Ron Kuivila quebrar e queimar no palco em Amsterdã em 1985, mas antes disso, não fez uma música muito interessante. A performance consistiu de digitação”⁶²

Nenhuma fonte sonora foi encontrada disponível online.

1.3.5.1 Quebra de sistemas

Poucas informações sobre a peça em questão foram encontradas. Neste sentido, realizamos um pequeno resgate de outra peça apresentada no início deste capítulo, para uma aproximação conceitual.

O conceito de *término de uma performance com a quebra do sistema* é revisitado não-intencionalmente em *screenBashing* (ver Figura 10, p.14). Isto é, durante uma breve conversa após a performance, o compositor Magno Calimanão foram relatou quebras do sistema durante ensaios.

⁵⁸ *SStudio for Electro-Instrumental Music*, disponível em <<http://steim.org/about/>>.

⁵⁹ Disponível em <http://toplap.org/wiki/TOPLAP_CDs>.

⁶⁰ 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....*

⁶¹ 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*

⁶² 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..*

A performance consistiu no processo de escrita de diversos programas em linguagem C, e na execução sucessiva de códigos preparados no *SuperCollider*. Um período de silêncio antecede os primeiros resultados visuais, seguidos de resultados sonoros derivados dos códigos pré-escritos no ambiente *SuperCollider*, criando uma síntese bastante peculiar⁶³. No aspecto de sua forma, ocorreu uma aglutinação de texturas contínuas em um mesmo campo visual (a projeção da tela), e em um mesmo campo sonoro (sons ruidosos). No campo visual, a tela é pouco a pouco A aglutinação decorre em um aumento da textura sonora, e da intensidade sonora no espaço aural, .

A execução de cada um deles criou centenas processos paralelos na memória do computador (aproximadamente 1000). Contínuos sonoros e visuais terminam com um silêncio brusco, e o congelamento da imagem, decorrente desta saturação.

1.3.6 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. O seu acrônimo LAPTOP representa o principal equipamento técnico utilizado.

Este manifesto expõe os ambientes de performance, bem como alguns ritos técnicos do improvisador. O fragmento de texto abaixo é um recorte que descreve os Espaços Conceituais do *algorave* e do *Code DJing*:

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* (realizada com o iTunes, mas aparelhos reais funcionam igualmente bem). Eles passam ao HDJ [*Headphone Disk Jockey*] de acordo com este conjunto de instruções, mas no meio do caminho modificam a lista. A lista está em um retroprojetor para que o público possa acompanhar a tomada de decisão e tentar obter um melhor acesso ao processo de pensamento do compositor. (WARD et al., 2004, p. 245)⁶⁴

⁶³ “Fusão mental espontânea e irresistível, completamente livre de qualquer lógica, que acontece entre um som e uma imagem quando estas ocorrem ao mesmo tempo”Cf. CHION, , p. 68.

⁶⁴ 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*

Adiante podemos ver outros dois conceitos aglutinados: a Música de Processos⁶⁵, e a Música Generativa⁶⁶:

Contudo, alguns músicos exploram suas idéias como processos de *software*, muitas vezes ao ponto que o *software* se torna a essência da música. Neste ponto, os músicos podem ser pensados como programadores explorando seu código manifestado como som. Isso não reduz seu papel principal como um músico, mas complementa, com a perspectiva única na composição de sua música. **Termos como “música generativa” e “música de processos” tem sido inventados e apropriados para descrever esta nova perspectiva de composição.** Muita coisa é feita das supostas propriedades da chamada “música generativa” que separa o compositor do resultado do seu trabalho. Brian Eno compara o fazer da música generativa com o semear de sementes que são deixadas para crescer, e sugere abrir mão do controle dos nossos processos, deixando eles “brincarem ao vento”.⁶⁷

1.3.6.1 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 é polissêmico; isto quer dizer que as primeira, terceira e quinta letras possuem diversos significados (ver Figura 11):

“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 escurecido pelo fumo em Hamburgo à uma da manhã em 15 de Fevereiro de 2004.”⁶⁸

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.

⁶⁵ Sobre a Música como um Processo Gradual, Cf. REICH, 1968. Uma outra abordagem 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.

⁶⁶ Cf. ENO, 1996: “Música Generativa é sensitiva às circunstâncias, isso quer dizer que irá reagir diferentemente dependendo das suas condições iniciais, onde ocorre e assim por diante.”. Tradução 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.*

⁶⁷ 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”.*

⁶⁸ 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.*

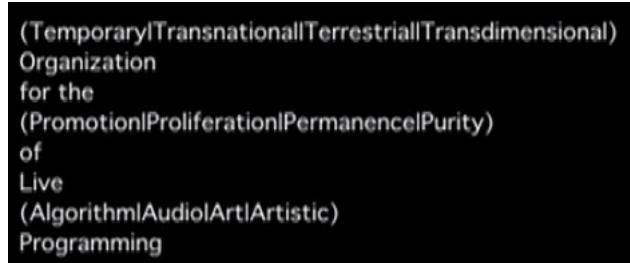


Figura 11 – Definição do significado de TOPLAP. **Fonte:** Ramsay (2010).

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 “Ps”, podem significar um ou outro termo indicado pelo algoritmo.

Este comportamento, de permitar 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. Junto com McLean, Collins cunhou o termo *Algorave*.

1.3.7 *Show us your screens*

Além das performances inaugurais nos festivais Europeus, o manifesto Lubeck04, “iniciado em um ônibus de trânsito Ryanair⁶⁹, em Hamburgo, para o aeroporto Lübeck(WARD et al., 2004, p. 247)⁷⁰, 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:

⁶⁹ Disponível em <<https://www.ryanair.com/pt/pt/>>

⁷⁰ Tradução nossa de *begun on a Ryanair transit bus from Hamburg to Lubeck airport*.

- 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. ⁷¹

Surgiu como uma resposta ao artigo “Using Contemporary Technology in Live Performance; the Dilemma of the Performer” de 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.

Poderia ser dito que o *live coding* é parte de uma construção histórica de uma tradição musical da *Computer Music Journal*. Porém o documento publicado por Ward et al. foi essencial para a divulgação de uma ideologia e de suas regras práticas. Foi escrito como resposta para um problema colocado por Schloss (2003, p. 241):

“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

⁷¹ 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, 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.

executante e instrumento que faz a mágica. Mágica é bom; muita mágica é fatal ”⁷²

Segundo Schloss (2003, p. 239), é necessário: “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”⁷³:

- “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 gesos.
- 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? ”⁷⁴

Escolhemos um ponto de interesse particular; a frase “Algoritmos são pensamentos, motosserras são ferramentas” foi muito discutida no processo de qualificação desta tese. Particularmente, existe um vídeo , em formato de *Vlog*, que discute o *live coding* sob esta perspectiva.

