

Guilherme Martins Lunhani

Live coding: ecologia de gêneros musicais na improvisação de algoritmos computacionais.

27 de junho de 2015

Guilherme Martins Lunhani

Live coding: ecologia de gêneros musicais na improvisação de algoritmos computacionais.

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

Universidade Federal de Juiz De Fora – UFJF

Instituto de Artes e Design – IAD

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

Orientador: Prof. Dr. Luiz Eduardo Castelões

27 de junho de 2015

Resumo

Este trabalho é um esqueleto da dissertação para a qualificação de mestrado em Artes Visuais, Música e Tecnologia. Apresentarei propostas de divisões de capítulos relacionadas ao seguinte tema: que tipo de músicas são, em categorias de gênero musical, improvisações feitas com computadores, uma prática que é conhecida por *live coding*. Este trabalho é um resultado parcial da pesquisa realizada. Descreverei brevemente um conjunto de conhecimentos que adquiri no percurso do ler, escrever, pensar, programar e conversar com os Professores Dr. Luiz Eduardo Castelões e Dr. Alexandre Fenerich: algumas definições, grupos, *softwares* e categorias de gêneros musicais encontradas na prática em questão.

Não é difícil entender que os primeiros computadores possuíam restrições técnicas que impediam uma performance imediatista, parecida com um instrumento musical tradicional. O manejo com máquina mais antigas por diversos pesquisadores, necessitava de um tempo consideravelmente maior que aquele que podemos experenciar: para que um código de computador pudesse resultar em som emitido por alto-falantes de um *mainframe*, foi necessária a emergência de uma comunidade de programadores, cientistas e compositores. No *live coding*, para que a mesma tarefa fosse realizada de maneira relativamente trivial, outros grupos emergiram nos anos 80 e, mais recentemente, nos últimos 15 anos.

Um grau de vivacidade com o computador, no seu uso, é presente no dia-a-dia e, por quê não, como música? Quais questões são pertinentes ao fazer musical? Observações e apontamentos preliminares dos problemas colocados nos dois parágrafos anteriores, foram planejados para 3 capítulos: 1) discussão sobre o que é *livecoding* (Parte I); 2) um panorama histórico da música realizada pelo computador, como pano de fundo para algumas questões apontadas no capítulo anterior (Parte I); 3) análise quantitativa de questões de gênero musical abordados no Capítulo 1 (Parte II).

No sumário serão utilizados os algarismos Romanos I e II para separar as partes desta dissertação e algarismos arábicos para os capítulos, seções e subseções.

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

Listas de ilustrações

Figura 1 – Diagrama de relações entre pessoas, dispositivos de controle e projeção (acústica e visual)	18
Figura 2 – Exemplo de performance de <i>livecoding</i> realizado pelo labMacambira em São Carlos - SP, onde podemos ver o código sobreposto ao vídeo. O software utilizado é o <i>Vivace</i> (VIEIRA et al., 2015). Fonte: < http://i.ytimg.com/vi/i9Lj3S1bSH4/hqdefault.jpg >	18
Figura 3 – Fonte: Esquema de concepção do projeto GROOVE descrito no artigo homônimo por Max Mathews (MATHEWS; MOORE, 1970)	24
Figura 4 – Poema traduzido para linguagem Perl. (COX; McLean; WARD, 2000, p. 4)	30
Figura 5 – Editor de texto contendo o código-fonte em Perl, uma janela de vídeo gerada pelo programa e a renderização visual realizada na janela. Fonte: < https://www.youtube.com/watch?v=BCUtu4urY9w >.	31
Figura 6 – Texto de código com uma imagem renderizada no mesmo quadro no ambiente <i>Fluxus</i> . Fonte: < http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf >	33
Figura 7 – Utilização do Fluxus pelo Kitchen Budapest com música e dança interativa através de <i>video mapping</i> . Fonte: < https://vimeo.com/30133155 >.	33
Figura 8 – Captura de tela de um frame de uma improvisação utilizando o <i>Impromptu</i>	34
Figura 9 – Ambiente de improvisação audiovisual <i>online</i> Gibber, que se aproxima da abordagem do <i>Fluxus</i>	35
Figura 10 – Diagrama, derivado da Figura 1, enfatizando relações entre pessoas, <i>hardwares</i> e dispositivo de projeção acústica, em um âmbito musical.	35
Figura 11 – Comando de DSP similares aos utilizados em uma performance de <i>live-coding</i> durante a PDCon09 < http://sussurro.musica.ufrj.br/pdcon09/ > pelo trio <i>FooBarBaz</i> . O primeiro comando é obsoleto e o segundo é sua versão funcional.	36
Figura 12 – Comando que captura o código binário da linguagem Python o comando da Figura 11 em sistemas linux	37
Figura 13 – Comando que captura o código binário da linguagem Python o comando da Figura 11 em sistemas linux	38
Figura 14 – Ambiente de improvisação audiovisual <i>Ibiniz</i> , onde resultados visuais e sonoros podem ser obtidos através de uma manipulação direta em <i>bytes</i> do computador. O código é extremamente curto. Fonte: < https://www.youtube.com/watch?v=aKMrBaXJvMs >.	39

Figura 15 – Captura de tela de um vídeo demonstrativo da peça <i>Disklavier Sessions</i> de Andrew Sorensen e Ben Swift (2013). Fonte: < https://vimeo.com/50061269 >.	40
Figura 16 – Captura de tela de um vídeo demonstrativo do ambiente <i>iXiLang</i> de Thor Magnusson (2009). Fonte: < https://www.youtube.com/watch?v=nmqEciL_O4U >.	41
Figura 17 – Diagrama de relações entre <i>performer</i> , <i>hardware/software</i> e dispositivos de projeção.	43
Figura 18 – <i>crackOP\$4</i> de André Damião, instalação multimídia: projeção, sensores de pressão, computador e sistema sonoro. Fonte: < https://vimeo.com/125100592 >.	45
Figura 19 – <i>Supercollider Live Code Session: The summer is over soon</i> por Abraham Hindle; Fonte: < https://www.youtube.com/watch?v=wJ7_5WCAj9Y >.	47
Figura 20 – Captura de tela de uma matéria jornalística a respeito da <i>Princeton Laptop Orchestra</i> (PLOrk). Esta imagem enfatiza a padronização de computadores e alto-falantes de alta-resolução, bem como ilustra uma mímica de orquestra. Fonte: < https://www.youtube.com/watch?v=q1Ymixm1xmw >, acessado em 27 de junho de 2015.	51
Figura 21 – Captura de tela de um vídeo promocional da respeito da <i>Stanford Laptop Orchestra</i> (SLOrk). Fonte: < https://www.youtube.com/watch?v=chA-4GRCb-I >, acessado em 27 de junho de 2015.	52
Figura 22 – Captura de tela de uma performance do <i>Barcelona Laptop Orchestra</i> (BLO). Esta imagem é da última seção da performance decupada, onde ocorre a manipulação dos pixels e uma experiência torrencial de eventos sonoros. Fonte: < https://www.youtube.com/watch?v=YCVKE7HG0Mg >, acessado em 27 de junho de 2015.	53
Figura 23 – <i>Texturas Interativas</i> . Ensaio da performance a ser realizada no dia 06/05/2015 no Solar da Baronesa em São João Del Rei. Fonte: < https://www.youtube.com/watch?v=6cwAQxIQ8fM >.	58
Figura 24 – Improviso com <i>drones</i> no SuperCollider Fonte: < https://www.youtube.com/watch?v=b8j4umQ2IIE >.	63
Figura 25 – Processo analítico da gramática musical dos <i>Mikrokosmos 41 e 73</i> de Bartók, e recombinação dos seus elementos melódicos e rítmicos para elaboração de um novo material musical, como um processo próprio da CAC. Fonte: (SOARES, 2015-03-13, p. 113)	65
Figura 26 – <i>Live coding</i> com Tidal Fonte: < https://www.youtube.com/watch?v=ZQqg9Ghxruo >.	66
Figura 27 – Lista de palavras-chave escolhidas para realização de uma prospecção de dados.	78

Figura 28 – Informações a respeito de X músicas procuradas no Soundcloud com respeito à localização geográfica daquele que fez a composição.	81
Figura 29 – Informações a respeito de X músicas procuradas no Soundcloud com respeito à localização geográfica daquele que fez a composição.	82
Figura 30 – Programa escrito em javascript que define uma função de fibonacci e cria uma série a partir dela, bem como uma lista de razões áureas	100
Figura 31 – Resultado do programa da Figura 30	101
Figura 32 – Programa escrito no SuperCollider que cria uma sequencia infinita de notas baseada na razão áurea.	102
Figura 33 – Programa escrito no SuperCollider que cria uma rede de delays infinita.	103
Figura 34 – Programa escrito no SuperCollider que cria uma rede de delays infinita.	103
Figura 35 – Algoritmo de um ruído branco sendo salvo no Wavepot	105
Figura 36 – Algoritmo de um ruído branco sendo salvo no Wavepot	105
Figura 37 – Duas funções básicas em um mesmo arquivo; um ruído e uma senóide.	106
Figura 38 – Síntese sonora com o <i>wavepot</i> . monoaural	106
Figura 39 – Síntese sonora com o <i>wavepot</i> . estereofônica.	106
Figura 40 – Algoritmos de processamento e gravação de uma função que pode ser chamada dinamicamente (<i>this.dsp</i>) em linguagem <i>coffeescript</i> . A função compilada em javascript está demonstrada na Figura 41.	107
Figura 41 – Função <i>dsp</i> compilada; esta será a função utilizada na figura Figura 43.	107
Figura 42 – Máquina virtual clone do Wavepot	107
Figura 43 – Algoritmos de processamento e gravação de uma função que pode ser chamada dinamicamente (<i>this.dsp</i>)	108
Figura 44 – Compilador de códigos em áudio, contendo o ambiente de áudio de funções pré-definidas pelo usuário.	108
Figura 45 – Compilador de códigos em áudio, contendo o ambiente de áudio de funções pré-definidas pelo usuário.	109
Figura 46 – Sequencia de comandos e a utilização do comando <i>def</i> para criar um ruído branco	110
Figura 47 – Criando um ruido branco e controlando a amplitude com um slider .	110
Figura 48 – Termo livecoding , analisado a partir da categoria created_at	114
Figura 49 – Termo livecoding , analisado a partir da categoria country	115
Figura 50 – Termo livecoding , analisado a partir da categoria city	116
Figura 51 – Termo livecoding , analisado a partir da categoria genre	117
Figura 52 – Termo livecoding , analisado a partir da categoria license	118
Figura 53 – Termo live-coding , analisado a partir da categoria created_at	119
Figura 54 – Termo live-coding , analisado a partir da categoria country	120
Figura 55 – Termo live-coding , analisado a partir da categoria city	121
Figura 56 – Termo live-coding , analisado a partir da categoria genre	122

Figura 57 – Termo live-coding , analisado a partir da categoria license	123
Figura 58 – Termo algorave , analisado a partir da categoria created_at	124
Figura 59 – Termo algorave , analisado a partir da categoria country	125
Figura 60 – Termo algorave , analisado a partir da categoria city	126
Figura 61 – Termo algorave , analisado a partir da categoria genre	127
Figura 62 – Termo algorave , analisado a partir da categoria license	128
Figura 63 – Termo algopop , analisado a partir da categoria created_at	129
Figura 64 – Termo algopop , analisado a partir da categoria country	130
Figura 65 – Termo algopop , analisado a partir da categoria city	131
Figura 66 – Termo algopop , analisado a partir da categoria genre	132
Figura 67 – Termo algopop , analisado a partir da categoria license	133
Figura 68 – Termo algorithmic_music , analisado a partir da categoria created_at .	134
Figura 69 – Termo algorithmic_music , analisado a partir da categoria country .	135
Figura 70 – Termo algorithmic_music , analisado a partir da categoria city . . .	136
Figura 71 – Termo algorithmic_music , analisado a partir da categoria genre . . .	137
Figura 72 – Termo algorithmic_music , analisado a partir da categoria license . .	138
Figura 73 – Termo bytebeat , analisado a partir da categoria created_at	139
Figura 74 – Termo bytebeat , analisado a partir da categoria country	140
Figura 75 – Termo bytebeat , analisado a partir da categoria city	141
Figura 76 – Termo bytebeat , analisado a partir da categoria genre	142
Figura 77 – Termo bytebeat , analisado a partir da categoria license	143
Figura 78 – Termo wavepot , analisado a partir da categoria created_at	144
Figura 79 – Termo wavepot , analisado a partir da categoria country	145
Figura 80 – Termo wavepot , analisado a partir da categoria city	146
Figura 81 – Termo wavepot , analisado a partir da categoria genre	147
Figura 82 – Termo wavepot , analisado a partir da categoria license	148

Lista de abreviaturas e siglas

AMC	<i>Composição Musical Automática/Algorítmica</i>
CAC	<i>Composição Assistida por Computador</i>
CGA	<i>Assistência Gerada por Computador</i>
CGC	<i>Composição Gerada por Computador</i>
CGM	<i>Composição Musical Generativa</i>
CSG	<i>Sons Gerados por Computador</i>
DJ	<i>code DJing</i>
MA	<i>Música Algorítmica</i>
MG	<i>Música Generativa</i>
MP	<i>Música Processual</i>

Sumário

Introdução	11
I ECOLOGIA DE SABERES NO <i>LIVECODING</i>	13
1 <i>LIVE CODING É MÚSICA?</i>	15
1.1 Gêneros musicais no <i>live coding</i> como reprodução de regimes de audição	19
1.1.1 Esclarecimento da notação de vínculos	20
1.2 Live coding como conjunto de conhecimentos	20
1.2.1 Duas ondas do live coding: anos 70-80 e anos 2000	22
1.3 Mapeamento de relações no Live Coding	25
1.3.1 <i>Show us your screens</i>	25
1.3.2 Vínculos não-musicais acompanhados de música no <i>live coding</i>	29
1.3.3 Vínculos musicais no <i>live coding</i>	34
1.3.4 Vínculos do executante com a partitura e público no <i>live coding</i>	42
1.4 Ecologia de práticas musicais	52
1.4.1 Apropriação da Música Algorítmica no <i>live coding</i>	54
1.4.2 Apropriação da Música Generativa no <i>live coding</i>	59
1.4.2.1 Música de algoritmos simples	59
1.4.2.2 Música de algoritmos complexos	62
1.4.2.2.1 Abordagens procedurais	62
1.4.2.2.2 Abordagens estruturais	63
1.4.3 Entre a música de vanguarda e a música de pista	67
2 PRÉ-HISTÓRIA DO <i>LIVE CODING</i>	71
2.1 Digitação de comandos e interatividade	71
2.1.1 MUSIC V	71
2.1.2 GROOVE	71
2.2 <i>Live computer Music</i>	71
2.2.1 <i>The Hub</i>	71
2.3 <i>Laptop Music</i>	71

II	IMPLEMENTAÇÕES COMPUTACIONAIS	73
3	LEVANTAMENTO DE DADOS SOBRE GÊNEROS DO LIVECODING E ÉPOCAS DE PRODUÇÃO EM UMA REDE SOCIAL	75
3.1	Metodologia de análise	77
3.2	A prática da Computer Music	79
3.3	Levantamento de dados em forma de torta	80
3.4	Levantamento de dados em forma de círculos empacotados	83
	Conclusão	85
	Referências	91
	Appendices	97
	A – TÉCNICAS DE RECURSIVIDADE	99
A.1	Fibonacci	99
A.2	Delays	99
A.3	Batimentos	99
	APÊNDICE B – WEBAUDIO API E WAVEPOT-RUNTIME	105
B.1	Wavepot	105
B.2	Coffeepot	106
B.3	Termpot	107
	APÊNDICE C – PEÇAS REALIZADAS PELO AUTOR DA TESE UTILIZANDO LIVE CODING COMO PROCESSO CRIATIVO	111
	APÊNDICE D – PLOTAGEM DE DADOS DO SOUNDCLOUD	113
D.1	Em formato de torta	113

Introdução

Tenho adotado desde meados de 2009 o computador como instrumento musical. De certa maneira, tenho praticado menos composição musical do que aquilo que Fernando Iazzetta (2009) e Guilherme Soares (2015-03-13) chamam de *luteria composicional*. Ao seguir esta nomenclatura, coloco a possibilidade de incluir outro, *improvisação de luteria composicional*, que no contexto deste trabalho, será chamado de *live coding*. No decorrer do texto apresento algumas definições deste último termo a partir de praticantes, os *live coders*. No momento peço ao leitor que considere *live coding* como uma improvisação musical através de uma performance com códigos de computador, escritos em um *laptop*.

Meu interesse pelo *live coding* começou com experimentos de programação, com a elaboração *patchs* de PureData¹ que realizam *meta-patching*² e improvisações musicais com SuperCollider³. Algumas poucas apresentações foram realizadas sob o rótulo de *live-electronics* com colegas praticantes de cinema e dança. Comecei a utilizar mais frequentemente o segundo *software*, que contém no seu manual de ajuda e utilização, o verbete *JITlib*, que por sua vez contém o termo *livecoding*.

Programação imediata (ou: programação de conversa, *livecoding* [<http://www.toplap.org>]), programação no fluxo, programação interativa) é um paradigma que inclue a atividade de programação ela mesma como uma operação do programa. Isto significa um programa que não é tomado como ferramenta que cria primeiro, e depois é produtivo, mas um processo de construção dinâmica de descrição e conversação - escrever o código e então se tornar parte da prática musical ou experimental. (SuperCollider.ORG, 2014, Verbete JITLib)⁴

¹ PureData ouPD, é um ambiente e linguagem visual de programação para Processamento de Áudio Digital (DSP) e renderização de gráficos inicialmente desenvolvido por Miller Puckette no início dos anos 90, como uma ramificação do Max/MSP (para mais informações, sugiro <https://cycling74.com/support/faq_max4/#1>). Patch é um termo derivado do paradigma de programação *dataflow* utilizado nestes dois softwares, onde se “remendam” objetos (*patching*) para realização – no contexto deste trabalho – de *Música Interativa Computacional*. Para mais informações, sugiro visitar de <<http://puredata.info/>>, acessado em 27 de junho de 2015.

² Patches que geram patches que criam instrumentos; vejo como uma técnica muito útil para síntese granular e elaboração de redes de delay no PD.

³ Um ambiente e linguagem de programação dedicado ao DSP desenvolvido inicialmente por James McCartney em 1996 para síntese em tempo-real e composição algorítmica; diferentemente do PD, é multiparadigmático, isto é, adota diversas abordagens de programação. Atualmente, sua estrutura de servidor/cliente (*SCSYNTH/SCLANG*) permite a substituição do *SCLANG* por outras linguagens de programação, segundo princípios do protocolo OSC (<<http://opensoundcontrol.org/>>). Para mais informações, <<http://supercollider.github.io/>>

⁴ Tradução de *Just in time programming (or: conversational programming, live coding , on-the fly-programming, interactive programming) is a paradigm that includes the programming activity itself in the program's operation. This means a program is not taken as a tool that is made first, then to be productive, but a dynamic construction process of description and conversation - writing code thus becoming a closer part of musical or experimental practice.*

essa divisão “prática musical x experimental” não é clara

A definição acima não inclui todos aspectos desta prática, mas ilustra o recorte feito, mais especificamente, no campo da “prática musical” em detrimento da “experimental”, embora perceba que dificilmente um se separa do outro. Por isso me refiro a procurar conhecer uma taxonomia de gêneros musicais que envolvem o *live coding*, próximo das práticas que incluem “trabalhos de música gerativa e filmes, jogos interativos, outras formas que enfatizam a narrativa e que elaboram cuidadosamente expressões que se desenvolvem no tempo” (MAGNUSSON, 2014b, p. 270)⁵.

??