⁷² 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 .*

⁷³ Tradução nossa de *It's 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..*

⁷⁴ 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?*

1.3.7.1 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 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 1.3.7, p. 24; mais especificamente, o sexto item:

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 (GRIFFITHS, 2008, p. 22; item 6).

O algoritmo como pensamento é um Espaço Conceitual abstrato; 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 o algoritmo 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 timpanos ⁷⁵.

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? ⁷⁶

⁷⁵ 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..*

⁷⁶ 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 its points of interest, actually. Or what would the livecoding analog be to a non-improvisational live performance of music?*

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 expressão ⁷⁷

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.”⁷⁸.

1.3.7.2 Obscurantismo é perigoso, mostre-nos suas telas

⁷⁷ 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.*

⁷⁸ 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..*

2 Metodologia

O *Quadro Conceitual de Sistemas Criativos* (ver [subseção 2.3.1](#), p. 36) foi utilizado como método de análise para uma improvisação de códigos (ver [Capítulo 3](#), p. 45). Em termos cognitivos, este trabalho segue a aplicação de *imagens mentais* (ver [seção 2.1](#), p. 29) e *espaços conceituais* (ver [seção 2.3](#), p. 35), cuja abordagem descreve um modelo computacional para o laço cognitivo entre o que é feito, observado, e transformado.

Por exemplo, um paradigma de criatividade descrito por McLean (2011, p. 119) ilustra a *prática reflexiva* do artista plástico (no caso Paul Klee). Supondo que o pintor cria uma imagem mental do que irá fazer, perturba o vazio do quadro através do movimento inicial da mão que segura o pincel. 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. O processo continua até que o ofício seja considerado completo (*obra*).

Este paradigma não é o mesmo 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 bricolagem (ver ??, p. ??).

Em ambos os casos, na prática reflexiva e na bricolagem, são necessários conceitos que suportem a intenção (e vontade) do ofício. Isso quer dizer que conceitos são elaborados através de restrições de idéias, que podem ser construídos 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 continuamente reelaborados, começando por uma *estratégia transversal*, passando para a observação, reação, e reformulação da imagem mental.

2.1 Imagem Mental do improvisador-programador

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 por órgãos perceptivos. 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,

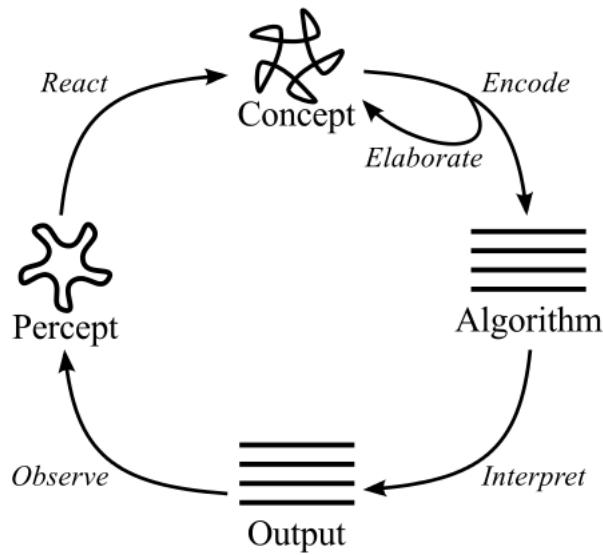


Figure 6.2: The process of action and reaction in bricolage programming

Figura 12 – Modelo de bricolagem para o processo criativo realizado por um artista-programador. **Fonte:** McLean (2011, p. 122).

mas ainda assim simboliza um conteúdo significativo.”¹ Isto é, na teoria da *Codificação Dual* (PAIVIO, 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: “humanos são capazes de compreender linguagem enquanto simultaneamente atendem às figuras [de linguagem].”²:

“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.”³

Por exemplo, quando falamos em “código de computador” não estamos nos referindo a um código escrito em uma linguagem específica. A imagem formada pode constituir na visualização mental de um conjunto de caracteres que, organizados de maneiras padronizadas, permitem a execução de alguma tarefa em algum computador.

Um sistema integrado entre códigos linguísticos e perceptuais é contextualizado por McLean no caso da improvisação de códigos (*live coding*): se atentarmos para uma

¹ Tradução nossa de *For example the use of prosodic intonation in speech is paralinguistic, not entirely notated in written text, yet symbolising meaningful content.*

² Tradução nossa de *humans are able to comprehend language while simultaneously attending to imagery..*

³ 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..*

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 destaque 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”⁴ ou DSL). Para McLean, DSLs “provêm termos padronizados para descrever demandas particulares em um domínio de uma tarefa”.(ver Tabela 1, p. 31).

Tabela 1 – Dimensões cognitivas da Notação para linguagens de programação. Fonte: (CHURCH; GREEN, 2008 apud MCLEAN, 2011).

Dimensão	Significado
Abstração	“Disponibilidade de mecanismos de abstração” ⁵
Dependências escondidas	“Invisibilidade de ligações importantes entre entidades.” ⁶
Compromisso prematuro	“Restrição na ordem de execução das coisas.” ⁷
Notação secundária	“Notação diversa da sintaxe formal.” ⁸
Viscosidade	“Resistência à mudança.” ⁹
Proximidade de mapeamento	“Proximidade de representação para o domínio-alvo.” ¹⁰
Consistência	“Semânticas similares são expressadas em formas sintáticas similares.” ¹¹
Dispersividade	“Prolixidade da linguagem.” ¹²
Tendência ao erro	“Probabilidade de erros.” ¹³
Operações mentais difíceis	“Demanda de recursos cognitivos.” ¹⁴
Provisoriedade	“Grau de compromisso com ações e marcos.” ¹⁵
Função de expressividade	“medida em que o efeito de um componente pode ser inferida.” ¹⁶

Não explicaremos todas, pois não são dimensões autônomas, o que diverge do foco de nosso trabalho. McLean exemplifica a viscosidade: a possibilidade de diferentes soluções para o mesmo resultado. Isto é, se uma Notação secundária é possível. Por exemplo, em

⁴ Tradução nossa de *DomainSpecific Languages*.

⁵ Tradução nossa de *Avaliability of abstraction mechanisms*.

⁶ Tradução nossa de *Invisibility of important links between entities..*

⁷ Tradução nossa de *Constraints on the order of doing things..*

⁸ Tradução nossa de *Notation other than formal syntax..*

⁹ Tradução nossa de *Resistance to change..*

¹⁰ Tradução nossa de *Closeness of representation to target domain..*

¹¹ Tradução nossa de *Similar semantics are expressed in similar syntactic forms.*

¹² Tradução nossa de *Verbosity of language..*

¹³ Tradução nossa de *Likelihood of mistakes..*

¹⁴ Tradução nossa de *Demand on cognitive resources..*

¹⁵ Tradução nossa de *Degree of commitment to actions or marks..*

¹⁶ Tradução nossa de *Extent to which the purpose of a component may be inferred..*

um *patch* de PD, cuja imagem mental é uma onda quadrada. Podemos utilizar objetos nativos ou objetos extendidos (ver [Figura 13](#), p. 32).

Uma dimensão notória, discutida nos corredores de universidades, são as Operações Mentais Difíceis que, devido à não-Invisibilidade de Dependências Escondidas, podem afastar o compositor de seu objetivo musical. Programas como PureData, Max/MSP e em menor grau, CSound, SuperCollider e ChucK praticam a invisibilidade de dependências em diferentes graus.

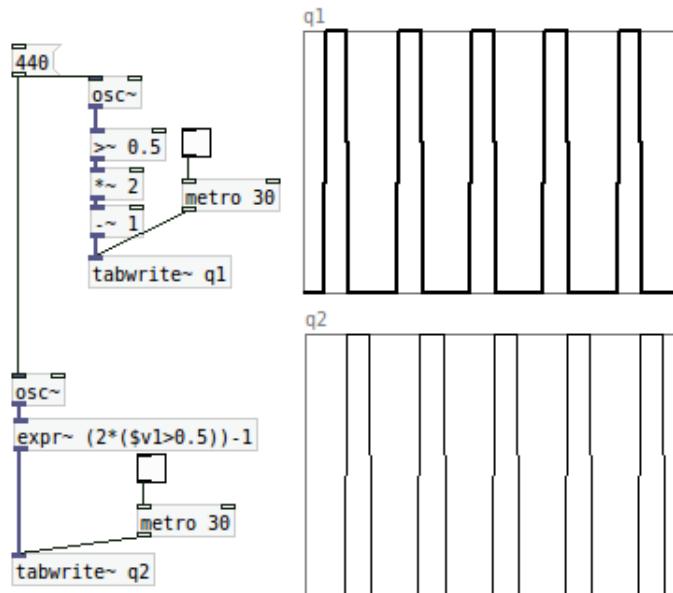


Figura 13 – Exemplo de uma característica de viscosidade e notação secundária no PureData. **Fonte:** autor.

2.1.1 Tidal

Vamos ilustrar um pequeno processo da *estratégia transversal* com o *software* Tidal.

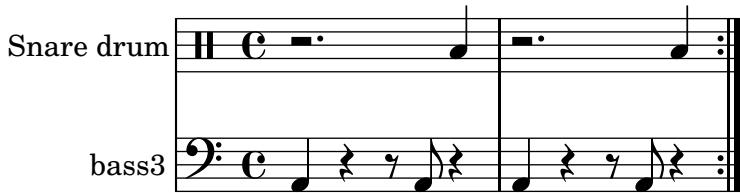
Segundo [McLean e Wiggins \(2010b, p. 2\)](#), *Tidal* é uma linguagem de composição generativa, onde “padrões podem ser compostos de numerosos subpadrões em uma variedade de maneiras e para uma profundidade arbitrária, para produzir [partes] inteiras complexas de partes simples”¹⁷. Amostras sonoras 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 “[” e “]”). Ritmos podem ser recombinados, permutados e rotacionados em padrões mais complexos. Existem efeitos de panoramização, atraso (*delay*), filtros e comunicação de rede. A citação abaixo é uma descrição técnica. No Exemplo 2.1. a imagem mental é a demanda da linguagem, que é produzir Música Eletrônica para Dançar.

¹⁷ 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.*

“Tidal é uma linguagem de padrões embebida 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 extensiva 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) ”¹⁸

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 (`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.

¹⁸ Tradução nossa de *Tidal is a pattern language embedded in the Haskell programminglanguage, 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]*.

House bass Snare drum bass3

```
-- 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, :

House bass Snare drum bass3