Tal tarefa será realizada, no Capítulo 1, a partir da discussão de textos de praticantes falantes do inglês, que se apropriam de termos históricos para significar a emergência de produções híbridas da música algorítmica, da música processual, da música gerativa, da *live computer music*, e do eletrônico de pista (*algorave*). Sugiro como segundo capítulo um retrospecto histórico da *live computer music* dos anos 80 de compositores americanos como Ron Kuivila, Tim Perkis, Chris Brown, Scot Gresham-Lancaster e Phil Stone⁶. Como terceiro capítulo, sugiro uma análise e discussão de alguns dados presentes no Apêndice D. Esta análise não ter por fim delimitar quais são todos aspectos musicais do *live coding*, mas sim apontar, em uma *rede social*, termos que descrevem o que uma parcela dos praticantes envolvidos entendem como música improvisada em um computador. Assumido esse interesse musical, faço as seguintes perguntas, com a intenção de nortear o trabalho:

- i)* uma performance de *live coding* é música?
- ii)* como se deu a emergência de práticas musicais utilizando computadores ao ponto de ser possível improvisar?
- iii)* se for música, quais são alguns dos termos indicativos que caracterizam o *live coding* como música?

⁵ Tradução parcial nossa de *Such work includes generative music and film, interactive games, and other forms that emphasise narrative and carefully designed expression that develops in time.*

⁶ Capítulo ainda em estruturação. Anteriormente estava fazendo um levantamento histórico a partir da família de linguagens MUSIC N, através de autores como Mathews (1963), Randall (1965), Mathews et al. (1969), Mathews e Moore (1970), Roads e Mathews (1980), Park e Mathews (2009), Nunzio (2010) e Nunzio (2014). Durante o processo de escrita percebi que essa abordagem poderia dispersar o escopo principal da pesquisa. Para atingir um meio termo, iniciei estudos preliminares sobre os compositores, mas que no entanto estão bastante incompletos.

Parte I

Ecologia de saberes no *livecoding*

1 *Live coding* é música?

Muitos termos¹ diferentes são utilizados para pontuar um mesmo conjunto de práticas, objeto da pesquisa: *live coding*, *live-coding*, *livecoding*, *live code*, *live algorithm programming*, *conversational programming*, *on-the-fly programming*, *creative code*; na internet ou em autores como Collins et al. (2003a), Collins (2003), Collins et al. (2003b), Wang e Cook (2004), Ward et al. (2004), Blackwell e Collins (2005), Collins (2007), Griffiths (2008), McLean e Wiggins (2009), Rohrhuber e Campo (2009), McLean et al. (2010), Magnusson (2011), McCallum e Smith (2011), Magnusson (2014a), Magnusson (2014b), Collins e McLean (2014), Sorensen, Swift e Riddel (2014) , é mais comum encontrar os termos *live coding* e *livecoding*. Me pareceu apropriado interpretar que o primeiro é o verbete utilizado em uma fonte digital de uso geral², bem como a separação possibilita variante. Por outro lado, é possível que o leitor encontre neste trabalho a utilização de *livecoding*: este pode ser o caso de citações. Outros termos possuem recorrências nos autores citados no começo deste parágrafo. Para evitar confusões sugiro que o significado de qualquer que for o termo usado, esteja de acordo com as seguintes premissas: 1) tem o mesmo significado que *live coding*; ou 2) possui um significado próximo, mas com algum detalhe que o destaca no contexto.

genero musical existe em generos musicais?

Parafraseando Sorensen, Swift e Riddel, o “*Live coding* é uma prática de performance cujo significado existe em um número de diferentes níveis” (SORENSEN; SWIFT; RIDDEL, 2014, p. 2).³, refaço a frase como “*Live coding* é uma prática de performance no qual o gênero musical existe em um número diverso de diferentes gêneros musicais”, procurando não ver como níveis hierárquicos de significados semióticos, mas como níveis coexistentes de gêneros musicais que contêm em si conhecimentos. **otimo ponto de partida. poderia ser mais claro!**

¹ Usarei o itálico como identificação de termos em inglês ou para termos capturados das traduções que são de interesse para a pesquisa. Termos em negrito serão utilizados para perguntas.

² <https://en.wikipedia.org/wiki/Live_coding>.

³ Tradução nossa de *Livecoding is a performance practice which meaning exists on a number of different levels*. No entanto o que os autores utilizaram por “significados [existem] em um número de diferentes níveis” são mundos de idéias hierarquizados, isto é, se concebe um programa de pesquisa que aborde problemas da sintaxe, da semântica e da pragmática (semótica) no *live coding*, ou poderia ser dito que o *live coding* está em interdisciplinariedade com a *semiótica* : “Que estes mundos conceituais existem pressupõe que semânticas podem operar em diferentes níveis. Uma questão importante é então como o significado cruza os limites semânticos. Morris (1938) propôs que a semiótica fosse quebrada em três campos; pragmática, a relação entre signos e intérpretes; semântica, a relação de signos para objetos; e sintaxe, a relação dos signos para um outro.” Tradução nossa de *An important question then is how meaning crosses semantic borders*. Morris (1938) proposed that semiotics be broken into three fields; pragmatics, the relationship between signs and interpreters; semantics, the relation of signs to objects; syntax, the relations of signs to one another.(SORENSEN; SWIFT; RIDDEL, op. cit., p. 5). No entanto a pesquisa realizada aqui busca abordar essa frase pelo ponto de vista da cultura, a partir de um levantamento de informações textuais e dados computacionais referentes a uma cultura de gêneros musicais. **nao entendo essa discussão aqui**

Se considerarmos o *Live coding* como um universo de conhecimentos, o primeiro conhecimento parte de uma interpretação de uma tradução literal: *live coding*, onde *live*, como “ao vivo”, do momento próprio da experiência da improvisação, que possue uma vivacidade (*liveness*); e o sufixo *coding* como “codificar”, isto é, programar um dispositivo. Em outras palavras, programar ao vivo em um contexto performático. [poderia começar por aqui](#)

Outros conhecimentos surgem de definições, como: 1) as que especificam o dispositivo programado do parágrafo anterior: "improvisação de vídeo e/ou música usando **linguagens de computador** que tem se desenvolvido em um campo ativo de pesquisa e prática artística ao longo da última década" (McLean et al., 2010)⁴; 2) descrevem o *live coding* como um programa de pesquisa que gira torno de um grupo:

Nos anos recentes houve uma expansão da atividade do *live coding*, e a formação de um corpo internacional para suportar *live coders* - TOPLAP. O sítio <toplap.org> e sua lista de email é a mais ativa casa para práticas artísticas, e o TOPLAP tem sido listado para tais festivais de música eletrônica como Ultrasound (2004), transmediale 2005 (Berlin) e Sonar 2005 (Barcelona) (BLACKWELL; COLLINS, 2005, p. 3-4).⁵

3) As que descrevem práticas a partir de personagens da música eletrônica de centros de entretenimento: no manifesto “*Live Algorithm Programming and Temporary Organization for its Promotion*” é apresentada algumas base conceituais de um tipo de *live coding* praticado em países falantes da língua inglesa:

O *Livecoding* permite a exploração de espaços algorítmicos abstratos como uma improvisação intelectual. Como uma atividade intelectual, pode ser colaborativa. Codificação e teorização podem ser atos sociais. Se existe um público, revelar, provocar e desafiar eles com uma matemática complexa se faz com a esperança de que sigam, ou até mesmo participem da expedição. Estas questões são, de certa forma, independentes do computador, quando a valorização e exploração de algoritmo é que importa. Outro experimento mental pode ser encarado com um DJ ao vivo codificando e escrevendo uma lista de instruções para o seu *set* (realizada com o iTunes, mas aparelhos reais funcionam igualmente bem). Eles passam ao HDJ [*Headphone Disk Jockey*] de acordo com este conjunto de instruções, mas no meio do caminho modificam a lista. A lista está em um retroprojetor para que o público possa acompanhar a tomada de decisão e tentar obter um melhor acesso ao processo de pensamento do compositor. (WARD et al., 2004, p. 245)⁶

⁴ Tradução parcial nossa de: *Live coding , the improvisation of video and/or music using computer language, has developed into an active field of research and arts practice over the last decade.* Grifo nosso.

⁵ Tradução parcial de: *Recent years have seen further expansion of live coding activity, and the formation of an international body to support live coders- TOPLAP (Ward et al. 2004). The toplap.org site and mailing list is the most active current home for this artistic practice, and TOPLAP have been booked for such electronic music festivals as Ultrasound 2004 (Huddersfield), transmediale 2005 (Berlin) and sonar 2005 (Barcelona).*

⁶ 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*

A partir das três definições acima, de uma prática multimidiática, de um grupo específico de pesquisadores, e improvisação musical para atividades de entretenimento , transformarei a primeira pergunta feita no **Item i** da Introdução, no último parágrafo da página 14: **considerando o *live coding* como uma prática interdisciplinar realizada através da improvisação com multimídias, quais são as relações musicais desta prática realizada por um grupo específico?** Essa pergunta abre espaço para discutir a diversidade de relações entre *i) agentes* (*live coders/intérpretes* e público); *ii) meios de expressão* (vídeo, música); e *iii) dispositivos* (computadores, projetores, alto-falantes, placas-de-som, placas-de-vídeo, cabos).

A mesma pergunta pode ser feita de outra maneira, de forma mais sintética: **em ambientes performáticos de (*live coding*), o que pode ser pressuposto musicalmente entre os agentes?**

Para começar a responder estas duas perguntas, necessito antes mapear as relações do parágrafo anterior como forma de recortar o objeto de pesquisa. Isso foi realizado através de um diagrama (ver [Figura 1](#)), tomando as seguintes suposições gerais: *a)* Uma improvisação entre uma parte ativa dos agentes (*live coders*): músicos, artistas plásticos, designers, cineastas, engenheiros de computação, utilizando um computador cada; *b)* um grupo de agentes passivos (*público*), no sentido de que ou está assistindo, ou dançando, mas dificilmente se relaciona diretamente com o computador; *c)* cada *performer* cria e/ou edita arquivos de computador (*scripts*) em *softwares* apropriados para obtenção imediata de resultados sonoros e visuais⁷; *d)* a projeção sonora é feita através dos alto-falantes; *e)* a projeção de imagens é feita através de um projetor direcionado para uma superfície, plana ou irregular; *f)* a superfície de projeção pode conter a exposição do código-fonte⁸ (ver [Figura 2](#)); *g)* os resultados sonoros e visuais serão julgados e posteriormente re-codificados, retornando para o [Item c](#).

Neste diagrama, sete nós (elementos) e dez relações (linhas) podem ser agrupados

can hopefully make them follow along or even take part in the expedition. These issues are in some ways independent of the computer, when it is the appreciation and exploration of algorithm that matters. Another thought experiment can be envisaged in which a live coding DJ writes down an instruction list for their set (performed with iTunes, but real decks would do equally well). They proceed to HDJ according to this instruction set, but halfway through they modify the list. The list is on an overhead projector so the audience can follow the decision making and try to get better access to the composer's thought process.

⁷ Essa imediaticidade é resultado de um processo conhecido como *Just In Time compilation* (JIT). Segundo [Aycock \(2003\)](#), as diferentes técnicas JIT servem para um propósito determinado de uma linguagem de programação particular, no nosso caso musical, e a diferença de tempo entre o que é escrito, compilado e interpretado pelo computador "não deve causar pausas desagradáveis em uma execução normal do programa" ([AYCOCK, 2003](#), p. 2). Tradução nossa de *A JIT system must not cause untoward pauses in normal program execution as result of its operation*

⁸ Código-fonte ou *source-code*, segundo a *The Linux Information Project* (LINFO), "é uma versão de um *software* originalmente escrita (i.e., digitada em um computador) por um ser humano em um texto simples(i.e., caracteres alfanuméricos legíveis por humanos" (ver <http://www.linfo.org/source_code.html>). Tradução nossa de (...) is the version of software as it is originally written (i.e., typed into a computer) by a human in plain text (i.e., human readable alphanumeric characters).

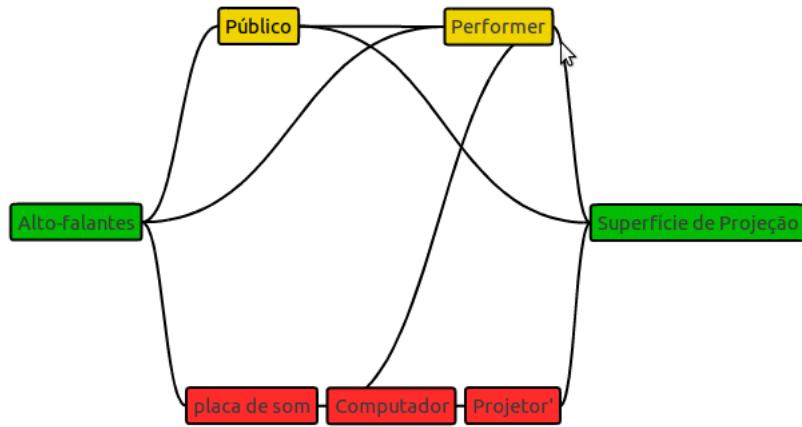


Figura 1 – Diagrama de relações entre pessoas, dispositivos de controle e projeção (acústica e visual)



Figura 2 – Exemplo de performance de *livecoding* realizado pelo labMacambira em São Carlos - SP, onde podemos ver o código sobreposto ao vídeo. O software utilizado é o Vivace (VIEIRA et al., 2015). **Fonte:** <<http://i.ytimg.com/vi/i9Lj3S1bSH4/hqdefault.jpg>>

em quatro tipos de vínculos: *I*) entre hardwares (vermelhos): conversão das amostras digitalizadas em sinais analógicos; *II*) entre *hardwares* e dispositivos de projeção (vermelhos e verdes): transmissão dos dados convertidos; *III*) entre dispositivos de projeção e agentes (verdes e amarelos): projeção dos dados transmitidos para o público; e *IV*) entre público e *performer* (amarelos): cultural, reconhecimento de uma manifestação passível de classificação (*erudito, pop, experimental*, etc.), ao qual posso indicar aqui como rótulos de gêneros musicais.

1.1 Gêneros musicais no *live coding* como reprodução de regimes de audição

Por gênero musical quero dizer algo próximo do que Simone Pereira de Sá (2006) chama de “regime de audição”, em seu artigo “A música na era de suas tecnologias de reprodução”. Uma forma de entender “regime de audição” é como

(...) o papel das comunidades de gosto tais como as cenas e as subculturas e das regras de gêneros musical mediando a relação da tecnologia com o consumo a partir do estabelecimento de valores e hierarquias, uma vez que a avaliação e a identificação, **presentes na pergunta ‘Com que se parece este som?’ ou do ponto de vista da indústria, ‘quem vai consumir esta música?’**, permanecem como nortes do consumo musical“ (Sá, 2006, apud Janotti Jr. 2003, p. 15).

A autora desenvolve um tema sobre a articulação cultural entre “comunidades de gosto” (gênero musical) e tecnologias de produção musical: 1) trio microfone, amplificação elétrica e alto-falantes; 2) luteria (eletrificação da guitarra); e 3) artefatos de audição (toca-discos, CD’s, etc.). No entanto, no *live coding* não se reproduz música da mesma maneira que um CD ou toca-discos; se codifica a música no momento da performance. Assim, o sentido dado no termo “reprodução musical”, neste trabalho, possui outro aspecto, que é a repetição de gostos musicais que ocorrem com as tecnologias de Sá, no *live coding*; isto é, no *live coding* não seria de fato um novo gênero, mas uma prática que dissemina os gêneros musicais que já existem.

O interesse pelo discurso da autora também reside na *desintermediação musical*, isto é, a quebra da mediação artista/público anteriormente feita por gravadoras, independente de gênero musical, agora realizada pelos próprio músico. Segundo Sá, “torna-se cada dia mais comum a história de bandas ou artistas que divulgam seu trabalho sem a intermediação das gravadoras”⁹, que têm seus próprios *home studio* e *sites* de divulgação.

No caso do *live coding*, o paralelo é: o *home studio* está empacotado no *laptop*, e músicos-programadores levam para o palco o próprio processo de produção musical

⁹ Sá, op. cit., p. 15.

(improvisação através do código); apesar dessa desintermediação, ainda assim gêneros musicais (ou regimes de audição) produzidos com as tecnologias de Sá (como por exemplo o *dance*, *techno*, *pop*, etc.) são reproduzidos no *live coding*.

Tais regimes de audição no *live coding* podem ser muitos, dependentes de um *universo de conceitos* (McLean, 2006); para poder conectar conceitos, diagramas, imagens e leituras, adotei o método de pesquisa proposto por Santos (2008), buscando observar na [seção 1.2](#), o *universo de conceitos do live coding* como um *conjunto limitado de conhecimentos*.

Durante a contextualização desse conceito, ocorrerá o processo de desconstrução do diagrama de conjunto de conhecimentos, na [seção 1.3](#). Na seção [seção 1.4](#) buscarei fazer um levantamento de conhecimentos composicionais (estéticas e/ou técnicas) que estão incluídos nos discursos realizados pelos autores que praticam o *live coding*.

1.1.1 Esclarecimento da notação de vínculos

As sessões supracitadas podem começar com uma proposição do tipo: $A \Leftrightarrow B \Leftrightarrow C$. A, B e C são um conjunto de conhecimentos, no caso deste trabalho, dependente do termo *live coding*. O símbolo \Leftrightarrow foi adotado para significar que uma área de conhecimento A está em relação com outra área de conhecimento B; esta forma também indica que mais termos podem ser relacionados: $A \Leftrightarrow B \Leftrightarrow C$ descreve uma relação direta de A com B e indireta entre A e C.

1.2 Live coding como conjunto de conhecimentos

Antes de iniciar o mapeamento específico das relações musicais, creio ser importante esclarecer o que entendo por multiplicidade de manifestações através de gêneros musicais, que por sua vez, possuem uma multiplicidade de conhecimentos: 1) a definição apresentada por McLean et al. (2010), no segundo parágrafo da página 18, de uma natureza multimidiática no *live coding*, apresenta a figura do *performer* (*live coder*) como organizando conjuntos de conhecimentos diversos, que atravessam práticas musicais, audiovisuais, literárias e técnicas (programação); 2) cada conjunto de conhecimento, pressupõe universos próprios de conhecimentos específicos que dificilmente podem ser tomados em separado (por exemplo, quais são as linguagens, poéticas e técnicas musicais ouvidas e praticadas? Quais linguagens, poéticas e técnicas audiovisuais utilizadas? Quais as técnicas e linguagens de programação empregadas para se atingir os resultados artísticos? Qual sua história?); 3) considerando que o *live coding*, na sua configuração atual, é uma pluralidade de práticas emergentes nos últimos quinze (15) anos (aproximadamente), buscarei fazer um recorte específico do campo musical: apresento na [subseção 1.2.1](#), através de Blackwell e Collins (2005), McLean e Wiggins (2009) e Brown e Bischof (2013), um panorama de duas frentes

históricas; a primeira, *live computer music*, e a segunda como característica das descrições de diferentes autores, entre eles, Ward et al. (2004) e McLean et al. (2010). .

Esse ponto de vista, do *live coding* como conjunto de conhecimentos que incluem outros conhecimentos, recorrerei ao que Boaventura de Souza Santos (2008) discorre no seu artigo “A filosofia à venda, a douta ignorância e a aposta de Pascal” como uma *ecologia de saberes*; esta ecologia por sua vez, é uma estratégia metodológica alternativa ao que o sociólogo chama de pensamento especialista, ou *pensamento ortopédico*, que ignora um conjunto de saberes externos ao que se discute, declarando a inexistência da diversidade, com a problemática de realizar:

“(...) o constrangimento e o empobrecimento causado pela redução dos problemas a marcos analíticos e conceptuais que lhes são estranhos. Com a crescente institucionalização e profissionalização da ciência – concomitante da passagem, assinalada por Foucault, do “intelectual universal” ao “intelectual específico” – a ciência passou a responder exclusivamente aos problemas postos por ela.”(SANTOS, 2008, p. 15).