```
-- 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")
```

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{E}_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. 35).

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

“Boden concebe o processo de criatividade como uma identificação e/ou localização de novos objetos conceituais em um espaço conceitual.”¹⁹. 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”²⁰.

2.3 Espaços conceituais e o processo de bricolagem

Vamos retornar à Figura 12. Três processos-chave são considerados: a primeira é a formação de um espaço de conceitos diante de imagem mental que o improvisador-programador experenciou.

“A Figura 6.2 [Figura 12] 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.”²¹

¹⁹ Tradução nossa de *Boden conceives the process of creativity as the identification and/or location of new conceptual objects in a conceptual space..*

²⁰ Tradução nossa de *Any creative act is thus founded on conceptualisation or the realisation of a point within a particular ‘conceptual space’.*

²¹ 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,*

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

Criatividade	“A performance de tarefas que, quando executados por um humano, são consideradas criativas” ²²
Computação criativa	“O estudo e suporte, através de meios e métodos computacionais, do comportamento exibido por sistemas naturais e artificiais, que são considerados criativos”. ²³
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” ²⁴

2.3.1 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).²⁵

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

Espaços conceituais \mathcal{C} , finitos ou infinitos são definidos como restrições de um universo \mathcal{U} , caracterizando um conjunto não-determinístico de conhecimentos: “A noção-chave na teoria de Boden é aquele do espaço conceitual. Enquanto nenhuma definição formal é provida, é comum interpretar esta frase literalmente, tomando o espaço conceitual sendo um espaço de conceitualizações, ou representações de conceitos (Thornton, 2007, p .7).”²⁶

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. .

²² Tradução de *The performance of tasks which, if performed by a human, would be deemed creative..*

²³ 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..*

²⁴ Tradução de *One or more of the behaviours exhibited by a creative system.*

²⁵ 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*

²⁶ 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..*

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).

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) ²⁷
\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.
$[[\cdot]]$	Função de interpretação	“Uma função parcial de \mathcal{L} para funções que resultam em números reais entre [0, 1] (...) 0.5 [ou maior] significa uma verdade booleana e menos que 0.5 significa uma falsidade booleana; a necessidade disso para valores reais se tornará clara abaixo” (WIGGINS, 2006, p. 452) ²⁸
$[[\mathcal{R}]]$	Regras de validação	“Uma função que interpreta \mathcal{R} , resultando em uma função indicando aderência ao conceito em \mathcal{R} ” ²⁹
$\mathcal{C} = [[\mathcal{R}]](\mathcal{U})$	Espaço Conceitual	“Todos espaços conceituais são um subconjunto não-estrito de \mathcal{U} ” ³⁰ . 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} ³¹
\mathcal{T}	Regras de detecção	“Regras definidas dentro de \mathcal{L} para definir estratégias transversais para localizar conceitos dentro de \mathcal{U} ” (McLean, 2006) ³²
\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) ³³ “Regras definidas dentro de \mathcal{L} para avaliar a qualidade ou a desejabilidade do conceito c ” (McLean, 2006) ³⁴
$<<< \mathcal{R}, \mathcal{T}, \mathcal{E} >>>$	Função de interpretação	Uma regra necessária para definir o espaço conceitual, “independentemente da ordem, mas também, ficcionalmente, enumerá-los em uma ordem particular, sob o controle de \mathcal{T} – isto é crucial para a simulação de um comportamento criativo de um \mathcal{T} particular (WIGGINS, 2006) ³⁵ . “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} .” ³⁶

2.4 O modelo de improvisação

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

Pressing discute comportamento criativo no contexto do Modelo de Improvisação, e de fato é parte da Ferramenta de Estruturação de Sistemas Criativos. (...) Durante a transferência de notação do Modelo de Improvisação para a Ferramenta de Sistemas Criativos, nós consideramos improvisação musical de uma maneira clara e temos uma linguagem comum na qual comparar com outros modelos.³⁷.

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

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

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

²⁷ Tradução de *A universe of all possible concepts*.

²⁸ 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.

²⁹ Tradução de *A function interpreting \mathcal{R} , resulting in a function indicating adherence of a concept to \mathcal{R}* .

³⁰ Tradução de *All conceptual spaces are non-strict subset*.

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

³² Tradução de *Rules defined within \mathcal{L} to define a traversal strategy to locate concepts within \mathcal{U}* .

³³ 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.

³⁴ Tradução de *Rules defined within \mathcal{L} which evaluate the quality or desirability of a concept c* .

³⁵ 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}* .

³⁶ 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}* .

³⁷ 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*.

³⁸ *A cluster of sound events*.

³⁹ A sequence of E event clusters, where event cluster onsets do not overlap with those of a following one

⁴⁰ An improvisation, partitioned by interrupts into a number of K sequences

⁴¹ An optional referent, such as a score or stylistic norm

⁴² A set of current goals.

⁴³ Long term memory.

⁴⁴ An array of objects.

⁴⁵ An array of objects Features.

⁴⁶ An array of Process

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

Representação	Significado
E'	Um bloco de eventos sonoros ³⁸
K'	Uma seqüência de blocos de eventos E, onde um bloco de eventos não se sobrepõe com o seguinte ³⁹
I'	Uma improvisação, particionada por interrupções em um número de K sequências ⁴⁰
R'	Um referente opcional, tal como uma partitura ou uma norma estilística ⁴¹
G'	Um conjunto de objetivos ⁴²
M'	Uma memória de longo prazo ⁴³
O'	Um conjunto de objetos ⁴⁴