A ecologia de saberes é um método que busca ser consciente de pluralidades e desproporções daquilo que se quer saber; é necessário, no entanto, ter o cuidado para que o próprio processo de ser consciente destas pluralidades e desproporções torne o objeto de pesquisa algo incomensurável, forçando o pesquisador a criar uma abordagem limitada do que se quer falar:

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

A palavra *ecologia* será traduzida como *estudo da casa* (do grego *oikos*, casa, *logia* estudo); para diferenciar o termo usado de sua aplicação original nas ciências biológicas, bem como parafrasear Boaventura, irei traduzir o termo *ecologia de saberes* como *estudo da casa dos saberes*, como metáfora para um conceito mais específico, próprio deste trabalho: **estudo do conjunto limitado e desproporcional de categorias de gêneros musicais do live coding**; a metáfora da casa é oportuna, pois é possível

imaginar a questão: quem construiu a casa? ou melhor, que grupo(s) de pessoas têm levantado um conjunto de conhecimentos pertinentes ao objeto de pesquisa?

ótimo!

Para desenvolver esse ponto de vista, indico um grupo reconhecido como TOPLAP¹⁰, uma comunidade de músicos-programadores interessados no tema *live coding*. Como vimos no início do capítulo, através de três definições diferentes, preocupações deste grupo têm reflexos na música, audiovisual, mas adicionaria aqui a literatura¹¹ e programação¹².

Seguindo o caminho proposto por Boaventura¹³, esse conjunto de conhecimentos multimidiáticos do *live coding* será limitado, isto é, manterei meu foco em conhecer uma dinâmica de gêneros musicais: busco entender neste capítulo como os praticantes, *live coders* do grupo TOPLAP, consideram o fazer musical em suas performances. Ademais, observei em textos algumas apropriações de outros conhecimentos: *Música Algorítmica* (que será abreviado como MA), *Música Processual* (MP), *Música Generativa* (MG), *Música de Pista* ou práticas de DJ (DJ).

1.2.1 Duas ondas do live coding: anos 70-80 e anos 2000

McLean e Wiggins (2009) expõem um conjunto de conhecimentos históricos que descrevem a emergência do *live coding*. A prática tem seu primeiro exemplo na metade dos anos 80 (como uma pré-história do *live coding*), mas foi apenas com o advento de computadores portáteis, somado ao advento de tecnologias de linguagens de programação interpretadas em tempo-real e a emergência do grupo TOPLAP, que foi possível delinear uma identidade cultural própria do *live coding*.

A primeira performance conhecida de *live coding* foi em 1985, por Run Kuivila na STEIM em Amsterdã (BLACKWELL; COLLINS, 2005). O *live coding* não possui sua própria identidade cultural até o TOPLAP, a Organização Temporária para a Promoção da Programação Ao Vivo de Algoritmos, formada na *Changing Grammars Workshop* em 2004 (WARD et al., 2004). Mesmo sendo possível pensar em fazer *live coding* sem computadores, através da auto-modificação de composições baseadas em regras, não existe evidência que isto foi feito antes da invenção dos computadores. Parece que foi necessária a invenção da interpretação

¹⁰ Organização Temporária para a Pragmática da Programação Artística Ao Vivo (*Temporary Organisation for the Pragmatics of Live Art Programming*): <http://toplap.org/wiki/Main_Page>. Este termo também pode ser entendido como uma permutação silábica do acrônimo LAPTOP formado no título do artigo “Live Algorithm Programming and a Temporary Organisation for its Promotion.” (WARD et al., 2004).

¹¹ Indico as adaptações poéticas para linguagens de programação e a literatura generativa de Cox, McLean e Ward (2004).

¹² Como o compartilhamento de experiência do processo de programação de um *software*, como Ajden Towfeek demonstra isso em <<https://www.youtube.com/watch?v=WrxKLoj4QFc>>.

¹³ Em realidade esse caminho já era imaginado através da leitura de autores como Thomas Kuhn (1970) e Paul Feyerabend (1975); porém o trabalho de Boaventura sintetiza uma série de questionamentos que surgiram durante a pesquisa.

dinâmica de códigos para o *live coding* aparecer como possível ou talvez desejável (McLean; WIGGINS, 2009, p. 1-2)¹⁴

Existem pontos que necessitam ser colocados: o primeiro é o que Blackwell e Collins (2005) comentam como “duas ondas” de *live coding*, sendo que a primeira poderia ser melhor denominada como *live computer music*; além de Ron Kuivila (1955-)¹⁵, é mencionada a banda *The Hub* em 1986 (BLACKWELL; COLLINS, 2005, p. 3-4), formada por John Bischoff (1949-)¹⁷, Tim Perkis, Chris Brown(1953-), Scot Gresham-Lancaster (1954-)¹⁸, Mark Trayle (1955-)¹⁹ e Phil Stone, com um primeiro disco lançado em 1989 intitulado *The Hub: Computer Network Music*. É interessante notar que Bischoff e Perkis já faziam experimentos similares no final dos anos 70, com a “*The League of Automatic Music Composers*”: segundo Brown e Bischof (2013), suas performances na cena musical da Baía de São Francisco eram apresentações informais, na qual os compositores passavam horas ajustando uma rede de microcontroladores²⁰, onde público podia observar a apresentação de forma mais ativa, chegando perto ou até mesmo com outros compositores trazendo seus equipamentos para tocarem juntos:

Na primavera de 1979, montamos uma série quinzenal regular de apresentações informais sob os auspícios da *Bay Center for the Performing Arts*. Todos outros domingos à tarde passávamos algumas horas configurando nossa rede de KIMs na sala *Finnish Hall*, na Berkeley, e deixávamos a rede tocando, com retoques aqui e ali, por uma ou duas horas. Os membros da audiência poderiam ir e vir como quisessem, fazer perguntas, ou simplesmente sentar e ouvir. Este foi um evento comunitário de tipos como outros compositores aparecendo, tocando ou compartilhando circuitos eletrônicos que tinham projetado e construído. Um interesse na construção de instrumentos eletrônicos de todos os tipos parecia estar "no ar". Os eventos da sala *Finn Hall* foram feitos para uma cena com paisagens sonoras geradas por computador misturado com os sons de grupos de dança folclórica ensaiando no andar de cima e as reuniões ocasionais do Partido Comunista na sala de trás do edifício velho venerável. A série durou cerca de 5 meses que eu me lembre.(BROWN; BISCHOF, 2013, online)²¹

¹⁴ Tradução nossa de: *The earliest known live coding performance was in 1985, by Ron Kuivila at STEIM in Amsterdam (Blackwell and Collins, 2005). Live coding did not get its own cultural identity until TOPLAP, the Temporary Organisation for the Promotion of Live Algorithm Programming was formed at the Changing Grammars workshop in Hamburg in 2004 (Ward et al., 2004). Although it is possible to do live coding without computers, through self-modifying rule-based composition, there is no evidence that this was done before the invention of computers. It would seem that it required the invention of dynamic code interpretation for live coding to appear possible or perhaps desirable.*

¹⁵ <http://www.foundationforcontemporaryarts.org/grant_recipients/ronkuivila.html>, Sound Designer e compositor (aluno de Alvin Lucier na Universidade de Wesleyan¹⁶) natural de Boston.

¹⁷ Foi aluno de compositores como James Tenney and Robert Ashley.

¹⁸ Foi aluno de compositores como Darius Milhaud, John Chowning, Ashley e Terry Riley.

¹⁹ Também estudou com Ashley.

²⁰ KIM-1; <<http://www.6502.org/trainers/buildkim/kim.htm>>.

²¹ 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*

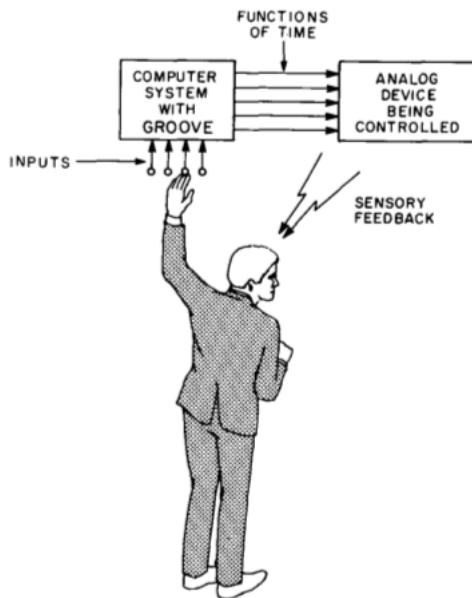


FIG. 1. Feedback loop for composing functions of time

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

Se seguirmos essa lógica de retrocesso temporal, conectando uma prática comum do paradigma atual do *live coding*, a imediaticidade, é possível voltar um pouco mais no tempo e concluir que o GROOVE, desenvolvido por Max [Mathews e Moore \(1970\)](#) um ano após a divulgação do MUSIC V, também foi um precursor do *live coding*; isto é, se for desconsiderado o plano performático e considerando apenas sua funcionalidade. O GROOVE foi um sistema desenvolvido na Bell Labs a partir da reflexão na questão da interatividade no processo criativo da síntese sonora: o compositor poderia escutar sons sintetizados enquanto fornecia uma sequência de comandos diretos para o sistema.

O segundo ponto abordado por [McLean e Wiggins \(2009\)](#) no início desta subseção (1.1.1), existe uma cena de *live coding* no início dos anos 2000 com as experimentações do compositor Julian Rohruber utilizando o SuperCollider²², bem como a cena musical noturna em torno do *Slub*, formado por Adrian Ward, Alex McLean e Dave Griffiths ([COLLINS et al., 2003a](#), p. 3). O *SuperCollider*²³, inicialmente desenvolvido por James McCartney (este também um ativo praticante do *live coding*), possuía apenas funcionalidades de composição algorítmica e síntese sonora em tempo-real, isto é, a demanda temporal entre o que é

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.

²² <<http://supercollider.github.io/>>, acessado em 27 de junho de 2015.

²³ <<http://github.io.supercollider>>, acessado em 27 de junho de 2015.

programado e seus resultados é consideravelmente reduzida; mas ainda assim, após uma edição, era necessário uma reinterpretação do código²⁴. O trabalho de Rohruber, *JITlib* a biblioteca *Dewdrop lib*²⁵ contribui para a conceitualização antes mesmo do *Slub*. É a partir destes adventos, uma sequência de apresentações musicais e publicações de artigos que a configuração cultural do que chamarei aqui de *live coding anglófono*²⁶ começou a tomar corpo.

1.3 Mapeamento de relações no Live Coding

De volta à proposição do mapeamento (ver Figura 1), discuto agora os vínculos não-musicais e os musicais. Em geral, relações não musicais são audiovisuais, mas também descrevo um uso literário e outro com as artes do corpo. Vínculos musicais utilizam alguns elementos visuais, mas o seu foco está no som. Na tentativa de seguir a metodologia da ecologia de saberes proposta por Santos (2008), vínculos visuais contextualizam a existência de gêneros musicais próprios destas manifestações, admitindo uma impossibilidade de tratá-las em todos seus aspectos técnicos ou artísticos. Apresentarei primeiro um manifesto do grupo TOPLAP na subseção 1.3.1, uma discussão dos vínculos não-musicais na subseção 1.3.2, e uma discussão sobre vínculos musicais na subseção 1.3.3.

1.3.1 *Show us your screens*

Performances de *live coding* podem seguir algumas premissas de comportamento visual ou procedimentos técnicos, mas não musicais. Assim é importante definir o que chamarei de *imagem-texto*²⁷, isto é: 1) a sobreposição do texto do código (estruturas lógicas e comentários) à imagem renderizada, em uma mesma tela; ou 2) a exposição de diferentes códigos em janelas separadas da imagem renderizada. Essas premissas são comportamentos comuns entre alguns programadores que fazem música e músicos que programam, no contexto do *live coding*; mas foram bem delimitadas em um manifesto disponível no “Toplap Manifest” e conhecidos pelo *slogan*, “*Show Us Your Screens*”(GRIFFITHS, 2008; McCallum; SMITH, 2011, p. 22; online):

Exigimos:

- Acesso à mente do intérprete, para todo o instrumento humano.
- Obscurantismo é perigoso. Mostre-nos suas telas.

²⁴ SuperCollider é uma linguagem interpretada, isto é, têm por base no código fonte linguagens compiladas. Diferentemente de linguagens como C e C++, que necessitam, dentre vários processos a compilação

²⁵ *Dewdrop* é o nome artístico de James McCartney. Para mais informações a respeito, <<http://www.dewdrop-world.net/bio/index.html>>

²⁶ Inglaterra, EUA, parte inglesa do Canadá e Austrália.

²⁷ Até o momento não encontrei alguma referência direta deste termo; assim que possível, buscarei em algum autor uma melhor fundamentação deste termo.

**algoritmos e motoserras
sao ferramentas e
pensamento. Ver
Flusser. Isso
deve ser questionado.**

- 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. Motoserras são ferramentas. É por isso que às vezes algoritmos são mais difíceis de perceber do que motosserras.
- Reconhecemos contínuos de interação e profundidade, mas preferimos:
 - Introspecção dos algoritmos.
 - A externalização hábil de algoritmo como exibição expressiva/impresiva de destreza mental.
 - Sem *backup* (minidisc, DVD, safety net computer).
- Nós reconhecemos que:
 - Não é necessário para uma audiência leiga compreender o código para apreciar, tal como não é necessário saber como tocar guitarra para apreciar uma performance de guitarra.
 - Codificação ao vivo pode ser acompanhada por uma impressionante exibição de destreza manual e a glorificação da interface de digitação.
 - Performance envolve contínuos de interação, cobrindo talvez o âmbito dos controles, no que diz respeito ao parâmetro espaço da obra de arte, ou conteúdo gestual, particularmente direcionado para o detalhe expressivo. Enquanto desvios na tradicional taxa de reflexos táticos 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.²⁸

“Dar acesso à mente do intérprete” e “obscurantismo é perigoso” descrevem um meio de evitar qualquer código mal intencionado; isto é uma hipótese: *i)* programas de *live coding* geralmente são programas em fase de desenvolvimento; *ii)* programas em desenvolvimento possuem, inevitavelmente, *bugs*²⁹ *iii)* *bugs* podem ser explorados e levar à

²⁸ 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.

²⁹ Segundo James S. Huggins, historicamente “O termo *bug* é usado de forma limitada para designar qualquer falha ou problema em conexões ou no trabalho com aparelhos elétricos” Tradução nossa de *The term "bug" is used to a limited extent to designate any fault or trouble in the connections or working of electric apparatus.* (ver <http://www.jamesshuggins.com/h/tek1/first_computer_bug.htm>). Nesse sentido, um *bug* em um programa é uma falha de operação, geralmente causada por algum erro de lógica, por parte do programador.

corrupção do sistema. Se esta hipótese estiver correta, justificaria a atitude de exposição da *imagem-texto*. No entanto não encontrei algum estudo crítico descrevendo se isso é verdade a partir do ponto de vista do público, isto é, será que o público pode estar realmente interessado na exposição da *imagem-texto*? Será que essa exposição não pode ser perigosa para o processo artístico e para a experiência do público? Embora sejam questões que fogem do escopo do trabalho, são importantes, necessitando verificar algumas performances para averiguar. **falsas questões? que público?**

“Programas são instrumentos” e “O programa será transcendido - Língua Artificial é o caminho”, são frases que fazem menção direta à experiência de usuário (*live coder*), isto é, um sistema que é programável de maneira facilitada. A seguinte hipótese pode ser feita: quanto mais simplificada a linguagem de programação, mais expressão visual ou musical um espetáculo poderá ter (o que pode não ser verdade, e sim que a expressão musical estaria no nível sensível). Por “Língua Artificial” entendo que o *live coder* pode criar *mini-linguagens* ou Linguagens de Domínio Específico (DSL)³⁰ que possibilitem criar programas para criar um espetáculo audiovisual ou musical. Segundo Collins e McLean (2014), tais DSLs estariam no formato de “mini-linguagens” bem desenvolvidas para a tarefa específica de codificar música ao vivo, operando técnicas compostionais como a transformação de um padrão musical (como por exemplo, técnicas barrocas como inversão e retrogradação ou técnicas aleatórias, como embaralhamento de um conjunto de eventos sonoros), facilitando a espontaneidade no processo criativo:

Existe um número crescente de sistemas de *livecoding* com “mini-linguagens” amigáveis, que facilitam o *loop* e contruções de camadas centrais típicas para dançar música. *Ixilang* é um exemplo primário, e possui um editor de código estruturado que, enquanto baseado em texto, suporta correspondências visuais. *Tidals* é outro, e, embora com foco na rapidez de utilização ao invés da facilidade de aprendizagem, está começando a ter mais ampla aceitação. Ambos *ixilang* e *Tidal* promovem padrões em termos de funções transformadoras como embaralhamento, inversão e extração de formas diferentes. (COLLINS; McLean, 2014, p .357)³¹

³⁰ Sobre esse tema recomendo o texto “Minilanguages, finding a notation that sings” de Raymond (2003): “Historicamente, linguagens de domínio específico são do tipo que são chamadas de ‘pequenas linguagens’ ou ‘minilinguagens’ no mundo do Unix, porque os primeiros exemplos eram pequenos e de pouca complexidade, em relação às linguagens de propósito geral (...) Nós manteremos o termo tradicional ‘minilinguagem’ para engatizar que no decorrer do curso é geralmente utilizado para projetar e manter-las o menor e simples possível” (RAYMOND, 2003, 3º parágrafo). Tradução nossa de *Historically, domain-specific languages of this kind have been called ‘little languages’ or ‘minilanguages’ in the Unix world, because early examples were small and low in complexity relative to general-purpose languages (...) We’ll keep the traditional term ‘minilanguage’ to emphasize that the wise course is usually to keep these designs as small and simple as possible.*

³¹ Tradução nossa de: *There are increasingly user friendly “mini-language” livecoding systems which facilitate loop and layer-centric constructions typical to dance music. ixilang is a primary example, and features a structured code editor which while text-based, supports visual correspondences. Tidal is another, and although its focus is on speed of use rather than ease of learning, is beginning to see wider take-up. Both ixilang and Tidal promote pattern in terms of transformative functions as scrambling, reversal and extrapolation in different ways.*

“O código deve ser visto assim como ouvido” entraria em um problema próprio de programas de pesquisa em notação musical, sendo que o processo de correlação entre o que está escrito e o que está sendo ouvido leva um tempo ou pode mesmo nem existir. Mesmo com o convite expressado pelo manifesto “*Live Algorithm Programming and Temporary Organization for its Promotion*” no início do capítulo, a questão não está nem no uso do computador nem em alguma abordagem musical, e conforme a performance avança, a imagem-texto vai se tornando tão poluída que poderia causar um desinteresse.

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

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

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

A respeito de “Não é necessário para uma audiência leiga compreender” e “A codificação ao vivo pode ser acompanhada por uma impressionante exibição de destreza” pode indicar uma proximidade com aquele modelo de prática musical virtuosística (como um espetáculo de habilidades técnicas); mas especificamente, este modelo poderia partir daquilo que Magnusson (2014a) chama de “adoção de um método pré-romântico de compor através da performance em tempo real, onde tudo fica aberto a mudar – o processo composicional o design do instrumento e a inteligência do sistema tocando a peça.” (MAGNUSSON, 2014a, p. 4)³² [Ver artigo de Suzan McClary: Terminal Prestige, the case of avant-garde music composition \(aqui esta simplista\)](#)

³² Tradução nossa de: *live coding adopts a pre-Romantic method of composing through performance in real time, where everything remains open to change – the compositional process, the instrument design, and the intelligence of the system performing the piece.*

1.3.2 Vínculos não-musicais acompanhados de música no *live coding*

Computador ⇔ projetor, projetor ⇔ superfície ⇔ performer e projetor ⇔ superfície ⇔ público, são relações que caracterizam performances utilizando o vídeo (finalizado ou por *webcams*) e a síntese de imagens 2D e 3D. Em suma, imagens renderizadas para um formato digital. Adicionalmente pode ocorrer a exposição de um ou mais *códigos-fontes* responsáveis pela renderização, na superfície de projeção junto com a imagem renderizada. Pode ser que em alguns vídeos demonstrativos no *Youtube* (que apresentarei logo a seguir), essa superfície não exista, isto é, são expostas como mídia em um *videoplayer*.

Também é possível notar um acompanhamento musical do vídeo, realizado de duas maneiras: *i*) códigos são separados, para imagem e para a música; *ii*) códigos são compartilhados pela imagem e pela música. Em ambos os casos o ato de codificar faz parte de uma improvisação livre, ou improvisação guiada, ou da livre associação de idéias que são posteriormente estruturadas³³. Pratiquei esse último modo em uma apresentação realizada no MIS/SP juntamente com Rodrigo Belotto e Felipe Stutzman no “A beleza por trás da inconsequência do motorista bêbado” em 2010³⁴. No caso do segundo item deste parágrafo, existe a tendência de uma *visualização da música*, isto é, de fazer uma tradução de movimentos visuais, cores, efeitos e outros artifícios que criem alguma conexão com os movimentos sonoros repetitivos ouvidos, de um regime de audição próximo da música eletrônica de pista. É comum que o acompanhamento musical realize aquilo que Collins e McLean (2014) chamam de *code DJing* ou *live electronic dance music* (COLLINS; McLean, 2014, p. 355-357).

Para realizar tais performances audiovisuais, *live coders* utilizam diferentes *softwares*; para que seja possível passar uma idéia mais clara sobre o que foi falado nos dois parágrafos anteriores, sugiro descrever alguns deles como forma de oferecer um panorama desta parte da prática do *live coding*: Perl / Perl-live, Scheme / Fluxus, Lisp / Extempore / Impromptu, Javascript / Gibber³⁵.

*Perl*³⁶ é uma linguagem de programação de alto nível e, como o nome sugere, permite que tarefas complicadas sejam realizadas de maneira facilitada. Cox, McLean e

³³ Com isso quero dizer que, como descrito por Collins et al. (2003b): “Meu ambiente de composição é usualmente uma combinação de tais linhas [improvisadas] bem como instrumentos, rotinas e padrões [códigos] prédefinidos que similarmente combinam para um referencial” (COLLINS et al., 2003b, p. 329). Tradução nossa de: *My own composition environment is usually a combination of such lines as well as predefined instruments, routines and patterns that similarly combine to a referential system.*

³⁴ <<https://www.youtube.com/watch?v=uClk-WHS9Ek4>>; no entanto não praticamos a projeção do código. Realizei um *mashup* de música de pista generativa, amostras de *blues*, Jovem Guarda, gravações de ambientes noturnos, gravações de vozes em conversas de bares; vídeos projetavam colagens do *Youtube* de acidentes de trânsito, noticiários e gravações de bares *in loco*.

³⁵ Este *software* foi durante um tempo, foco da pesquisa, no sentido de que existia a exploração de um subgênero de *live electronic dance music* conhecido como *algorave*. No decorrer da pesquisa, uma versão atualizada foi lançada com capacidades de síntese de imagens em OpenGL e Processing, muito utilizados por artistas visuais.

³⁶ *Practical Extraction and Report Language*, mais informações em <<http://linux.die.net/man/1/perl>>.

```

if ((light eq dark) && (dark eq light)
    && ($blaze_of_night{moon} == black_hole)
    && ($ravens_wing{bright} == $tin{bright})) {
    my $love = $you = $sin{darkness} + 1;
};

```

If light were dark and dark were light
The moon a black hole in the blaze of night
A raven's wing as bright as tin
Then you, my love, would be darker than sin.

Figura 4 – Poema traduzido para linguagem Perl. (COX; McLean; WARD, 2000, p. 4)

Ward (2000) e Cox, McLean e Ward (2004) descrevem algumas abordagens de utilização do código em *Perl*, caracterizando o termo *creative code*, utilizado no início deste capítulo: *i*) práticas poéticas anteriores ao TOPLAP, como a tradução de poemas em códigos (ver Figura 4); *ii*) programas auto-poéticos, isto é, o programa escrito em um editor customizado, que modifica ele mesmo; *iii*) problemas e soluções de sincronização temporal entre computadores (importante fator para performances musicais baseada em ostinatos ou outros padrões rítmicos repetitivos, que possuem uma unidade de tempo musical que não se altera); e *iv*) a utilização do *beep* do computador para criação musical³⁷.

*Perl-live*³⁸ é uma extensão para um editor de texto chamado *Emacs*³⁹, desenvolvido por um anônimo autodenominado *Vivdsnow*, para execução em tempo-real de fontes em linguagem *Perl*. Na Figura 5 apresento um exemplo de performance predominantemente

³⁷ *Beep* é uma onomatopéia para um som curto, de elevada frequência em *Hz*, utilizada originalmente nos *mainframes* dos anos 1950 (computadores de grande porte) mas ainda existentes em *Desktops* e *Laptops*, durante a inicialização da máquina. Paul Doornbusch (2004) salienta que, o uso do *beep* foi fundamental para as pesquisas realizadas no inicio da segunda metade do século XX, uma vez que tecnologias visuais ainda não tinham sido produzidas, e o som indicava que operações no computador tinham sido realizadas. (DOORNBUSCH, 2004, p.12). Existe documentação comprovando o uso musical do *beep* já com os *mainframes*. Alex di Nunzio (2010) comenta que, no caso de um dos primeiros *mainframes*, o CSIR Mk1 - Sydney, “a presença de um alto-falante (...) não constitua nenhum escopo musical e se tratava de um importante suporte de programação (...) foi o próprio aspecto técnico que estimulou a curiosidade do pesquisador [Geoff Hill], interessado em entender se era possível ir além da produção de sinais sonoros simples, programando o computador para reproduzir sequências musicais mais longas. A primeira melodia tocada através do computador foi *Colonel Bogey*, um motivo popular bastante simples”(NUNZIO, 2010, p. 39). (Tradução nossa de: *La presenza degli alto parlanti, in una macchina come quella di Sidney, non aveva alcuno scopo musicale, si trattava soltanto di un importante supporto alla programmazione (...) Fu proprio quest'aspetto tecnico a stimolare la curiosità dei ricercatori, interessati a capire se fosse tato possibile andare oltre la produzione di semplici bip programmando il calcolatore per la riproduzione di sequenze musicali più lunghe. La prima melodia eseguita attraverso il calcolatore fu Colonel Bogey, un motivo popolare piuttosto semplice*).

³⁸ <<https://github.com/vividsnow/perl-live>>.

³⁹ <<https://www.gnu.org/software/emacs/>>; muito usado por *live coders*. Poderia ser discutido se esse editor não teve influência direta no desenvolvimento do *live coding*, ao permitir o uso de diversas linguagens de programação. No entanto prefiro me afastar dessa questão para evitar ser prolixo.

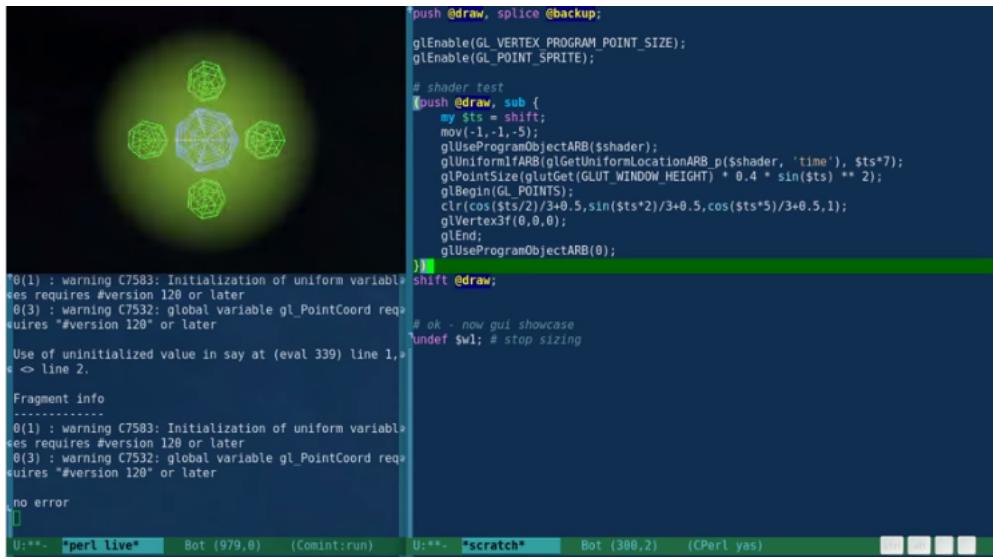


Figura 5 – Editor de texto contendo o código-fonte em Perl, uma janela de vídeo gerada pelo programa e a renderização visual realizada na janela. **Fonte:** <<https://www.youtube.com/watch?v=BCUTu4urY9w>>.

visual utilizando este *plugin* para *Emacs*; é interessante notar que a performance se utiliza da comunicação em rede com outro *software*, *Supercollider*⁴⁰, para realizar alguns experimentos sonoros, mas musicalmente não apresenta muito interesse, uma vez que realiza poucos testes com essa comunicação.

Fluxus, desenvolvido por Griffiths (2008), é um ambiente que possibilita exibir o texto do código com sua respectiva imagem renderizada (ver Figura 6). Neste vídeo é possível notar estruturas rítmicas, melódicas e harmônicas repetitivas, que caracterizam a música de pista, mas com uma diferença: o termo *Music Visualisation* é empregado, isto é, existe uma pretensão de tradução da música sendo executada em imagens abstratas construídas por estruturas geométricas. Outro exemplo multimidiático com o *Fluxus* são *jam sessions* realizadas por um grupo nomeado Kitchen Budapest (2011) (ver Figura 7)⁴¹. Enquanto dança e projeções se colocam em um mesmo nível hierárquico na apresentação, música existe como plano de trilha sonora para a dança⁴², característica por um som com eventos sonoros periódicos em tessitura de baixo, com *delays* bem separados temporalmente (criando um ritmo que decai em amplitude); sons friccionados, que se assemelham a uma

⁴⁰ <<https://supercollider.github.io/>>; é um ambiente e linguagem de programação para síntese sonora e composição algorítmica em tempo-real. Foi desenvolvido originalmente em 1996 por James McCartney (também praticante do *live coding*) e distribuído como *software-livre* em 2002.

⁴¹ Loes Bogers (org.), Zoltán Csik-Kovacs, Krisztián Gergely, Gáspár Hajdu, Gábor Papp (*live coders*), Gábor Czap, Kornél Biharyi, Szilvia Németh, Maria Paz Ramirez T., Mária Roskó, Borbála Anna Simányi, Gábor Török, Dóra Zsámbokréty (Dança), András Molnar, Andrés Ortega, Tamás Marquetant (Música).

⁴² Não sei como a música foi feita, se compratilhando códigos ou de maneira separada. Estou dando aqui um relato perceptivo: primeiro visualizando e escutando o vídeo, e em seguida, realizando posteriormente uma escuta separada do vídeo. Não é uma análise

síntese granular parametrizável; utilização de *synths* e *pads* semelhantes a ambientes *lounge*: gestos pontilhistas e acordes tonais de duração prolongada; materiais sonoros que intencionam ser um bumbo e uma caixa (mas são sintetizados): breves eventos sonoros aperiódicos, variação rítmica, dada uma unidade de tempo.

Deste último grupo, *Kitchen Budapest*, é interessante fazer uma pequena digressão para o começo deste trabalho: no prelo do vídeo é feita uma definição de *live coding* que contrasta com aquela usadas pelo grupo TOPLAP, no início deste capítulo, separando a idéia de músico do *live coder*, enfatizando a prática do programador como animador visual:

Live coding é um tipo de performance onde programador mostra suas habilidades de programação visual em tempo-real. Completamente ao vivo, você vê gráficos animados ganhando vida em projeções, enquanto um grupo de programadores digitam códigos para entreter sua audiência com um espetáculo (audio)visual. (BUDAPEST, 2011, online)⁴³

Aqui é emblemática a concepção de espetáculo: de um ponto-de-vista cênico, uma apresentação de *live coding* (como esta do Kitchen Budapest) se caracterizar por um tipo de apresentação que separa o público do espaço cênico, e os *live coders* no espaço técnico. Não me refiro à posição dos músicos, pois fico em dúvida se os *live coders* estão com os músicos (operando algum *software*), ou se a música estava finalizada em mídia digital.

*Impromptu*⁴⁴ (ver Figura 8) é um ambiente de improvisação audiovisual desenvolvido por Andrew Sorensen e tem por base outro *software* de *live coding* conhecido como *Extempore*⁴⁵. Enquanto o *Extempore* não faz integração do ambiente de edição com a imagem renderizada, o *Impromptu* faz. É importante notar que Sorensen é um membro ativo no grupo TOPLAP.

*Gibber*⁴⁶ desenvolvido por Charlie Roberts (ROBERTS; KUCHERA-MORIN, 2012), é significativo neste trabalho: a presente pesquisa partiu do ponto de vista de entender o suporte utilizado por este *software*, na *internet*, como plataforma para a improvisação musical. A utilização de navegadores como máquinas de síntese e técnicas de sampleamento, foram possíveis com o lançamento da biblioteca *Web Audio API*, uma biblioteca de códigos que possibilita a síntese e sampleamento utilizando o navegador como Processador de Sinais Digitais (DSP). Além dessa camada sonora, existe a camada de imagens, que pode ser feita usando OpenGL ou *P5.js*. Existem vídeos demonstrativos

⁴³ Tradução parcial de: *Live coding is a type of performance where programmers show off their visual coding skills in realtime. Completely live, you see animated graphics come to life in projections, while a group of programmers punches in code to entertain their audience with a (audio)visual spectacle. A group of coders from Kitchen Budapest have been experimenting with this recently, especially with implementing the Kinect in the Fluxus environment. Kinect captures movement data in 3D using infrared cameras, which makes it a perfect combination with dance.*

⁴⁴ <<http://impromptu.moso.com.au/>>, acessado em 27 de junho de 2015.

⁴⁵ <<https://github.com/digego/extempore>>, acessado em 27 de junho de 2015.

⁴⁶ <<http://gibber.mat.ucsb.edu/>>, acessado em 27 de junho de 2015.

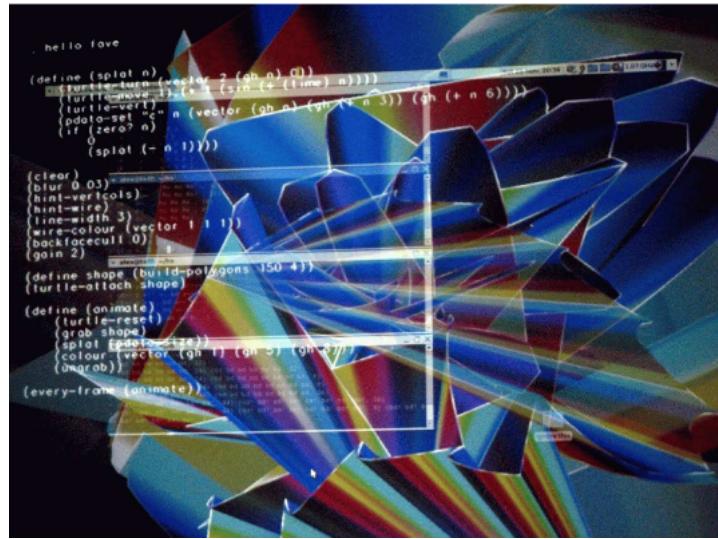


Figura 6 – Texto de código com uma imagem renderizada no mesmo quadro no ambiente *Fluxus*. **Fonte:** <<http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>>.



Figura 7 – Utilização do Fluxus pelo Kitchen Budapest com música e dança interativa através de *video mapping*. **Fonte:** <<https://vimeo.com/30133155>>.

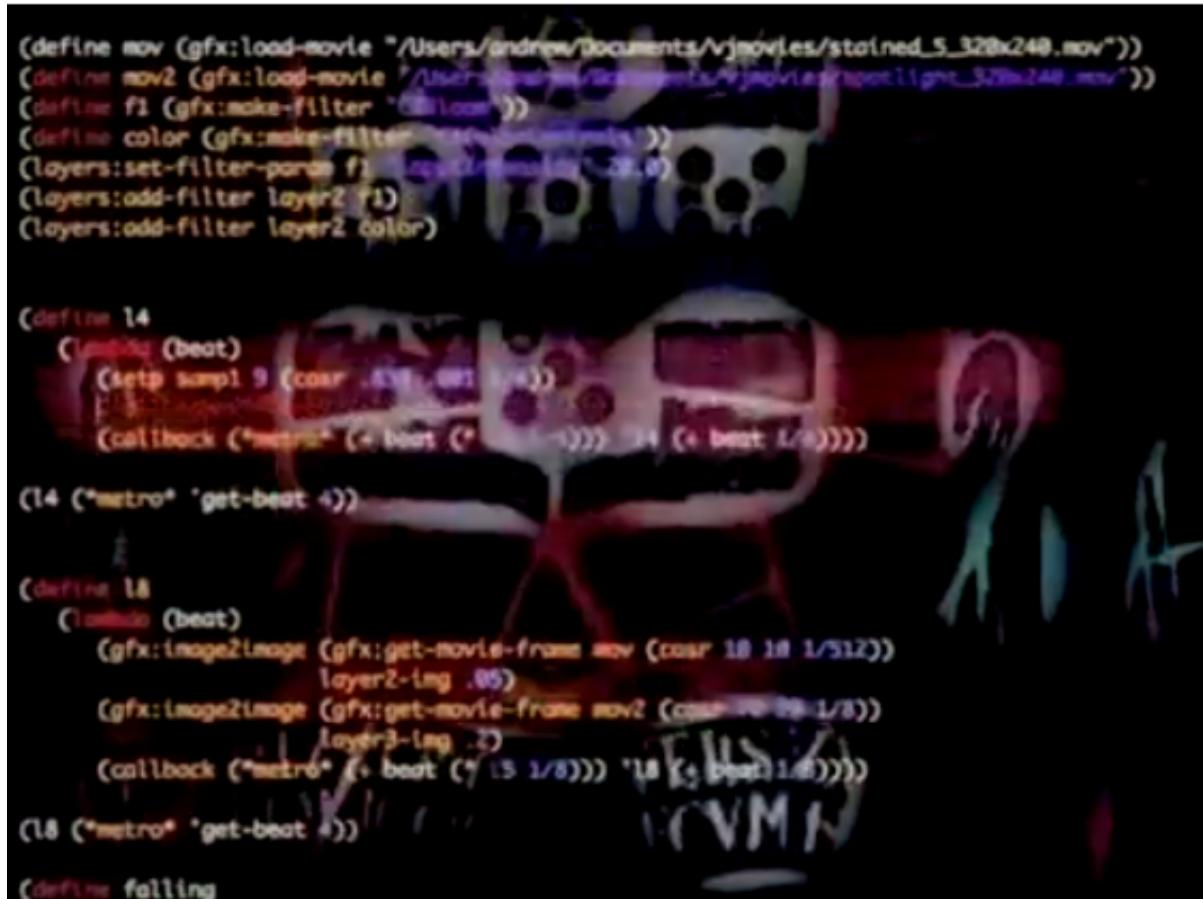


Figura 8 – Captura de tela de um frame de uma improvisação utilizando o *Impromptu*

ilustrando as potencialidades do *software*, como um gráfico de FFT⁴⁷ e nuvens de formas geométricas que se movimentam de acordo com eventos lógicos predeterminados para a imagem e para o som.

1.3.3 Vínculos musicais no *live coding*

No início dessa pesquisa, o objeto de estudo era a relação performer \Leftrightarrow computador, isto é, como o executante se relaciona com o instrumento musical, mais especificamente, em um computador conectado a uma rede de computadores. Durante o processo notei que

⁴⁷ Segundo Steven Smith (2012-06), “é um outro método para calcular a DFT. Enquanto produz o mesmo resultado com outra abordagem, é incrivelmente mais eficiente, geralmente reduzindo o tempo de computação em [escala] de centenas.” (SMITH, 2012-06, Capítulo 12). Tradução nossa de *The Fast Fourier Transform (FFT) is another method for calculating the DFT. While it produces the same result as the other approaches, it is incredibly more efficient, often reducing the computation time by hundreds.* Por sua vez, a DFT (Transformada discreta de Fourier), faz parte de uma família de Transformadas de Fourier, um tipo de análise de sinais inventados por Jean Baptiste Fourier (1768-1830). Outros tipos de transformadas são a de Joseph-Louis Lagrange (1736-1813) e Pierre-Simon Laplace (1749-1827) Laplace. A Transformada de Fourier é notável como aplicação em diversas áreas exatas, como propagação de calor (aplicação original usada por Fourier), análise de espectro de cores em uma imagem, análise de espectro de componentes de um som.