F'	Um conjunto de características dos objetos ⁴⁵
P'	Um conjunto de processos ⁴⁶

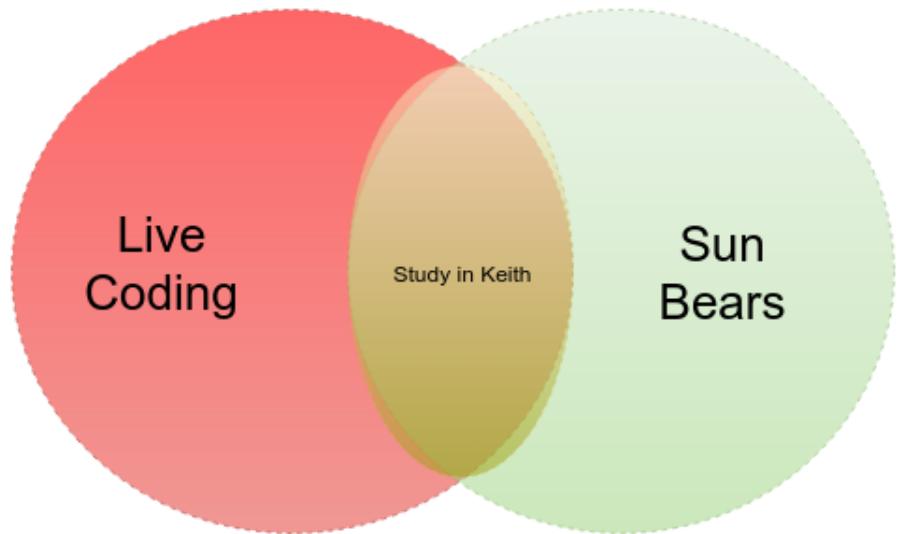


Figura 14 – Representação da justaposição entre dois espaços conceituais. A região em marrom representa um grupo de conceitos transitórios, bem como os limites desta transição. **Fonte:** autor.

2.5 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 14)

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

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

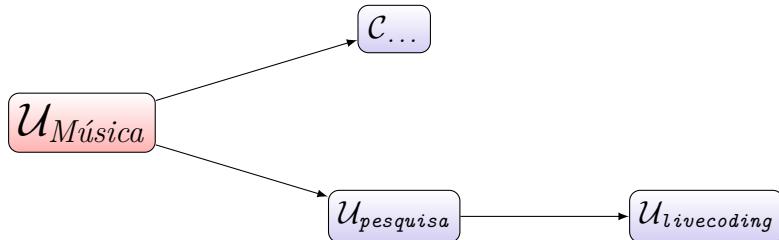
Exemplo 2.3 (Representação formal das propriedades da Figura 14)

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

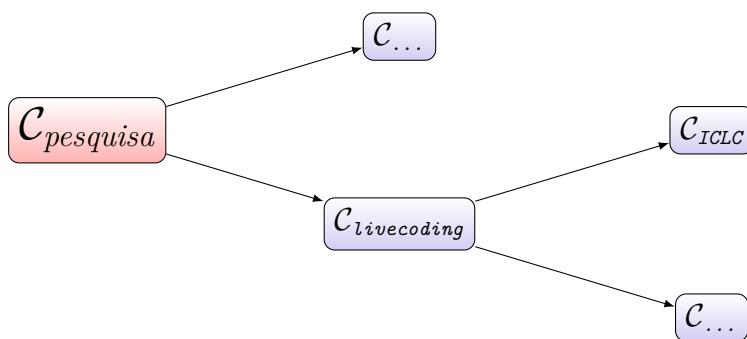
Nos diagramas abaixo, $C\dots$ representa qualquer espaço conceitual abstrato (que pode incluir outro previamente apresentado). Entre os elementos iniciais (raízes, vermelho) e transitórios (nós, azul), ocorrem as ramificações (ramos, linhas pretas), isto é, a exploração de conceitos dentro de outros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

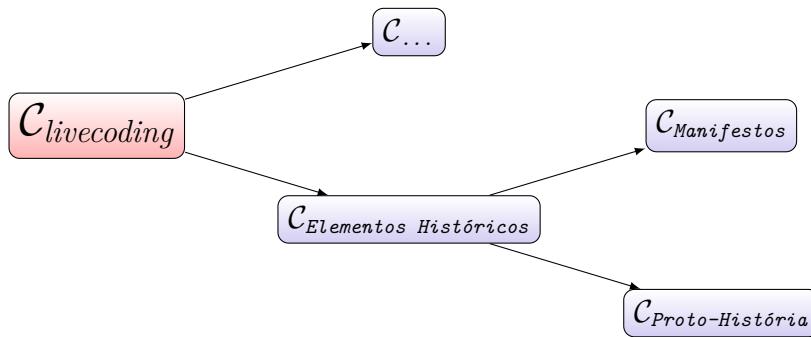
O Universo de Conceitos da pesquisa, $\mathcal{U}_{pesquisa}$, é um recorte do universo conceitual da música, $\mathcal{U}_{música}$:



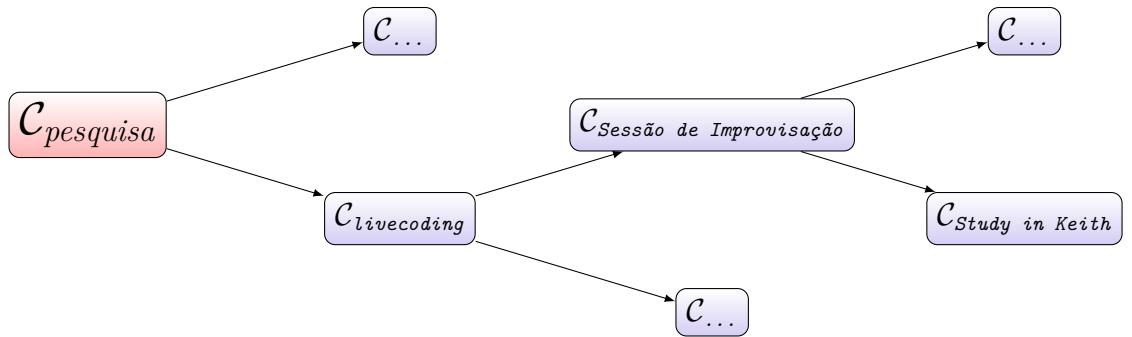
No primeiro capítulo, incluímos um subconjunto neste Espaço Conceitual da Pesquisa. Este subconjunto é constituído pelos termos representados na ?? (p. ??), e no Apêndice A (p. 69).



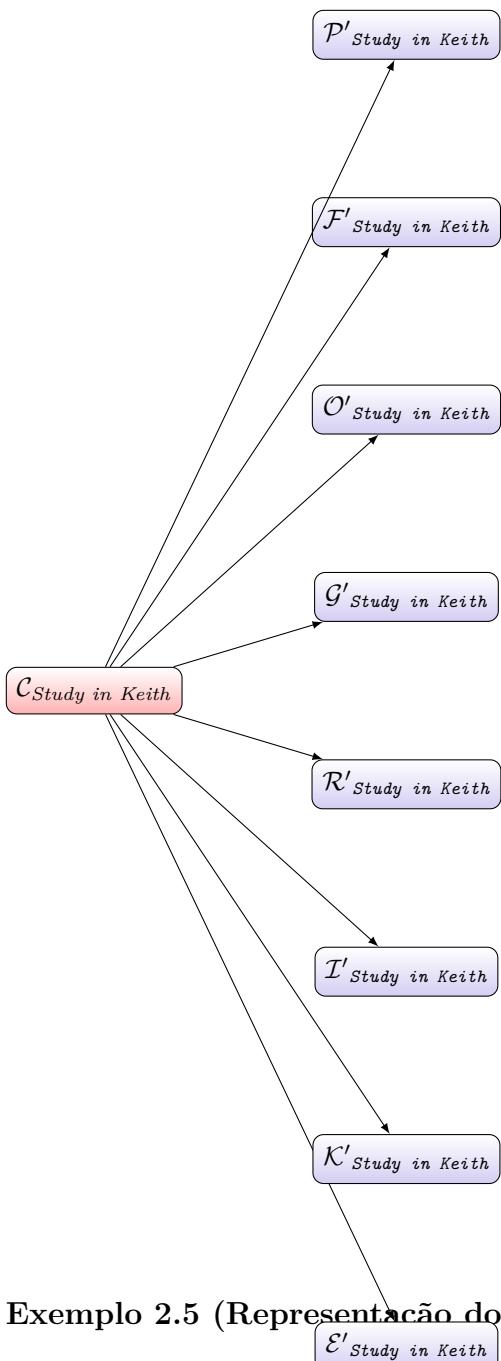
Podemos incluir elementos históricos, o período transitório entre 1970 e 2000 (*circa*), onde emanciparam as práticas e as regras heurísticas.



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, 39).



Exemplo 2.5 (Representação do modelo de improvisação para *Study in Keith*.)

2.6 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):

Exemplo 2.6 (Delimitação de regras para o *live coding* e para *Study in Keith*.)

$$\mathcal{C}_{livecoding} = <<< \mathcal{R}_{livecoding}, \mathcal{T}_{livecoding}, \mathcal{E}_{livecoding} >>> \quad (2.5)$$

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

$$\mathcal{R}_{live\ coding} = \mathcal{R}_{Proto-história} \cup \mathcal{R}_{Manifestos} \quad (2.6)$$

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

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

Expomos na equação 2.7 o espaço conceitual multidimensional do *Study in Keith*. Isto é, a aplicação de regras de validação do *live coding* e regras de validação dos Concertos *Sun Bear*:

Exemplo 2.7 (Aplicação)

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

$$\mathcal{C}_{pesquisa} = [[\mathcal{R}_{live\ coding} \cup \mathcal{R}_{Sun\ Bears}]](\mathcal{U}_{live\ coding}) \quad (2.7)$$

2.7 Discussão

A manutenção deste conhecimento, nesta forma, desenvolve um pensamento cirúrgico no conceito “criatividade na improvisação de códigos musicais”. Mas compramos a idéia por que supomos algum valor ([SANTOS, 2008](#), p. 20), “Afinal, se há compra e venda é porque as teorias e disciplinas têm alguma utilidade. Doutro modo, seriam simplesmente

deitadas ao lixo.”. O preço estabelecido não é grande, e só uma parte da teoria será utilizada para a análise de um trecho de uma improvisação musical discutido no [Capítulo 3](#).

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. 46](#)). [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. 49](#)):

“*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. 46](#)). 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. 49](#)). 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. 53](#)):

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. 53). 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 15, p. 48).

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. 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 segue, curiosamente, uma progressão de substituição

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.

The musical transcription consists of three staves of piano music. Staff 1 (measures 1-3) starts with a dynamic of *pp*, followed by *f*. Staff 2 (measures 4-6) includes dynamics *pp*, *f*, *p*, and *pp*. Staff 3 (measures 7-8) includes dynamics *f*, *mp*, *f*, and *pp*.

Figura 15 – Transcrição do motivo gerador do disco Kyoto, parte 1. **Fonte:** autor.

por tritono (este motivo está sendo considerando como parte do *gospel* supracitado): Sol Bemol Maior, Fá Maior com sexta adicionada (ou Ré menor com sétima) e Dó Maior com sétima menor. Inicialmente, consideramos esta progressão do ponto de vista de Soares (2015-03-13, p. 27), Fá sustenido Maior pode ser descrito como um *contra-polo* de Dó (ver Figura 16). Porém Dó realiza no final deste bloco uma função de dominante (V^7), o que sugere que o acorde anterior é uma subdominante relativa com a sétima no baixo (VII^7). Uma possibilidade neste tipo de análise, mais próxima da visão do jazz, seria indicar o Fá sustenido como uma $\flat V/V$ (subV), sem a sétima e acompanhado de uma nota pedal.

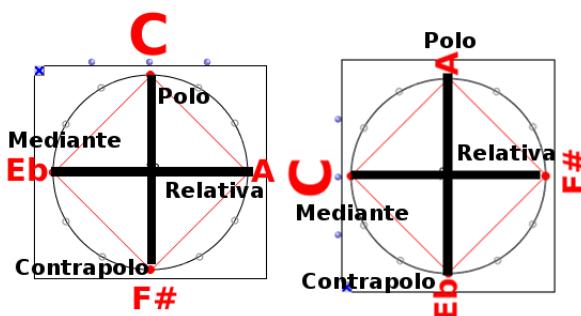


Figura 16 – Sistema de Eixos - Rotacão entre primário e secundário. **Fonte:** Soares (2015-03-13)

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 sistema de tomada de som. Na [Figura 17](#), a gravação de um outro VSTi ilustra como é realizado o processo (no caso, por Uli Baronowsky)

Figura 17 – 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 18](#), p. 50). 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*:

“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

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 linguagens 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*.

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

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

¹⁶ Tradução nossa de *In the Disklavier Sessions the programs beign written in real-time by Ben and Andrew are generating a live stream of note data which is sent to a mechanized disklavier 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 disklavier 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.* .



Figura 18 – Piano Disklavier de armário, com a parte interna exposta para exibir a placa-mãe. **Fonte:** wikimedia.org

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

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*

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*:

“*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:

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..

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/>>.