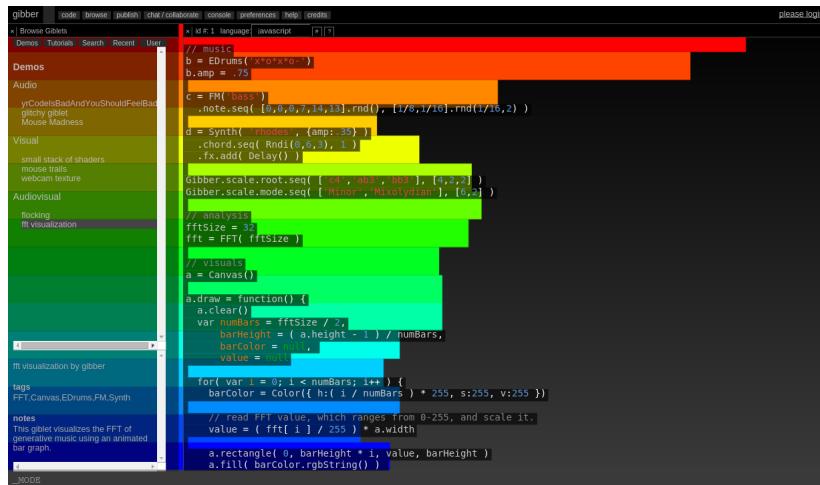


Figura 9 – Ambiente de improvisação audiovisual *online* Gibber, que se aproxima da abordagem do *Fluxus*.

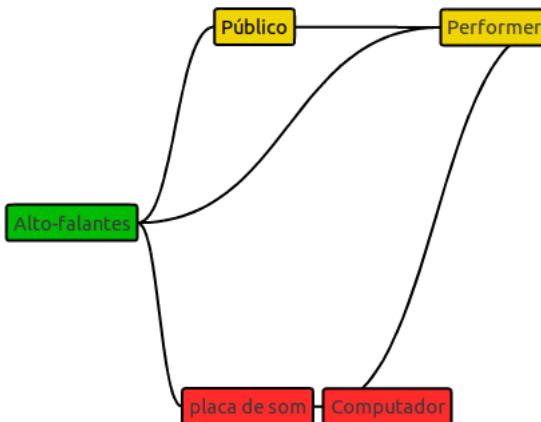


Figura 10 – Diagrama, derivado da Figura 1, enfatizando relações entre pessoas, *hardwares* e dispositivo de projeção acústica, em um âmbito musical.

o ambiente de pesquisa é impróprio para um assunto de especificação demasiadamente especialista e ortopédica. Por isso descartei esta relação como objeto de pesquisa, para focar em aspectos culturais. No [Apêndice A](#) apresento experimentos de uma mesma técnica em diferentes linguagens e no [Apêndice B](#) um código-fonte de síntese sonora em *javascript*.

Desconstruí o mapeamento feito na [Figura 1](#), retirando dispositivos de vínculo visual, como demonstrado na [Figura 10](#). Nesta seção apresentarei exemplos da relação computador \Leftrightarrow placa-de-som; chamo de placa-de-som a interface mediadora entre o computador e os dispositivos de projeção sonora, seja um alto-falante ou outro instrumento, através da comunicação de áudio ou MIDI.

Antes de descartar items do diagrama, apresento exemplos do que será descartado; neste caso interfaces conectadas ao computador. Especificamente, por interfaces eu quero

```
usuario@computador:\$> cat /dev/urandom > /dev/dsp
```

```
usuario@computador:\$> cat /dev/urandom | padsp tee /dev/audio > /dev/null
```

Figura 11 – Comando de DSP similares aos utilizados em uma performance de *livecoding* durante a PDCon09 <<http://sussurro.musica.ufsj.br/pdcon09/>> pelo trio *FooBarBaz*. O primeiro comando é obsoleto e o segundo é sua versão funcional.

dizer placas-de-som, ou dispositivos controladores de instrumentos musicais. O primeiro caso é do grupo brasileiro *EKP* (*Emotional Kernel Panic* de São Carlos–SP⁴⁸) que realizou em 2009 uma performance no PDCon03⁴⁹ mais próxima da *noise-music*. O trio utilizou uma técnica de programação sonora que escreve, a partir de binários de computador (i.e., o código de máquina compilados) diretamente na placa de som de seus computadores.

Em comunicação com Fabbri (2015),

Renato: O Fábio Furlanete⁵⁰ comentou comigo que daria para mandar o kernel⁵¹ para o */dev/dsp*⁵², e daria para ouvir o kernel como binarios. Mas comentou como prova de conceito. (...) Quando fui para Recife soltei essa para Machado, ele era o cara a frente do CDTL, Centro de Desenvolvimento de Tecnologias Livres, pontão de cultura digital de Recife. Aí rolou alguns desenvolvimentos a partir dele com Brasileiro; aí fizeram umas coisas por conta; aí fiz umas coisas por conta; aí com Brazileiro também. Aí fiquei sem notícias sobre. É isso o EKP; na real a proposta transcendeu bastante alguém ou algum grupo.

Guilherme: Mas essa ideia do kernel era do Furlanete, ou já tinha trabalho anterior a isso?

Renato: Não sabemos. Na real parece que já teve umas idéias do tipo.

A técnica funciona apenas para sistemas operacionais *Linux* e escreve dados diretamente no arquivo que é responsável pelo DSP. Exemplos da técnica são ilustrados na Figura 11; ambos são comandos para um ruído branco, com uma diferença: enquanto o primeiro é um comando obsoleto⁵³, o segundo é seu código funcional, que funciona na maioria dos sistemas operacionais do *Linux*⁵⁴.

⁴⁸ Entre eles, Renato Fabbri, Felipe Machado e Ricardo Brasileiro.

⁴⁹ PureData Conference que ocorreu no MIS/SP, organizado, dentre varias pessoas, Alexandre Porres Torres e Paloma Oliveira .

⁵⁰ Professor de Música, Universidade Estadual de Londrina (UEL).

⁵¹ Ou “semente” e o programa principal de um Sistema Operacional (Linux, MacOS, Windows, por exemplo) que gerencia um sistema operacional.

⁵² Arquivo padrão em sistemas Linux anteriores ao protocolo ALSA, responsável pelo processamento de sinais digitais (DSP).

⁵³ O anônimo *ajross* (<<https://news.ycombinator.com/user?id=ajross>>), descreve que isso é devido a uma incompatibilidade com protocolos de áudio e modernos sistemas operacionais (<<https://news.ycombinator.com/item?id=3212928>>).

⁵⁴ Realizei alguns testes em um Ubuntu Linux 14.04 que estão disponíveis no último item do Apêndice C.

```
usuario@computador:\$> cat /usr/bin/python | padsp tee /dev/audio > /dev/null
```

Figura 12 – Comando que captura o código binário da linguagem Python o comando da Figura 11 em sistemas linux

Isso significa que, em uma linha de comando (terminal de sistema Linux, *usuario@computador\$>*), qualquer arquivo binário pode ser concatenado (*cat*), nesse caso um arquivo gerador de números randômicos (*/dev/urandom*), repassado para outro comando (símbolo *pipe*, *|*) e escrito no dispositivo de áudio do computador (*padsp tee /dev/audio*). Este comando retorna um fluxo de dados no terminal, por isso, se não existir interesse no resultado visual, é interessante encaminhar este comando para um “buraco negro” (*/dev/null*). Um outro exemplo pode ser feito capturando outros arquivos binários, como por exemplo, o próprio binário de uma linguagem de programação (por exemplo, *Python*, ver Figura 12). Exemplos sonoros estão como *hiperlnks* no Apêndice C.

É interessante notar que este procedimento de *live coding* cria materiais sonoros bastante ruidosos, mas com uma banda limitada. Enquanto o arquivo */dev/urandom* é um ruído contínuo, outros arquivos poderão conter o mesmo ruído articulando com silêncios e gestos frenéticos. Incluo um espectrograma de uma amostra de áudio seguindo esse procedimento de concatenação de um arquivo⁵⁵ (ver Figura 13). É possível notar uma nuvem de eventos sonoros na região *3kHz* a *7kHz*.

De fato, sons mais complexos podem ser gerados através da programação desses arquivos binários: Heikkilä (2011) publicou um artigo que demonstra procedimentos de síntese sonora que exigem o mesmo procedimento utilizado pelo trio brasileiro. Heikkilä aborda seu trabalho como uma descoberta acidental que seria útil para programar músicas de baixa resolução (*noise-music*, *glitch-music*) de uma maneira bem rápida e otimizada, e que poderia ser útil para o *live coding*:

Durante a história da *computer music* e da composição algorítmica, uma plethora de conceitos técnicos e matemáticos tem sido tentados, indo da simples troca de registros e automatos celulares para modelos complexos e estatísticos. É fácil capturar a impressão que todo simples conceitos usáveis em potencial tem sido pelo menos estudados. A fim de encontrar novos conceitos, computacionalmente simples, úteis para uma música de computadores com baixa complexidade, uma excursão não sistemática foi realizada no espaço de programas de computador muito curtos em que retornam dados sonoros crus. (...) O interesse demonstrado em várias comunidades on-line para o processo de exploração e a música resultante, sugere relevância para as práticas artísticas, tais como *demoscene*, *glitch art*, *chip music* e *live coding*. O

⁵⁵ Para realizar a fixação dos dados em um arquivo de áudio, substitui a segunda parte do comando (*padsp tee /dev/audio > /dev/null*) para um arquivo de áudio cru (*Raw*) e abri em um editor de áudio, que no caso foi o *Audacity*.

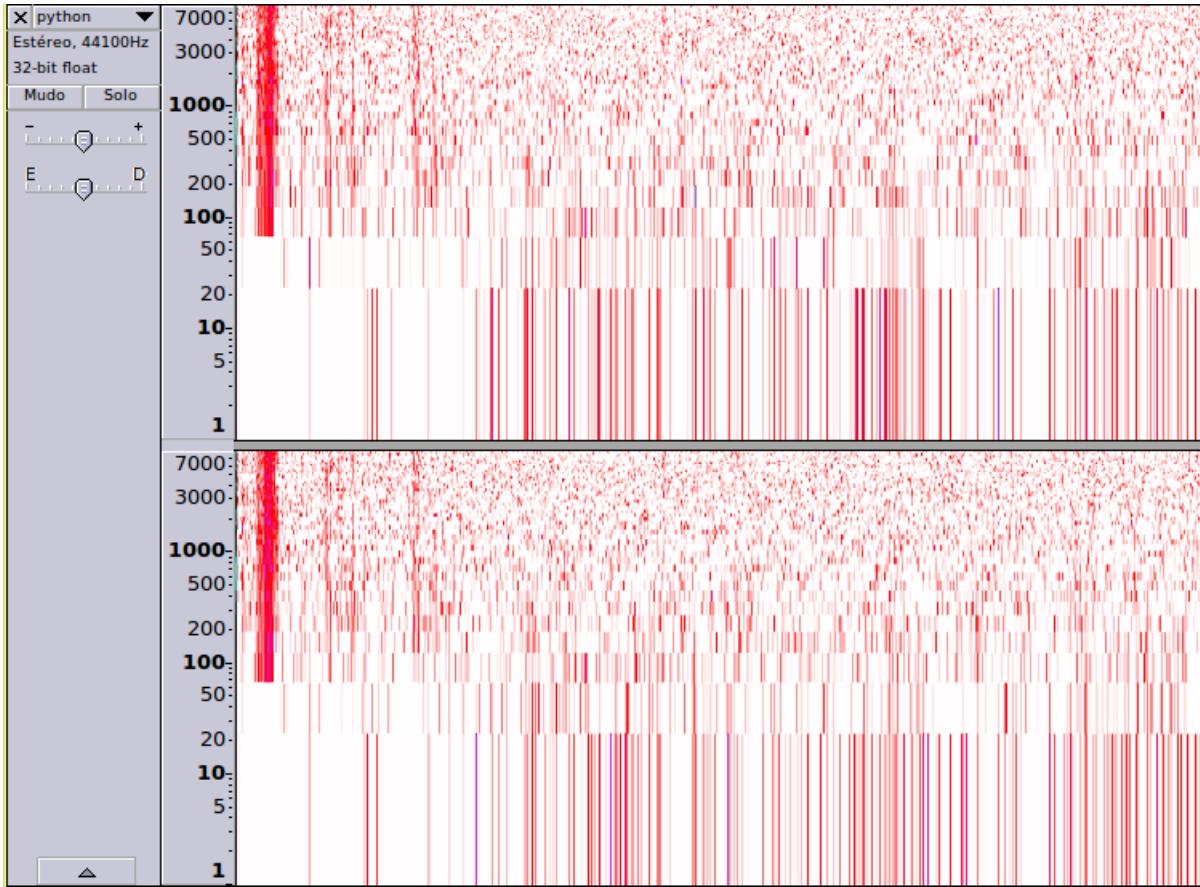


Figura 13 – Comando que captura o código binário da linguagem Python o comando da Figura 11 em sistemas linux

termo *bytebeat* foi recentemente cunhado para se referir a este tipo de música.(HEIKKILÄ, 2011, p .1)⁵⁶

Um exemplo interessante de *bytebeat* é uma performance realizada por Heikkilä sob o avatar *Viznut*, com um *software* desenvolvido por ele chamado *Ibiniz*, onde uma sequência de caracteres é inserida de maneira virtuosística, afim de obter resultados visuais de baixa resolução; musicalmente, o *bytebeat* pode ser incluído na categoria de gênero musical *live electronic dance music* (ver Figura 14). É interessante notar que, durante a performance, o programador busca dizer antes o que vai fazer; assumo que isso cria uma

⁵⁶ Tradução nossa de: *During the history of computer music and algorithmic composition, a plethora of technical and mathematical concepts has been tried out, ranging from simple shift registers and cellular automata to complex physical and statistical models. It is easy to get the impression that every potentially useful simple concept has already been at least superficially studied. In order to find new, computationally simple concepts useful for low-complexity computer music, a rather non-systematic excursion was undertaken into the space of very short computer programs that output raw sound data. (...) The interest shown on various on-line communities towards the explorative process and the resulting music suggests relevance to artistic practices such as demoscene, glitch art, chip music and live coding. The term “bytebeat” was recently coined for referring to the type of music.*

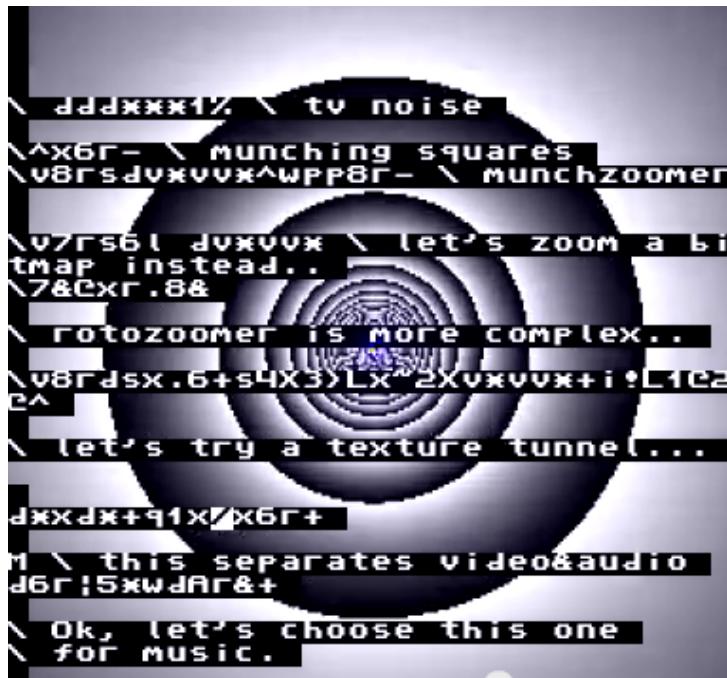


Figura 14 – Ambiente de improvisação audiovisual *Ibiniz*, onde resultados visuais e sonoros podem ser obtidos através de uma manipulação direta em *bytes* do computador. O código é extremamente curto. **Fonte:** <<https://www.youtube.com/watch?v=aKMrBaXJvMs>>.

relação de proximidade maior que os outros exemplos descritos até o momento, no sentido de que busca colocar o espectador como parte do processo.

Passando para outro trabalho, onde o *live coder* controla instrumentos musicais externos, apresentarei apenas o *The disklavier sessions* de Sorensen e Ben Swift (ver Figura 15)⁵⁷, não utilize os sinais sonoros de uma placa de som, mas provavelmente de tecnologia MIDI. Discutirei um exemplo de *live coding* que se utiliza de dispositivos externos ao computador.

Através da programação com o software *Extempore*⁵⁸, Sorensen controla o mecanismo interno de um piano acústico. Nesse vídeo, além do controle do instrumento musical através de dispositivos adequados, o músico externaliza para o público o processo de criação e edição do código-fonte que controla o dispositivo.

Uma pequena análise pode ser valiosa do ponto de vista da ecologia de gêneros musicais: de 0' até 1'35'', a performance indica uma sessão de improvisação diversa do ambiente *rave*, sendo minimalista com harmonias tonais, alargamento/compressão temporal de estruturas rítmicas e valorização de pequenas dissonâncias; de 1'36'' até 2'28'', realiza

⁵⁷ Um vídeo promocional pode ser visualizado em <<https://vimeo.com/50061269>>, acessado em 27 de junho de 2015

⁵⁸ <<https://github.com/digego/extempore>> ou <<http://extempore.moso.com.au/>>, acessado em 27 de junho de 2015.



Figura 15 – Captura de tela de um vídeo demonstrativo da peça *Disklavier Sessions* de Andrew Sorensen e Ben Swift (2013). **Fonte:** <<https://vimeo.com/50061269>>.

uma automatização de uma improvisação de *jazz*, intercalação de gestos ascendentes e/ou descendentes que repousam em acordes complexos que excedem a capacidade técnica da mão humana (talvez com adição e alteração de intervalos de 7^a, 9^a, 11^a, 13^a ou superiores); e de 2'29" a 6'41", retorna a um minimalismo instrumental em tom menor, expondo a técnica de transformação gradual de um motivo repetitivo.

Outro programa linguagem que merece menção é o *iXiLang* de Magnusson (2012)⁵⁹. Segundo o próprio desenvolvedor, no README do código-fonte diz que:

O ambiente *ixilang* é um sistema visual simples que apresenta um controle de alto-nível sobre definições de sintetizadores e samples no SuperCollider. A idéia central é a de representar eventos em uma disposição espacial, fundindo assim o código musical e partituras musicais. A partitura está ativa, ou seja, se um método é executado na partitura, ele muda em tempo real. (MAGNUSSON, 2012, online)⁶⁰

Uma improvisação demonstrativa de Magnusson está disponível desde 2009 no Youtube (ver Figura 16)⁶¹; o vídeo apresenta um editor de texto customizado, com fundo preto, junto com um *prompt* de comando do SuperCollider⁶², com o qual o *iXiLang* foi

⁵⁹ <<https://www.github.com/thormagnusson/ixilang>>, acessado em 27 de junho de 2015.