```

;; uma operacao de divisao
(/ 1 2)

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

;; execucao
(divide 1 2)

```

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 etéreo 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 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, 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.

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 Figura 19).

```
;;;;;;;;;;;;;;;;;;
;; 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/concert_grand.aupreset")
```

Figura 19 – Primeiros eventos musicais gerados a partir das primeiras estruturas válidas de código. **Fonte:** autor.

Em 0'52", Sorensen define um tempo base (ver Figura 20):

```
;;;;;;;;;;;;;;;;;;
;; 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/concert_grand.aupreset")
```

Figura 20 – Definição do tempo base. **Fonte:** autor.

Até 1'07", uma rotina que executa acordes é escrita (ver Figura 21). É interessante notar que a função *chords* ainda não foi definida.

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>>
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>>

```
;;;;;;
;; Andrew Sorensen andrew@moso.com.au

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

Figura 21 – Criação da rotina que irá executar acordes. **Fonte:** autor.

Em 1'53" a função *chords* é definida de maneira válida, e os primeiros eventos sonoros são criados (ver Figura 22):

```
;;;;;;
;; Andrew Sorensen andrew@moso.com.au

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

(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))
```

Figura 22 – Definição da função *chord*. **Fonte:** autor.

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 [Figura 22](#) indica que um jogo entre primeiro grau menor, e sétimo grau diminuto será conduzido (ver [Figura 24](#)).



Figura 23 – 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 24 – 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.

Ao final, este conjunto de eventos sonoros

Este conceito nos pareceu bastante similar àquele apresentado por [Hiller e Isaacson \(1959\)](#). Durante um breve período, algumas regras de contraponto não mantidas;

3.3.3 Terceiro Evento Sonoro

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 Internet⁴⁵ 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)⁴⁶

Ward et al. (2004) definem a improvisação de códigos como “atividade da escrita integral (ou partes) de um programa enquanto ele é executado”⁴⁷. 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.”⁴⁸. Para Sorensen e Gardner (2010), *live coding* (ou *livecoding*) envolve a premissa de uma programação-partitura audiovisual reativa:

⁴⁵ Disponível em <<http://www.toplap.org>>

⁴⁶ 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.*

⁴⁷ Tradução nossa de *Live coding is the activity of writing (parts of) a program while it runs.*

⁴⁸ 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)⁴⁹.

Magnusson (2011), Collins (2014) sintetizam o *live coding* como improvisação audiovisual. Sorensen (2014) define como “programar sistemas de tempo-real em tempo real”⁵⁰. Em um discussão entitulada “Wtf is livecoding”⁵¹ diz que o “*Live coding* celebra a efemeridade da própria definição”⁵² (ver Figura 25, p. 60).

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.