⁶⁰ Tradução nossa de *The ixi lang live coding environment is a simple visual system presenting a high entry-level control over synth definitions and samples in SuperCollider. The core idea is to represent events in a spatial layout, thus merging musical code and musical scores. The score is active, i.e., if a method is performed upon the score, it changes in real time.*

⁶¹ <https://www.youtube.com/watch?v=nmqEcil_O4U> link quebrado

⁶² <<http://superollider.github.io/>>, acessado em 27 de junho de 2015. Ambiente e linguagem de programação *open-source* para síntese sonora e composição musical com algoritmos.

desenvolvido. Proponho uma breve análise do vídeo para encotrar práticas musicais. O vídeo inicia com o comando *grid 4*, que cria no próprio editor caracteres conhecidos como *pipes* para definir uma unidade de tempo (| | | |). Em seguida um modo de tonal na música é selecionado, através do comando *scale minor*. Após isso três funções são criadas, *aa*, *bb* e *cc*; é interessante notar que em tais funções, definidas pelo caractere \Rightarrow , os *pipes* definem a subdivisão em 4 tempos (4 espaços de caracteres), ou uma célula rítmica vazia. Letras diversas (maiúsculas ou minúsculas) são colocadas nesses espaços vazios; cada letra representa um material sonoro que pode ser uma gravação. A ordem das letras é tomada da seguinte forma: se estiverem diretamente abaixo dos *pipes* de tempo e dentro dos *pipes* de compasso, serão tocados em tempos fortes ou fracos; se estiverem deslocadas, abaixo e à direita/esquerda do *pipe*, ritmos diversos começam a ser definidos com materiais sonoros pré-gravados.

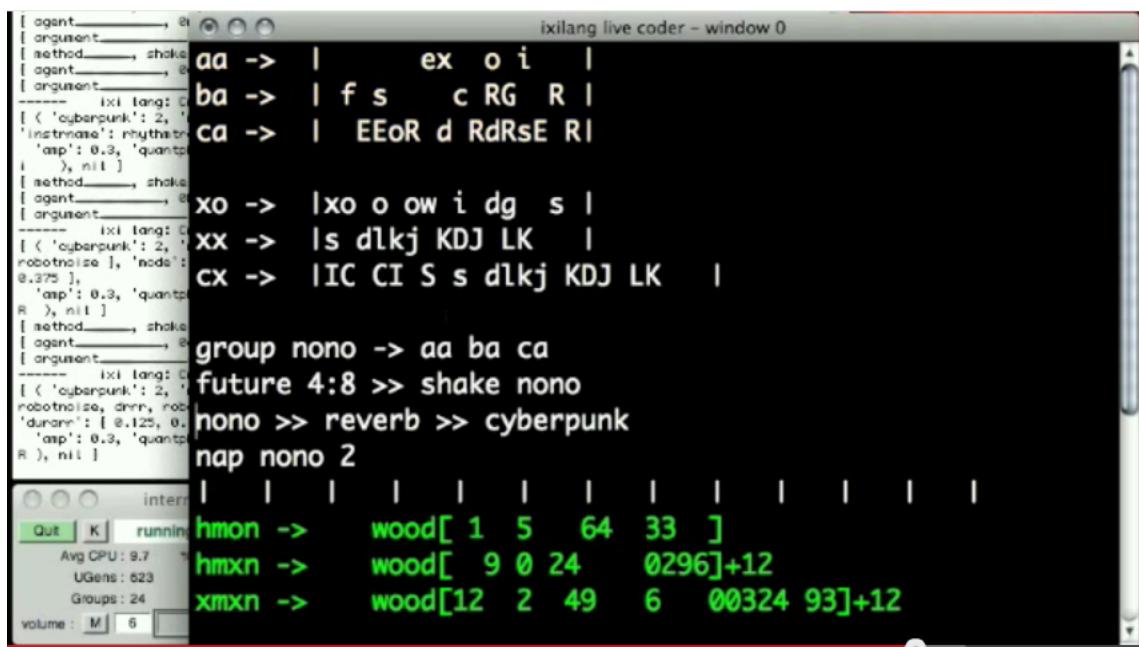


Figura 16 – Captura de tela de um vídeo demonstrativo do ambiente *iXiLang* de Thor Magnusson (2009). **Fonte:** <<https://www.youtube.com/watch?v=nmqEciLO4U>>.

Um comando interessante do ponto de vista composicional é o comando *future 4 : 8 >> swap nono; nono*; *nono* foi definido como um grupo instrumental de três funções diferentes (*xa*, *xb*, *xc*), semelhantes aos *aa*, *ba*, *ca*, através do comando *group nono \Rightarrow aa ab ac*. Isto é *nono* é um grupo de células rítmicas. O comando *swap* permuta a ordem dos caracteres escritos no editor, alterando as célula musical programaticamente, que até então era pré-definida. *future 4 : 8* define uma razão, um intervalo de tempo iterativo em que *swap nono* será executado. O comando completo define, portanto que, aqueles ritmos de amostras sonoras definidos no grupo *nono* serão de momentos em momentos permutados automaticamente pelo ambiente *iXiLang*, expondo uma estrutura musical dinâmica. Outra

característica interessante é a adição de efeitos, como *reverb* (espaço acústico virtual), *distort* (distorção) e *cyberpunk*; deste último é possível começar a extrair alguns gêneros musicais, no sentido de que a performance traz um *ethos* do gênero eletrônico de dança que surgiu no final da década de 1970. Verificando o código-fonte do *iXiLang*, na linha 251⁶³, o efeito *cyberpunk* é definido por um Objeto chamado *Squiz* que, segundo o manual do SuperCollider é:

Um algoritmo simplista de troca de altura. Isso não deve soar “natural”; o som é uma reminiscência de uma mistura estranha de filtro, modulador-em-anel e troca-de-altura, dependendo do sinal de entrada. O algoritmo funciona cortando o sinal em fragmentos (...) e apertando os fragmentos no domínio do tempo (i.e., simplismente tocando eles de volta mais rápido do que vieram), deixando silêncios entre eles (...) Este UGen⁶⁴ é dedicado ao *Suburban Base Records* (não soa igual a ele, mas foi parcialmente inspirado nele).⁶⁵

Suburban Base é uma gravadora reconhecida por divulgar estilos eletrônicos de dança de pista como *jungle* e *drum'n'bass*, ao que se pode associar o termo *Hardcore Rave*. O *cyberpunk*⁶⁶, por sua vez é um tema característico de livros e filmes de ficção científica, que descrevem sociedades distópicas, megacorporativas, altamente tecnológicas e com personagens marginalizados; em trilhas sonoras de filmes *cyberpunk*, tomadas são acompanhadas de sons com atmosferas densas, isto é, timbres complexos com parciais harmônicos e inarmônicos, efeitos sonoros que aplicam distorções complexas (como o *Squiz*) e, no caso da performance de Magnusson, um discurso temporal mais lento, como no caso da desaceleração do tempo de 60 BPM para 30 BPM⁶⁷.

1.3.4 Vínculos do executante com a partitura e público no *live coding*

O vínculo performer \Leftrightarrow público possui alguma relação com a pesquisa se observado do ponto de vista que tipo de música o *live coder* transmite para o público. Discutirei nesta subseção algumas características desta relação a partir da concepção do programa de computador como partitura musical. Por sua vez, a partitura pode ser pensada como

⁶³ <<https://www.github.com/thormagnusson/ixilang/XiiLang.sc>>, acessado em 27 de junho de 2015.

⁶⁴ *Unit Generator*. “UGens representam cálculos de sinais [de áudio]. Eles são os blocos básicos de construção de definições de sintetizadores”. Tradução nossa de: *UGens represent calculations with signals. They are the basic building block of synth definitions.* <<http://doc.sccode.org./Classes/UGen.html>>, acessado em 27 de junho de 2015.

⁶⁵ <<http://doc.supercollider.org/Classes/Squiz.html>>. Tradução nossa de: *A simplistic pitch-shift algorithm. It's not meant to sound natural; it's sound is reminiscent of some weird mixture of filter, ring-modulator and pitch-shifter, depending on the input. The algorithm works by cutting the signal into fragments (...) and squeezing those fragments in the time domain (i.e. simply playing them back faster than they came in), leaving silences imbetween. (...) This UGen is dedicated to Suburban Base Records. (It doesn't sound like them, but was half-inspired by them)..*

⁶⁶ *Cybernetic punk*.

⁶⁷ *Beats Per Minute* ou batidas por minuto é a unidade temporal utilizada em metrônomos mecânicos e eletrônicos para controle preciso do ritmo musical. [desnecessario](#)

um dispositivo usado em formações orquestrais e em instituições de produção de conhecimento musical, como universidades, escolas de música. Afim de investigar a existência de gêneros musicais realizados com programas pensados como partituras, em um ambientes institucionalizados, voltarei a desconstruir o mapa na [Figura 17](#); ali podemos observar as seguintes relações: A) músico-programador \Leftrightarrow computador; B) público \Leftrightarrow computador; C) músico-programador \Leftrightarrow alto-falante; e D) público \Leftrightarrow alto-falante . Da comparação do item A com o [Item a](#) do segundo parágrafo da página 17, abaixo da pergunta em negrito, proponho a observação de relações derivadas: 1) o grupo de músicos-programadores está usando um mesmo *software* de edição dos *scripts* (no entanto isso não é uma regra); 2) o grupo pode estar restrito a um conjunto de problemas, relativos ao ambiente em que se insere (universidade, *night club*, palco); e 3) o grupo oferece soluções para a materialização do fazer musical.

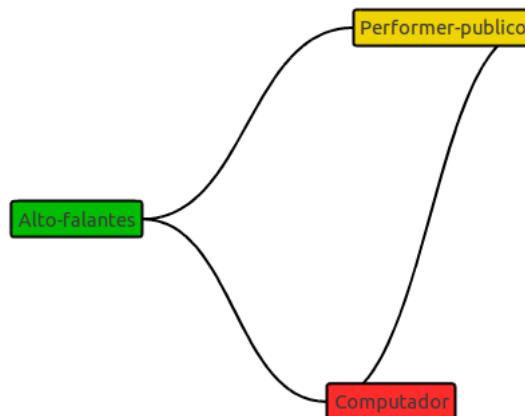


Figura 17 – Diagrama de relações entre *performer*, *hardware/software* e dispositivos de projeção.

O item A), se observado do ponto de vista de um pensamento especialista, pode originar assuntos de estudos simbólicos e formalização semântica, como discutido no começo do capítulo a partir de [Sorensen, Swift e Riddel \(2014\)](#):

Significado no contexto do *livecoding* é portanto multifacetado – um entrejogo complexo de significados simbólicos, computacionais e internalizados no corpo. (...) Neste ensaio nós buscaremos desempacotar alguns dos muitos significados do *livecoding*, prestando uma atenção particular para as semânticas formais que são tão proeminentes na prática do *livecoding*, com este compromisso que apresenta o código fonte e a importância dos algoritmos. ([SORENSEN; SWIFT; RIDDEL, 2014](#), p. 2)⁶⁸

⁶⁸ Tradução nossa de *Meaning in a livecoding context is therefore multifaceted – a complex interplay of symbolic, computational and embodied meaning. (...) In this essay we attempt to unpack some of livecoding's many meanings, paying particular attention to the formal semantics which are so proeminent in livecoding practice, with this commitment to display of source code and the importance of algorithms.*

Este sentido semiótico será descartado deste trabalho por parecer um assunto que não posso conhecer teórico; por outro lado, a expressão “muitos significados do *livecoding*” é bastante pertinente na pesquisa até então realizada. Lembrando que o *live coding* está sendo observado pelo ponto de vista de uma multiplicidade de gêneros musicais, farei comentários nos items *B), C), D), 1), 2) e 3)*, no 1º parágrafo desta subseção, tendo em mente que existe uma tradição eletroacústica presente nesta diversidade.

O item *B)* pode ser um problema demasiado complicado para o escopo deste trabalho, no sentido de que estou assumindo que o público não interage com o computador, como indicado no **Item b** da página 17. No entanto, tomei conhecimento de um trabalho de praticante ativo de *live coding* que problematiza esta questão, ao externalizar pedaços de códigos de um *patch* de PD para um espaço onde o público é o performer; *crackOP\$4* (ver [Figura 18](#))⁶⁹ é uma instalação de André Damião, compositor formado em composição pela UNESP. O espaço utilizado é o de instalação artística apresentado no SESC-SP. Nesta instalação o público pisa em cima de sensores de pressão, que visualmente se parecem com os *bangs* ou *toogles* de um *patch* de PD. *Bangs* (quadrados com uma circunferência inscrita) e *toogles* (quadrados com um X) tem no PD a tarefa básica de disparar um evento ou ligar/desligar eventos; em realidade são pedaços de código modulares, isto é, pode-se utilizar vários para diferentes eventos, visuais ou sonoros. Sensores de pressão, estilizados como esses pedaços de código estão espalhados em um espaço e, ao pisar em cada um, uma pessoa, parte do público, envia comandos para o computador, interferindo no ambiente audiovisual. Na dimensão musical, é possível ouvir eventos sonoros ruidosos e contínuos (algumas vezes parecendo com buzinas) e glissandos em alta velocidade, caracterizando uma prática musical próxima da *noise-music*.

Dos items *C)* e *D)*, descarto temas de psicoacústica ou cognição musical que podem emergir, mas reflito se ela não pode ser observada por outro ponto-de-vista, o como as pessoas que escutam e nomeiam uma manifestação musical. Esse ponto de vista requer (na minha opinião) que a pergunta do segundo parágrafo da página 17, em negrito, seja refeita para **como e quantas músicas tem sido produzidas, em um tipo de computador específico, o Laptop, com alto-falantes de qualquer tipo, e classificadas em gêneros musicais pelos agentes?**⁷⁰

Do problema *1)*, sugiro a investigação de algumas manifestações musicais a partir de dois *softwares* fundamentais para o desenvolvimento do *live coding*, em situações de improvisação musical utilizando o computador *laptop*, nos Estados Unidos e na Inglaterra, SuperCollider e ChucK⁷¹. Em algumas situações específicas que comentarei abaixo, os

⁶⁹ <<https://vimeo.com/125100592>>, acessado em 27 de junho de 2015.

⁷⁰ Seria possível também perguntar: “Qual desses agentes nomeia o gênero, os executantes ou o público?” Esta pergunta norteia o trabalho sendo realizado para o [Capítulo 3](#) e que pode ser contextualizado no [Apêndice D](#).

⁷¹ <<http://chuck.cs.princeton.edu/>>, acessado em 27 de junho de 2015

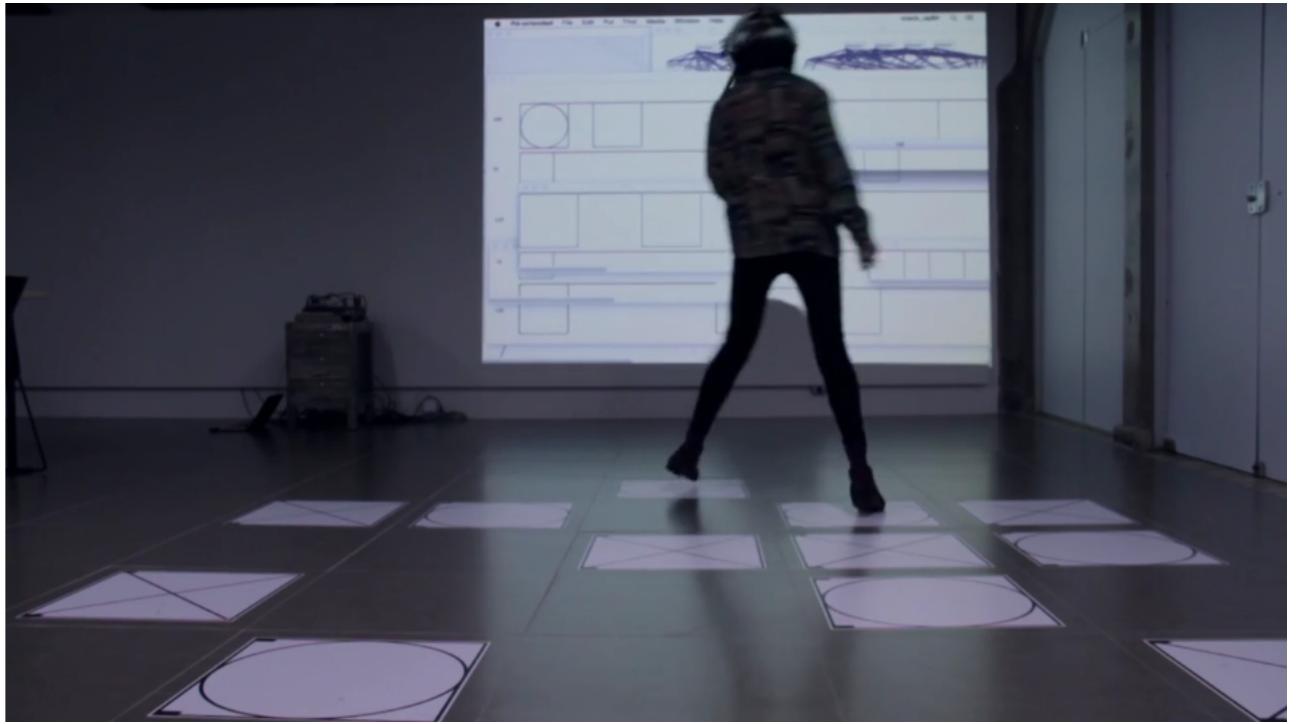


Figura 18 – *crackOP\$4* de André Damião, instalação multimídia: projeção, sensores de pressão, computador e sistema sonoro. **Fonte:** <<https://vimeo.com/125100592>>.

grupos usam ou um ou outro *software*.

Do problema 2), utilizei o termo *universo de conceitos* citado na página 20, segundo parágrafo, que McLean (2006) faz em seu artigo “Music improvisation and creative systems”. Este artigo discute uma formalização lógica de uma improvisação, ou um Modelo de Improvisação (IM)⁷² observando-a como um sistema criativo, “uma coleção de processos, naturais ou automáticos, que são capazes de alcançar ou simular comportamentos que em humanos seria considerado criativo” (McLean, 2006 apud BODEN, 1990, p. 1). A esta coleção de processos, McLean chama de Estrutura de Sistemas Criativos (CSF)⁷³. Embora

⁷² *Improvisation Model*; Cf. McLean, 2006, p. 2-5.

⁷³ *Creative Systems Framework*. A letra *c* é tomada como um conceito tomado na improvisação e *U* é um conjunto de conceitos; no entanto *c*, sozinho, não é capaz de abranger o espaço conceitual de uma performance como um todo; assim, segundo McLean, “Nós devemos considerar o conceito [*c*] como uma união de certos estados de *O'*[conjunto de objetos], *F'*[conjunto de características] e *P'*[conjunto de processos]”(Cf. McLean, 2006, p. 2-4). Tradução parcial de *Instead we should consider a concept as a union of a certain state of O', F' and P'*. Objetos são “entidades cognitivas ou perceptivas tais como acordes, notas ou pausas”; para cada objeto, *F'* “lista os parâmetros disponíveis, descrevendo propriedades compartilhadas do objeto, como altura ou modulação”; e *P'* “lista os processos que governam como um objeto ou uma característica mudam com o tempo” (McLean, 2006, loc. cit.). Tradução nossa de: *cognitive or perceptual entities such chords, notes or rests (...) lists the available features parameters, describin shared properties of an object, such as pitch or modulation (...) lists the processes which govern how an object or feature changes over time*. Em comunicação pessoal com o compositor Jean Pierre Caron (2015), essa formalização foi comentada como *algebrização da improvisação*. É interessante notar que este comentário poderia ser usado para caracterizar o CSF

estou um pouco perdido nessa enumeração...

não utilizando este termo no mesmo contexto de McLean (ali quem define o universo de conceitos é o improvisador que vai executar uma performance), me propôs a apontar no Capítulo 3 um universo de conceitos como gêneros musicais, demonstrados no Apêndice D.

Do problema 3), surge uma dúvida aqui como essas resoluções surgem com a dispersão da prática em diferentes ambientes específicos; estou supondo aqui que estes grupos utilizam uma tecnologia que permita a intercomunicação de pessoas e compartilhamento de trabalhos realizados em diferentes partes do planeta.⁷⁴. Suponho aqui que tais resoluções tenham suas bases em uma ou outras teorias musicais (que depende do passado musical do músico-programador), ao que formalizei na pergunta **que teorias musicais estão intimamente conectadas às culturas musicais, em forma de gêneros musical, na comunidade de músicos-programadores de live coding?**

Explorarei um pouco mais o item 1), 2), enquanto o item 3) será explicitado na próxima seção. O vínculo músico-programador \Leftrightarrow computador é mediado por um código-fonte, que pode ser considerado uma partitura. Por partitura não digo a notação musical feita em papel, com pentagramas, tablaturas, cifras, dinâmicas, articulações, fraseado e tempo (embora o computador possa assistir o compositor para este uso). Neste trabalho a partitura é estabelecida pela criação/edição de *scripts* (ver Item c do terceiro parágrafo da página 17). Estes, por sua vez, podem ser pensados como um híbrido de programa e partitura, no sentido que códigos especificam modelos sonoros e como estes eventos ocorrem. Thor Magnusson (2011) chama estes códigos contendo algoritmos de síntese e definições formais dos eventos sonoros de "algoritmos como partituras" (MAGNUSSON, 2011); Fenerich, Obici e Schiavoni (2014) chamam o código que materializa o som de "programação-partitura" (FENERICH; OBICI; SCHIAVONI, 2014, p. 5) ⁷⁵. Uma programação-partitura necessita ser executada em um ambiente de programação apropriado. Por exemplo, a programação-partitura .pd é executada no PD; a partitura-programação .sc é executada no SuperCollider; e a partitura-programação .ck no ChucK. Cada ambiente que executa uma programação-partitura permite um circuito entre atividades de criação/edição, percepção acústica e cognição musical; isto é, o circuito de performance descrito nos itens a) f) na página 17.

como um pensamento ortopédico (proposto por Santos (2008),) no sentido de que uma proposição com um ponto de vista lógico seria demasiadamente especialista e que poderia excluir elementos da diversidade de práticas de improvisação. No entanto ela pode ser útil neste trabalho justamente por, no contexto apresentado até aqui, ser uma ferramenta para uma formalização de uma ecologia de gêneros musicais.

⁷⁴ Um caso de intercomunicação e compatilhamento é o SoundCloud, rede social que investiguei no Capítulo 3; podem ser feitas por mensagens pessoais (*mailing*), *likes*, e *retweets*.

⁷⁵ O termo usado é específico de um contexto onde os autores realizaram a peça *Marulho Transoceânico*: “registro de uma apresentação realizada em 30/08/2012 em João Pessoa e Berlim simultaneamente. Nesta pecaa os resíduos sonoros decorrentes dos dispositivos de transmissão e recepção de áudio em rede foram explorados, de forma a tornar evidente ao ouvinte tais processos técnicos que estão embutidos na transmissão e comunicação sonora via protocolos digitais que nos cercam” (FENERICH; OBICI; SCHIAVONI, 2014, p. 2). Penso que, mesmo em um contexto diferente, a palavra é útil para a situação.

No caso do SuperCollider existem performances chamadas de *sessions*, ou seções de improvisação. Esta manifestação podem ser virtuais (por vídeos no Youtube, ver [Figura 19](#)) ou presenciais, encontros que agregam indivíduos com diferentes passados musicais e técnicas de programação.

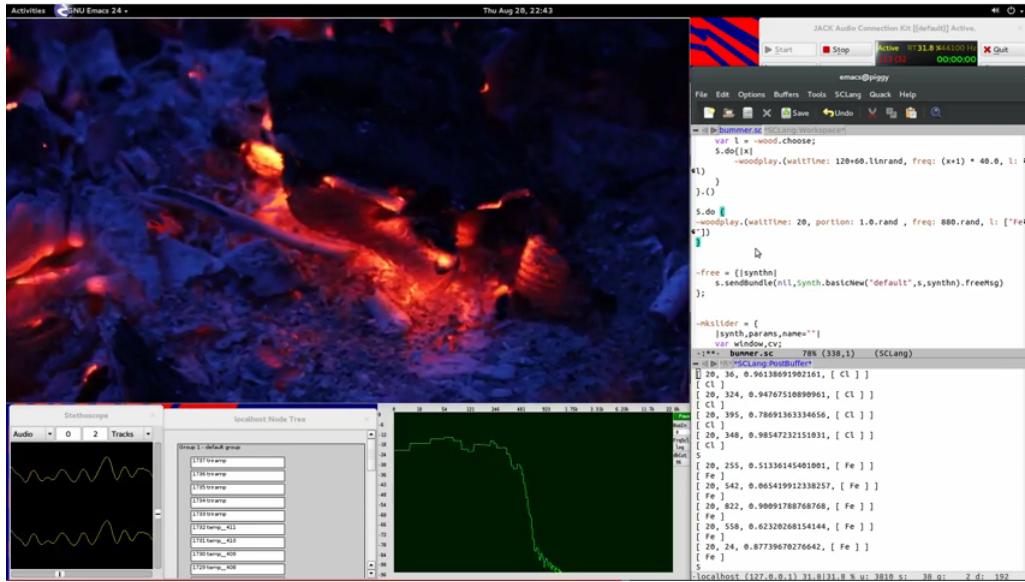


Figura 19 – *Supercollider Live Code Session: The summer is over soon* por Abraham Hindle; [Fonte:<https://www.youtube.com/watch?v=wJ7_5WCAj9Y>](https://www.youtube.com/watch?v=wJ7_5WCAj9Y).

A improvisação de Abraham Hindle apresenta uma característica semelhante ao *Perl-live* apresentado na [subseção 1.3.2](#): utilização do editor *Emacs* e separação de janelas do vídeo, com a diferença que apresenta janelas de auxílio à performance, como um espectrograma⁷⁶. O vídeo é um quadro que apresenta a transformação da lenha sendo consumida pelo fogo em um recipiente de metal, enquanto o áudio inicia como um contínuo sonoro do espaço acústico que o microfone da câmera de vídeo capturou; este áudio é então processado no SuperCollider: a partir de 51" o fluxo de áudio possui interrupções que poderiam ser interpretadas como *glitches* (mas que provavelmente não são intencional, mas um erro causado pelo alto custo de processamento que o computador está realizando durante a sessão [*session*], tocando um vídeo, processando som e gravando o que está sendo feito). O processamento de áudio então altera este espectro inicial, permeando o espaço acústico com um ruído de banda limitada contínuo, na região dos médio/médio-agudos (aproximadamente em 1'20"); em torno de 2'15" sons granulares emergem dos pequenos estalos causados pela brasa. Em 3'17" aproximadamente o ruído de banda limitada, com poucas intervenções das brasas, ficam em primeiro plano. Em 4'20" aproximadamente, um som agudo e simples (do ponto de vista de espectro) oscila em amplitude (provavelmente este som seja resultado de uma prática de retroalimentação) e disputa o plano principal com

⁷⁶ Espectrograma é uma ferramenta de análise geralmente baseada na tecnologia FFT, para extrair informações pertinentes do som como uma soma de sons simples (componentes senoidais).

esta descrição eh necessaria? melhor mais sintético

o ruído de banda limitada (4'40''), momento em que uma nuvem de grãos (brasas) entra em jogo. Em 5'30'', aproximadamente, um segundo ruído de banda limitada aparece, na região dos graves. Filtragens em bandas de frequência alteram o espectro do som do vídeo, deixando apenas o ruído de banda limitada na região dos graves ficar em primeiro plano, em uma dinâmica *pp* (até 8'10''). A partir de então sons complexos e simples intervêm no plano do ruído de banda limitada. Em 9'33'' a nuvem de grãos reaparece disputando o plano principal de escuta e some em torno de 10'10''. Em 10'34'', o vídeo sofre um corte para uma imagem avançada do processo de consumo da lenha pelo fogo, assumindo uma atmosfera noturna com muitas cinzas e o brilho característico de regiões da lenha que ainda estão sendo consumidas. Em 10'40'' a nuvem de grãos (brasas) assumem uma posição no espaço de escuta à frente do ruído, sendo que em 12'17'' já se percebe a volta do primeiro ruído de banda estreita, no início da performance. Até 15'49'' aproximadamente, o performer dá mais atenção a uma sensação psicoacústica de quietude⁷⁷, com pequenas intervenções próprias do espaço acústico que está sendo capturado, quando a nuvem de grãos retomam o plano principac. Em 16'52'' emerge um ruído de banda estreita na região média, finalizando a improvisação em 17'36''. Esta abordagem discursiva do fazer musical caracteriza uma prática de composição musical eletroacústica divulgada em espaços de produção de conhecimento (universidades), com todos seus clichês sonoros (sons ambientes, granulação, filtragem). **bom! talvez só essa frase bastasse.**

Sobre as *sessions* realizadas presencialmente, ocorre além do fenômeno de agregação física de pessoas, uma agregação virtual dos computadores utilizados: tecnologias de comunicação em rede são usadas para compartilhamento dos códigos musicais. Através da abertura de canais de comunicação em rede (portas), algoritmos são compartilhados entre os participantes da *jam session*. Durante a escrita deste trabalho, tomei conhecimento de um interessante exemplo, uma pesquisa recente no IME/USP, realizada por Rodolfo Valente e equipe: *live coding* com redes remotas entre Ceará, São Paulo (Brasil), Califórnia e Michigan (EUA) utilizando uma versão customizada do *SuperCollider* chamada *SuperCopair*⁷⁸. A utilização de redes de computadores como parte da prática com o *SuperCollider*, possibilita ver o *live coding* como um paradigma de programação conversacional (*conversational programming*); isto é, onde os intérpretes se comunicam virtualmente – por comentários no código ou em *chats*–, para discutir o que está sendo realizado (prática semelhante ao de *Viznut* com seus *bytebeats*, quando parecia que conversava com o público). Além dessa sociabilização, é importante lembrar que, em ambos os casos, o de uso de redes locais ou redes remotas, o áudio não é compartilhado na rede, apenas as variáveis, funções ou

⁷⁷ Segundo Haouli e Mannis (2011), “silêncio, mais do que uma propriedade física, é um estado da percepção, uma sensação psicoacústica de quietude, acionada por fatores externos e internos. O baixo nível de intensidade sonora, próximo ao limiar de audição, pode induzir a percepção de silêncio.” (HAOULI; MANNIS, 2011, p. 3)

⁷⁸ O código-fonte do trabalho pode ser encontrado em <<https://atom.io/packages/supercopair>>, acessado em 27 de junho de 2015.

pedaços de códigos; isso leva a considerar que a execução do mesmo código em diferentes computadores oferece a liberdade de escolha de qual alto-falante utilizar. Neste sentido, *sessions* realizadas presencialmente possue a característica de que muitos *live coders* irão executar os sons a partir de seus próprios alto-falantes (do computador, ou conectados ao computador), criando um ambiente específico onde podemos ouvir improvisações diferentes (ou semelhantes) em diferentes pontos do espaço acústico.

No caso do ChucK⁷⁹ o fenômeno ocorre de forma particular, característico da configuração de uma integração de grupos praticantes de *live coding* com universidades. Neste último, alto-falantes, computadores e outros dispositivos são utilizados de maneira padronizada. Por usar de maneira padronizada quero dizer que os *agentes ativos* (*live coders*) estão seguindo regras específicas de um ambiente institucionalizado; esta institucionalização é materializada com a apropriação do termo *Orquestra*, sendo que o posicionamento espacial de cada *live coder* depende de uma estrutura semi-circular. Por exemplo, o PLork⁸⁰ (ver Figura 20) e SLork⁸¹ (ver Figura 21) são *orquestras de laptop* que executam música eletroacústica ao vivo através do ChucK.

Um terceiro exemplo de orquestra de laptops, embora não sendo específico do ChucK, é o BLO⁸² (ver Figura 22), que utiliza alto-falantes no padrão de apresentações audovisuais, o estereofônico de dois canais e a disposição espacial dos *live coders* segue a prática do *performer* como um técnico que fica em uma *sala técnica* controlando o espetáculo multimidiático.

Em um documentário jornalístico sobre o PLOrk, realizado pela *Fox News*⁸³, é possível notar como os músicos-programadores utilizam seus instrumentos-computadores para criar e editar programações-partituras. A captura de tela demonstra um modo da performance em que os músicos ficam sentados, editando códigos de computador. Embora não tendo conhecimento de como esta orquestra é organizada internamente, sugiro que a palavra *orquestra* pressupõe *naipes instrumentais*. Músicas são produzidas em um ambiente artístico institucionalizado, um espaço de produção de conhecimento musical nos Estados Unidos da América que, segundo Nunzio (2010), é onde “nascem os primeiros resultados importantes da música informática (...) devido ao contexto [histórico] específico em que se desenvolve a música informática, determinado nos anos seguintes à Segunda Guerra Mundial”(NUNZIO, 2010, p. vi) ⁸⁴; em outras palavras, a performance do PLOrk está

⁷⁹ “ChucK é uma nova linguagem de programação de áudio para síntese em tempo real, composição e desempenho, que executa em sistemas operacionais disponíveis no mercado. ChucK apresenta um novo modelo de programação concorrente baseada em tempo, que tem suporte a controle de taxa dinâmico, simultâneo e múltiplo e a habilidade de adicionar, remover e modificar o código, sob demanda, enquanto o programa está em execução, sem parar ou reiniciar” (Central de Programas do Ubuntu, 2015).

⁸⁰ Princeton Laptop Orchestra, mais informações em <<http://plork.princeton.edu/index.php>>.

⁸¹ Stanford Laptop Orchestra, <<https://www.youtube.com/watch?v=YCVKE7HG0Mg>>.

⁸² Barcelona Laptop Orchestra, <<http://blo.cat/>>.

⁸³ <<https://www.youtube.com/watch?v=q1Ymixm1xmw>>

⁸⁴ Tradução nossa de: *Una particolare attenzione è dedicata alla situazione americana, giacché in*

carregada de uma tradição conhecida como *Computer Music* (CM).

Sobre o termo *Computer Music*, Barry Truax (2000) e Fernando Iazzetta (2009) discutem que este termo é de difícil definição. Truax dicute que *Computer Music*, “(...) como possa ser definido hoje, continuou a tradição musical ocidental - européia - como uma forma de arte abstrata, i.e., uma forma de arte onde os materiais são projetados e estruturados principalmente através de suas relações internas.”(TRUAX, 2000, p. 1)⁸⁵. Já Iazzetta, a partir de Guy Garnett, discute a questão pelo viés da concepção do computador como ferramenta de composição, e se produções musicais utilizando esta ferramenta poderiam utilizar o tempo *Computer Music* como delimitadora de um gênero musical:

O computador, quando se tornou ferramenta de auxílio à composição, ampliou ainda mais o nível de abstração envolvido no processo criativo musical. Frequentemente tem se discutido se seria possível usar o termo *computer music* para distinguir um gênero, uma corrente, ou uma estética musical, embora hoje cautelosamente se coloque a ideia de que o termo trata simplismente de ‘música que não pode ser criada sem o uso de computadores’⁸⁶. O fato de se confundir uma tecnologia com um gênero musical pode parecer, inicialmente, mera ingenuidade, fruto da sedução que as tecnologias eletrônicas e digitais exercem nos dias de hoje. Porém não se deve desprezar de imediato esta questão. Se o computador não define necessariamente uma estética, ela define um modo de operação que o diferencia de outras tecnologias. (IAZZETTA, 2009, p. 150)⁸⁷

esta citação reflete Flusser em ‘a escrita’ (ler)

As performances do PLOrk transparecem esses modos de operação, o que a torna singular em relação a uma orquestra tradicional, no sentido de como sons são produzidos, e não sua organização institucional. Alguns desses modos de operação com o computador, seriam, dentre várias, algumas técnicas de síntese sonora modeladas por diferentes algoritmos computacionais; dentre estas técnicas, algumas são consideradas básicas no processo de transmissão de conhecimento em universidades (ao qual listamos algumas poucas, embora a lista seja limitada se comparada com o número de todas as técnicas possíveis): Aditiva, Subtrativa, AM, FM, Granular.

O exemplo do SLOrk (ver Figura 21) é curioso do ponto de vista instrumental, de quais equipamentos estão sendo usados e uma formação semi-circular da orquestra, com o maestro na frente, semelhante ao PLOrk. Um vídeo disponível⁸⁸, “Twilight” (2013) de Ge Wang, demonstra uma orquestra interativa, isto é, no sentido que está utilizando

quell’ambiente nasce e consegue i primi importanti risultati la musica informatica (...) Il contesto specifico, quindi, in cui si sviluppa la musica informatica si determina dagli anni successivi alla seconda guerra mondiale

⁸⁵ Tradução de: (...) however it may be defined today, has continued the Western, i.e. , European, tradition of music as an abstract art form, i.e., an art form where the materials are designed and structured mainly through their internal relationships.

⁸⁶ “The aesthetics of Interactive Computer Music”, *Computer Music Journal*, v. 25, n1, p.21-33.

⁸⁷ Grifo nosso.

⁸⁸ <<https://www.youtube.com/watch?v=chA-4GRCb-I>>.



Figura 20 – Captura de tela de uma matéria jornalística a respeito da *Princeton Laptop Orchestra* (PLOrk). Esta imagem enfatiza a padronização de computadores e alto-falantes de alta-resolução, bem como ilustra uma mímica de orquestra. **Fonte:** <<https://www.youtube.com/watch?v=q1Ymixm1xmw>>, acessado em 27 de junho de 2015.

dispositivos manuais externos, sensíveis ao movimento espacial tridimensional; tais dispositivos são tomados como controladores de movimento do instrumentista, ocorrendo uma tradução dos gestos corpóreos em gestos musicais. Do ponto de vista da relação com o público, a performance configura-se como uma prática de acordo com a etiqueta da orquestra tradicional; do ponto de vista de uma improvisação da programação-partitura, a performance tem poucas codificações, e não parece ser *live coding* no sentido até então discutido; ocorre no entanto uma re-parametrização de códigos pré-definidos no tempo, através dos gestos corpóreos, estes também padronizados pelo movimento do maestro. Essa padronização dos movimentos pode ser associada à textura sonora percebida, como uma espécie de homofonia: vários intérpretes realizando uma mesma classe de movimento – gestos lentos ou rápidos, ascendentes ou descendentes –, que possuem microvariações, de acordo com o movimento individual de cada músico.

É interessante aqui fazer uma pequena análise de uma performance do BLO, a première de “Duet for two Nat King Cole’s and scratched disk”⁸⁹ em 2008 na *Escola Superior de Musica de Catalunya*. A peça é contrastante com as diversas performances musicais até então utilizadas, na qual improvisam com *loops*⁹⁰ da imagem e voz de Nat

⁸⁹ <<https://www.youtube.com/watch?v=YCVKE7HG0Mg>>.

⁹⁰ Uma gravação inteira, ou um trecho, que fica em constante reiteração; o material é tocado de um ponto A (que pode ser seu início ou depois), tocado até outro ponto B (que pode ser o fim, ou antes), e então retorna ao ponto A. Se B for tocado antes de A, então é possível dizer que sua leitura ocorreu “de-trás-para-frente”. A velocidade de leitura da amostra também pode ser modificada, acelerando ou desacelerando, modificando o espectro sonoro da gravação (e consequentemente nossas percepções



Figura 21 – Captura de tela de um vídeo promocional da respeito da *Stanford Laptop Orchestra* (SLOrk). **Fonte:** <<https://www.youtube.com/watch?v=chA-4GRCb-I>>, acessado em 27 de junho de 2015.