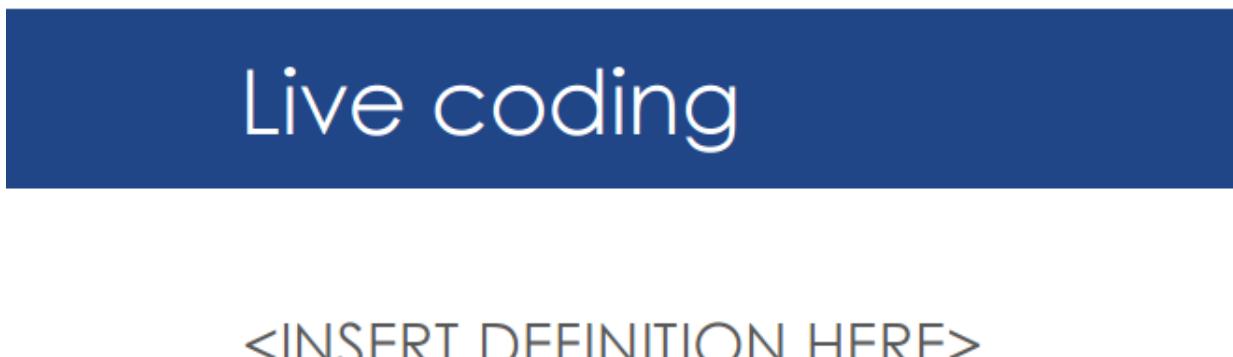


Figura 25 – Definição de *live coding*: “Insira a definição aqui”. **Fonte:** Collins (2014).

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*

⁴⁹ 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*

⁵⁰ Tradução nossa de *programming real-time systems in real-time*.

⁵¹ Disponível em <<http://lurk.org/groups/livecode/messages/topic/ofAxZpxsKFpDRLnoA48Bh>>

⁵² 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)⁵³. 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 de Espaços 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.

⁵³ 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 15, 17 e 73.
- 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 22 e 59.
- BODEN, M. *The Creative Mind: myths and mechanisms*. 2. ed. Routledge, Taylor & Francis Group, 1990. ISBN 0-203-34008-6. Disponível em: <<http://www.pauladaunt.com/books>>. Citado na página 35.
- BROWN, C.; BISCHOF, J. *INDIGENOUS TO THE NET: Early Network Music Bands in the San Francisco Bay Area*. 2002. Disponível em: <<http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html>>. Citado 2 vezes nas páginas 19 e 20.
- 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 6.
- CHION, M. Introduction to audiovisual analysis. In: *Audio-Vision: Sound on Screen*. [S.l.]: Columbia University Press. p. 29. ISBN 0-231-07898-6. Citado na página 21.
- 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 31.
- COLLINS, N. *Origins of live coding*. Durham University, 2014. Disponível em: <<http://www.livecodenetwork.org/files/2014/05/originsoflivecoding.pdf>>. Citado 2 vezes nas páginas vi e 60.
- 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 2 vezes nas páginas 7 e 8.
- 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 8.
- 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 8.
- 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 22.

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 50.

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 47.

GRIFFITHS, D. Fluxus: Scheme livecoding. 2008. Disponível em: <<http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>>. Citado na página 26.

GRIFFITHS, D. *A cryptoweaving experiment*. 2015. Disponível em: <<http://kairotic.org/a-cryptoweaving-experiment>>. Citado 4 vezes nas páginas v, 1, 4 e 5.

GRIFFITHS, D. *Weavecoding performance experiments in Cornwall*. 2015. Disponível em: <<http://www.pawfal.org/dave/blog/tag/weavecoding>>. Citado 3 vezes nas páginas v, 6 e 7.

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 46 e 56.

IAZZETTA, F. *Música e mediação tecnológica*. [S.l.]: Ed. Perspectiva-Fapesp, 2009. ISBN 9.788527308724E12. Citado na página 78.

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 6 e 43.

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 15 e 18.

LAKOFF, G.; JOHNSON, M. *Metaphors. We Live By*. 2nd. ed. [S.l.]: University of Chicago press, 1980. Citado na página x.

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 na página 18.

MAGNUSSON, T. Algorithms as scores: Coding live music. v. 21, p. 19–23, 2011. Citado na página 60.

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 22.

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 17.

- MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 7 vezes nas páginas [vi](#), [xi](#), [50](#), [51](#), [52](#), [75](#) e [76](#).
- McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: <https://www.academia.edu/467101/Music_improvisation_and_creative_systems>. Citado 7 vezes nas páginas [i](#), [15](#), [36](#), [37](#), [39](#), [60](#) e [61](#).
- McLean, A. *hacking perl music*. Youtube, 2006–07–30. Disponível em: <<https://www.youtube.com/watch?v=fbefIdbSmD4>>. Citado 2 vezes nas páginas [59](#) e [60](#).
- 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 10 vezes nas páginas [v](#), [x](#), [xi](#), [2](#), [5](#), [29](#), [30](#), [31](#), [46](#) e [59](#).
- 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 [16](#) e [20](#).
- 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 [33](#).
- 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 3 vezes nas páginas [13](#), [32](#) e [33](#).
- 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 3 vezes nas páginas [15](#), [16](#) e [73](#).
- 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 [75](#).
- OPENSOUNDCONTROL.ORG. *OSC: Open Sound Control*. opensoundcontrol.org, 2002. Disponível em: <<http://opensoundcontrol.org/about>>. Citado na página [33](#).
- PAIVIO, A. *Mental representations: A Dual Coding Approach*. Oxford Psychology Series. [S.l.]: Oxford University Press, 1990. Citado na página [30](#).
- 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 2 vezes nas páginas [38](#) e [39](#).
- RAMSAY, S. *Algorithms are Thoughts, Chainsaws are Tools*. Vimeo, 2010. Disponível em: <<https://vimeo.com/9790850>>. Citado 4 vezes nas páginas [v](#), [23](#), [26](#) e [27](#).

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 22 e 78.

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 6.

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 21.

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 15, 18 e 73.

ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado na página 18.

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

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 3 vezes nas páginas ix, x e 43.

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 na página 25.

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 51.

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 3 vezes nas páginas v, 48 e 78.

SORENSEN, A. *Disklavier sessions*. Vimeo, 2013. Disponível em: <<https://vimeo.com/50061269>>. Citado na página 49.

SORENSEN, A. *Programming in Time - Live Coding for Creative Performances*. 2014. Disponível em: <<https://www.youtube.com/watch?v=Sg2BjFQnr9s>>. Citado na página 60.

SORENSEN, A. *Weavecoding performance experiments in Cornwall*. 2015. Disponível em: <<https://www.youtube.com/watch?v=cFEadvBeBqw>>. Citado 2 vezes nas páginas 45 e 49.

- 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 50, 52, 59 e 60.
- SORENSEN, A.; SWIFT, B. *A Study in Keith*. Vimeo, 2009. Disponível em: <<https://vimeo.com/2433947>>. Citado 6 vezes nas páginas i, 45, 46, 49, 59 e 61.
- 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 vi, 75, 76 e 77.
- SuperCollider.ORG. *SuperCollider Overviews: JITLib - An overview of the Just In Time library*. 2014. Citado na página 59.
- 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 46.
- Sá, S. P. d. A música na era de suas tecnologias de reprodução. v. 12, n. 19, p. 19, 2006. Disponível em: <<http://www.compos.org.br/seer/index.php/e-compos/article/viewFile/92/92>>. Citado 2 vezes nas páginas 6 e 43.
- Sá, S. P. d. Se você gosta de madonna também vai gostar de britney! ou não? gêneros, gostos e disputa simbólica nos sistemas de recomendação musical. v. 12, n. 2, p. 1808–2599, 2009. Citado 2 vezes nas páginas 6 e 43.
- Thornton, C. A quantitative reconstruction of boden's creativity theory. p. 29, 2007. Disponível em: <<http://users.sussex.ac.uk/~christ/papers/boden-reconstruction.pdf>>. Citado 2 vezes nas páginas 35 e 36.
- 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 21.
- 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 9 vezes nas páginas 4, 15, 22, 23, 24, 25, 59, 60 e 73.
- 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 5 vezes nas páginas i, 34, 36, 37 e 60.
- 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 73.

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 26). 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
```

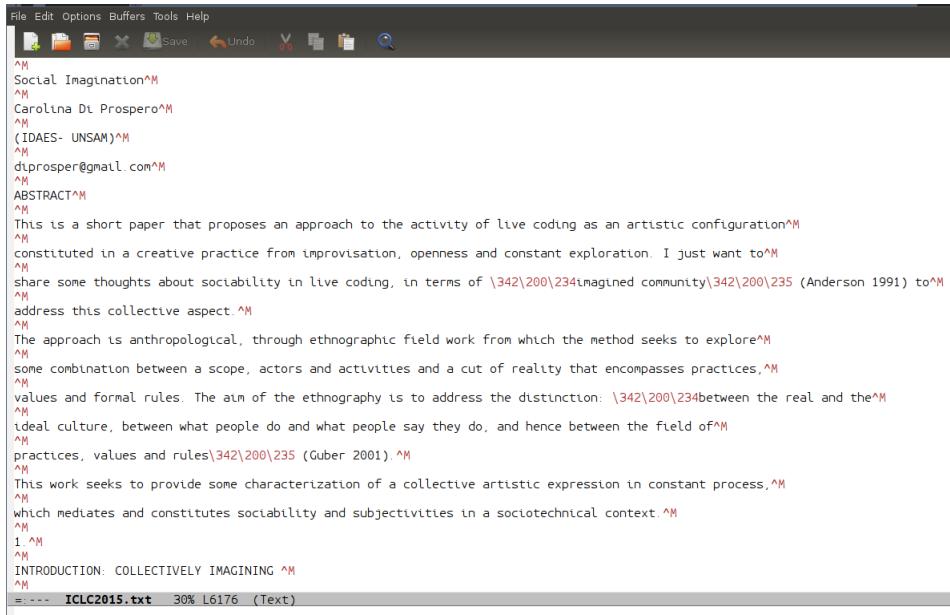


Figura 26 – Resultado da conversão do arquivo *pdf* para *txt* resulta em problemas de codificação que necessitaram ser corrigidos por comparação com o arquivo original. **Fonte:** autor.

```
# - 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 numéricos,
# levando em conta as frequências 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, international	University, conference, proceedings, network, interface, environment, text, form, context, musician, space, paper, program, audience, function, change, control, human, laptop, interaction, structure, part, session, tool, result, create, object, case, algorithm, value, development, material, set, technique, parameter, idea, screen, video, application, support, composition, piece, knowledge, feature, cell, activity, art, action, information, method, web, rule, group, need, particular, project, allow, collaboration, programmer, member, play, output	use, coder, process, state, example, way, software, research, problem, experience, design, improvisation, different, machine, pattern, audio	work, instrument	system, computer, user, language, time, practice, sound	programming	performance, code	“live coding”, music

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 – GROOVE

Descrevemos neste capítulo um trabalho de Mathews e Moore (1970), GROOVE, ainda pouco observado por improvisadores de códigos. 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.

GROOVE, ou *Generated Real-time Operations On Voltage-controlled Equipment* foi um computador desenvolvido na Bell Labs por (MATHEWS; MOORE, 1970). Alex Nunzio (2010) discute como um precedente direto do família de softwares MUSIC N¹.

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 27]. Assim ele é capaz de modificar as funções instantâneamente 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. 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

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.*

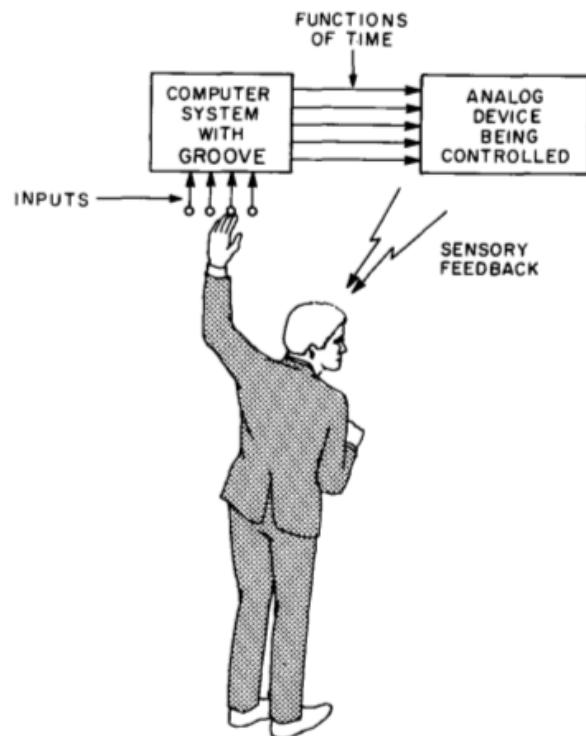


FIG. 1. Feedback loop for composing functions of time

Figura 27 – Esquema de concepção do projeto GROOVE. **Fonte:** (MATHEWS; MOORE, 1970).

do que aqueles que um regente convencional, que pode principalmente controloar o tempo, intensidade, e estilo (MATHEWS; MOORE, 1970, p. 715-716)³.

Como exemplo , selecionamos uma descrição da compositora Laurie Spiegel (1975) (ver Figura 28) 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 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

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>>.

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 28 – 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 composticionais. 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:

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.*

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 consite 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 28 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*.

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.*