King Cole: 1) o cantor se apresenta e fala sobre uma apresentação(18'' a 42''); 2) o cantor mantém uma contagem regressiva de cinco (5) segundos, que fragmenta seções de *loops* (45'' a 1'44''); 3) elaboração de uma atmosfera de canção (1'45'' a 3'43''), onde existem: *i*) um material harmônico (piano e baixo) também em *loop*; *ii*) um ruído proveniente da reiteração do material sonoro anterior, que acaba criando um pulso; e *iii*) a manipulação da velocidade de leitura de um trecho da voz de Nat King Cole, um material sonoro “it’s not” ; 4) enunciação da frase “besame mucho” e acompanhamento de uma *rumba*, também em *loop*, agora com o material da voz de Nat King Cole criando materiais melódicos ascendentes e descendentes (3'44'' a 4'44''); 5) sobreposição de quadros (e seus respectivos trechos sonoros), criando um efeito de *delay* (4'44'' a 5'48''); e 6) finalização através da manipulação das cores da imagem – até então em preto-e-branco– ,e no plano sonoro: *i*) uma torrente de eventos eletrônicos; e *ii*) alteração na velocidade de leitura da voz de Nat King Cole para o final, onde ele convida a entrar na cortina atrás dele.

nao entendi essa descrição descolada

1.4 Ecologia de práticas musicais

Para elaboração de músicas que podem ser categorizadas em gêneros musicais, estou considerando que, no momento da improvisação, seguindo o Modelo de Improvisação (IM) de McLean (2006), o *live coder* admite um universo de conceitos *U*. Este universo de conceitos inclui, além dos gêneros musicais nas páginas anteriores, outros termos que podem delimitar linguagens e técnicas criativas. Tais linguagens técnicas e criativas, segundo o

que universo
eh esse???

de altura e timbre). Para mais informações a respeito da técnica e seus resultados sonoros, sugiro a leitura do segundo capítulo do “The theory and technique of electronic music” de Puckette (2006)



Figura 22 – Captura de tela de uma performance do *Barcelona Laptop Orchestra* (BLO). Esta imagem é da última seção da performance decupada, onde ocorre a manipulação dos pixels e uma experiência torrencial de eventos sonoros. **Fonte:** <<https://www.youtube.com/watch?v=YCVKE7HG0Mg>>, acessado em 27 de junho de 2015.

fragmento de texto de Ward et al. (2004) apresentado no início do capítulo 1, na página 18, são: 1) “(...) exploração de espaços algorítmicos abstratos” (WARD et al., 2004, p .204): Música Algorítmica (MA); 2) “Outro experimento mental pode ser encarado como um DJ ao vivo codificando e escrevendo uma lista de instruções para o seu *set*”⁹¹: Música de Pista ou prática DJ ⁹².

No manifesto “*Live Algorithm Programmin and Temporary Organization for its Promotion*”, em um trecho posterior àquele citado na página 18, Ward et al. realizam o seguinte comentário:

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 processador” 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 deixa-

⁹¹ WARD et al., 2004, loc. cit.

⁹² Deste último é mais apropriado, a partir de Collins e McLean (2014), utilizar os termos *code DJing*, *live computer dance music* como práticas que foram incluídas naquilo que é conhecido por *algorave* (Cf. COLLINS; McLean, 2014, p. 355-357). Na subseção 1.4.3, descreverei um pouco mais sobre o seu significado.

das para crescer, e sugere abrir mão do controle dos nossos processos, deixando eles “brincarem ao vento”.⁹³

A partir desta citação, é possível incluir outros dois itens aos 1) e 2) apresentados no primeiro parágrafo desta seção: 3) Música Generativa (MG); e 4) Música de Processador, ao qual tomarei como um empréstimo conceitual de Música Processual (MP).

Admitindo de antemão esta multiplicidade de técnicas criativas e gêneros musicais na relação *live coder* ⇔ computador ⇔ público, reformularei novamente perguntas que tenho feito até aqui como, **por meio de quais técnicas criativas o *livecoding* se manifesta como um universo de conceitos, na forma de gêneros musicais?**

O método utilizado nessa sessão para investigar técnicas criativas incluídas no universo de conceitos do *live coding*, é procurar por menções em de autores a respeito das *Música Algorítmica* (que será abreviada para MA, ver [subseção 1.4.1](#)), *Música Generativa/Música Processual* (MG, MP, ver [subseção 1.4.2](#), [subseção 1.4.2.1](#) e [subseção 1.4.2.2](#)), e *code DJing* (DJ, ver [subseção 1.4.3](#)). Estes termos serão contra-argumentados com outros autores, entre eles [Reich \(1968\)](#), [Eno \(1978\)](#), [Kramer et al. \(1999\)](#), [Roads \(2001\)](#), [Wooler et al. \(2005\)](#) [Malt \(2006\)](#), [Walker e Kramer \(2006\)](#), [Essl \(2007\)](#), [Cope \(2008\)](#), [Iazzetta \(2009\)](#), [Mailman \(2013\)](#) e [Collins e McLean \(2014\)](#) [Castelões, Oliveira e Franco \(2015\)](#).

1.4.1 Apropriação da Música Algorítmica no *live coding*

Seguindo a nomenclatura de David [Cope \(2008\)](#), o que chamo de Música Algorítmica (MA), pode se encaixar nas seguintes áreas de pesquisa: Composição Musical Automática/Algorítmica (AMC), Composição Musical Generativa (CGM), Composição Assistida por Computador (CAC), Sons Gerados por Computador (CGS), Assistência Gerada por Computador (CGA) e Composição Gerada por Computador (CGC) ([COPE, 2008, p. ix-x](#))⁹⁴. Neste trabalho a MA segue a perspectiva da improvisação musical utilizando a CGS em conjunto com a CGC, e Linguagens de Programação (PL), em detrimento das Linguagens Visuais de Programação⁹⁵ para performances eletrônicas. [as siglas atrapalham](#)

⁹³ [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”.*

⁹⁴ Tradução nossa de: AMC—Algorithmic Music Composition; CGM—Computer Generated Music; CAC—Computer Aided Composition; CGS—Computer Generated Sound; CGA—Computer Generated Assistance; e CGC—Computer Generated Composition.

⁹⁵ [COPE, 2008](#), loc. cit.. Tradução nossa de Programming Languages (PL). e Visual Programming Language (VPL).

Em seu artigo “*Concepts et Modèles, de l’imaginaire à l’écriture dans la Composition Assistée par Ordinateur*”, Mikhail Malt (2006) discute o significado de algoritmo na AMC como modelo. Embora seja uma concepção externa à abordagem utilizada (Sons Gerados por Computador, CGS e Composição Gerada por Computador, CGC), acredito que a discussão de *algoritmos como de modelos* pode ser útil na investigação dos gêneros musicais no *live coding*.

“O conceito de modelo cobre um campo semântico complexo, com um espectro muito amplo de significados, variando de formalizações matemáticas às maquetes e moldes” (MALT, 2006, p. 12)⁹⁶, sendo que este campo se divide em três significados: A) modelo como figuração; B) modelo como esquema diretor; C) modelo como fixador interdisciplinar de leis de domínios não musicais:

O primeiro significado é o modelo como “figuração”. Neste sentido, é uma materialização de muitos níveis de enunciados de uma ciência em um objeto concreto, algumas vezes quase autônomas, que a intuição ou pensamento tem facilidade para identificar (...) O segundo significado atribuído à noção de modelo é o de “esquema diretor”, por assim dizer uma transcrição abstrata (porém controlada pelo pensamento lógico e matemático) de uma realidade concreta, empírica, cujo estudo direto (quando este é possível) é feito de relações aproximadas. (...) Finalmente o modelo que participa dessas duas naturezas, retirado de um domínio de objectos particulares e beneficiado como um fixador para as leis de outro domínio. Temos como exemplo o uso do modelo "massa-mola"(modelo essencialmente mecânico) para modelar fisicamente a produção de som em um clarinete. (MALT, 2006, p. 12)⁹⁷

O segundo significado foi escolhido por permitir um paralelo com uma outra definição de algoritmo musical, também a partir do termo “modelo”; segundo Karl Essl (2007), “Dentro do campo de composição algorítmica, o algoritmo constitue um modelo abstrato que define e controla alguns ou todos aspectos estruturais da música” (ESSL, 2007, p. 108)⁹⁸; é importante notar que Essl definiu o algoritmo musical baseado em regras e como um método semi-automático de composição, e não necessariamente o algoritmo computacional para a música. O autor discute o problema de se definir o que é MA, a

⁹⁶ Tradução nossa de *La notion de modèle couvre un champ sémantique complexe, avec un spectre très large de significations, qui va des formalismes mathématiques aux maquettes et aux moules.*

⁹⁷ Tradução nossa de *La première signification est le modèle comme “figuration”. Dans ce sens, il est une matérialisation à plusieurs niveaux des énoncés d'une science dans un objet concret, quelquefois presque autonome, que l'intuition ou la pensée ont des facilités à cerner. (...) Le deuxième sens donné à la notion de modèle est celui de “schéma directeur”, c'est-à-dire une transcription abstraite (mais contrôlée par la pensée logique et mathématique) d'une réalité concrète, empirique, dont l'étude directe (quand elle est possible) ne donne que des relations approximatives. (...) Il arrive enfin que le modèle participe de ces deux natures, qu'il soit prélevé sur un domaine d'objets particuliers et qu'il profite comme fixateur pour les lois d'un autre domaine. Nous avons comme exemple l'utilisation du modèle de « masse – ressort » (modèle essentiellement mécanique) pour modéliser physiquement la production de sons dans une clarinette.* Grifo nosso.

⁹⁸ Tradução nossa de: *Within the field of algorithmic composition the algorithm constitutes an abstract model which defines and controls some or all structural aspects of music.*

partir de uma perspectiva histórica, que vai desde o *Musica enchiriadis* (c.895) de Hucbald de St. Amande, passando pelo *Illiac Suite* (1957) de Lejarn Hiller (1924-1994), e a Música Assistida por Computador⁹⁹

Nas palavras de Philippe Leroux (2008),

Assim, todas tradições musicais tiveram algum recurso com modelos, sejam eles biológicos, por exemplo, a simetria bilateral do corpo, o andar, o respirar, etc.; cósmicas, por exemplo, a trajetória e distâncias dos planetas que serviram de modelo para o monocórdio de pitágoras, que servia tanto para a medição astronômica como instrumento musical; cosmogônicas e teológicas como vistos acima; ou científicas, por ex, equações ou operações matemáticas. (...) Ao longo da história da música, o uso de modelos naturais tem sido alternado com modelos mais teóricos, como aqueles que derivam de matemática. Assim, na verdade, música nunca foi feita sem um modelo” (LEROUX, 2008, p. 150).¹⁰⁰

Neste trabalho, reforço a idéia de que estou considerando o modelo do ponto de vista da CGS e CGC, e portanto, como uma formalização lógico-matemática que está em constante processo de modificação em um computador (improvisação).

Para que essa improvisação de formalizações lógico-matemáticas de modelos abstratos em computadores se materialize na performance musical, é necessária a utilização de linguagens de programação. Na época do ILLIAC, por exemplo, linguagens específicas foram inventadas para os componentes daquele computador utilizado, uma família de linguagens conhecida como *Assembly*. Nos exemplos apresentados na seção 1.2 deste trabalho, as linguagens são Perl, LISP, javascript, linhas de comando ou mini-linguagens. Nelas, os modelos abstratos são escritos na forma de *funções*, *classes*, *métodos*, dependendo do paradigma de programação da linguagem. Em geral, considerarei apenas as funções.

Dependendo da estrutura destas funções, é possível formalizar em um programa de computador, leis de um domínio de conhecimento específico (geralmente das ciências exatas ou biológicas) para a prática musical. Aqui descrevo que, com linguagens de programação, é possível usar dados brutos como material para composição musical. Fernando Iazzetta (2009) exemplifica esta situação na metáfora de "(...) flutuações das bolsas de valores [que] podem dar origem ao vai-e-vem rítmico de uma determinada passagem musical" (IAZZETTA, 2009, p. 151). Por outro lado, estruturas mentais que caracterizam a informação das ciências podem não ser as mesmas que regem a informação musical; meu ponto de vista é que a formalização de uma passagem musical se dá em nível estrutural

⁹⁹ Cf. ESSL, 2007, p. 109-112.

¹⁰⁰ Tradução nossa de *Hence, all musical traditions have recourse to models, be they biological, for example the bilateral symmetry of the body, of walking, of breathing, etc.; cosmic, for example the trajectory and distances of the planets that served as a model for Pythagora's monochord, which was both an astronomical measuring toll and a musical instrument; cosmogonic, and theological, as seen above; scientific, for example the composition of matter, equations or mathematical operations; literary, etc. (...) Throughout the history of music the use of natural models has alternated with that of more theoretical models, such as those that derive from mathematics. Thus in fact, music has never done without models.*

acima da formalização dos sons processados digitalmente (DSP), que são outros modelos matemáticos representando ondas sonoras. Assim tais modelos em externos ao DSP, e seu uso musical, não garantem que esta sonoridade seja aceitável no plano sensitivo. Iazzetta coloca esse problema a partir da seguinte forma:

As chamadas músicas algorítmicas, inauguradas com a *Illiad Suite* de Hiller e Isaacson em 1957, seguem o modo tradicional de compor em que certos modelos e padrões gramaticais são empregados na criação de cada nova música. (...) Entretanto, nas composições algorítmicas, especialmente nas realizadas sob o rótulo de *experimentais*, modelos e padrões não mais necessariamente precisam se remeter a questões musicais. Qualquer estrutura, gramática ou modelo pode, em princípio, ser transposto para o âmbito sonoro com a intenção de produzir música. (...) Uma estrutura qualquer que apresente um comportamento interessante num determinado contexto pode ser facilmente transposta a outro dentro do domínio digital (...) Certamente essas transposições de um campo a outro não destroem a coerência interna dos fenômenos transpostos, mas de nenhuma maneira asseguram a geração de uma coerência musical, pelo menos não no nível perceptivo.¹⁰¹

Esta idéia de transpor um modelo científico para a elaboração de algoritmos musicais, é conhecida como *sonificação*. A partir de Edward D. Cope, Castelões, Oliveira e Franco (2015) designam sonificação como “conversão de dados de um domínio a outro”; a utilização do termo “conversão” para os pesquisadores brasileiros é neutra em comparação ao que Cope chama de “tradução (ou normalização) informações tradicionalmente inaudíveis para o campo de frequências audíveis - servindo, a partir daí, como matéria-prima potencial para qualquer técnica que faz uso do som” (COPE, 2008, p .xi). Outra definição vê a sonificação como um subconjunto de um programa de pesquisa conhecido como *Exibição auditiva* (*Auditory Display*, abreviado como AD). Segundo Walker e Kramer (2006), a International Community for Auditory Display (ICAD), sonificação é definida como:

(...) a utilização de áudio não falado para transmitir informação. Mais especificamente, sonificação é a transformação das relações de dados em relações percebidas em um sinal acústico para efeitos de facilitar a comunicação ou interpretação. Por sua própria natureza, sonificação é interdisciplinar, integrando conceitos de percepção humana, acústica, design, artes e engenharia. Assim, o desenvolvimento de representações auditivas eficazes de dados vai exigir colaborações interdisciplinares usando o conhecimento combinado e os esforços de psicólogos, cientistas da computação, engenheiros, físicos, compositores e músicos, juntamente com a perícia de especialistas nas áreas de aplicação a ser examinados. (WALKER; KRAMER, 2006 apud KRAMER et al., 1999, p. 1)¹⁰²

¹⁰¹ IAZZETTA, 2009, loc. cit.

¹⁰² Tradução nossa de *Sonification is defined as the use of nonspeech audio to convey information. More specifically, sonification is the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation. By its very nature, sonification is interdisciplinary, integrating concepts from human perception, acoustics, design, the arts, and engineering. Thus, development of effective auditory representations of data will require interdisciplinary collaborations using the combined knowledge and efforts of psychologists, computer*

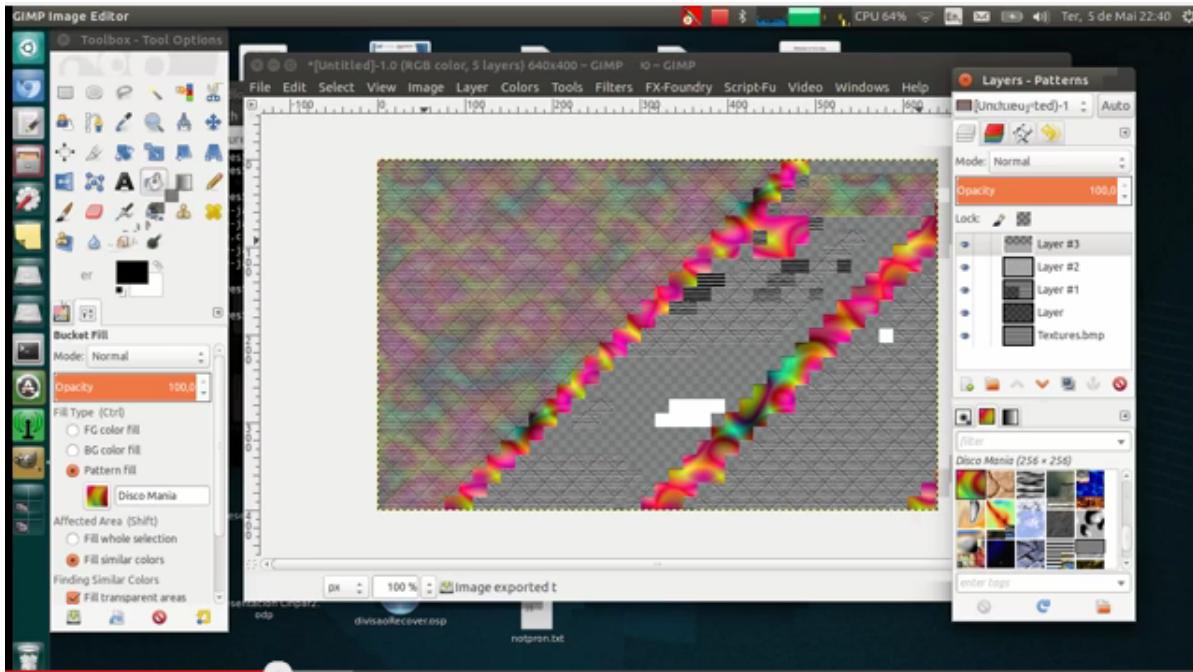


Figura 23 – *Texturas Interativas*. Ensaio da performance a ser realizada no dia 06/05/2015 no Solar da Baronesa em São João Del Rei. **Fonte:** <<https://www.youtube.com/watch?v=6cwAQxIQ8fM>>.

Kramer et al. (1999) ainda descrevem tecnologias de sonificação anteriores à prática de composição musical, tais como o Contador Geiger, sonar, ferramentas que auxiliam químicos cegos a examinar dados espectrográficos de infravermelho e monitores para controle de produção em fábricas (KRAMER et al., 1999, online, 2º parágrafo do item 2. Introdução). Qualquer que seja a aplicação, na composição musical, modelos abstratos ou esquemas diretores podem ser: 1) formalizações para assistência à composição, que podem gerar partituras; 2) formalizações que controlam parâmetros de uma síntese sonora por DSP; e 3) formalizações que controlam o comportamento de amostras pré-gravadas.

Das práticas observadas no *live coding*, tenho a dúvida se a sonificação é a mais utilizada no contexto da seção 1.2, no sentido que o inverso é predominante, isto é, a visualização dos sons. Dos exemplos apresentados na seção 1.2, a prática de *live coding* que mais se encaixa na idéia de sonificação do que é inaudível, é a conversão do programas executáveis (fluxo de informações binárias) em ruído, descritas nas Figura 11 e Figura 12 da subseção 1.3.3, nas páginas 36 e 37. Ainda no processo de escrita desta seção, tomei conhecimento de um trabalho de Flávio Schiavonni intitulado *Textures*¹⁰³ e *Texturas Interativas* (ver Figura 23), elabora materiais sonoros periódicos através da sonificação de padrões repetitivos em imagens digitalizadas.

scientists, engineers, physicists, composers, and musicians, along with the expertise of specialists in the application areas being addressed.

¹⁰³ <<https://github.com/flschiaffoni/textures>>, acessado em 27 de junho de 2015

Por outro lado, parece que a sonificação por modelos mais tradicionais (como matemática ou economia) no *live coding*, necessita de uma camada anterior à improvisação, possível na pré-codificação da performance; isto é, se escolhe um modelo, codifica-se em um arquivo específico que posteriormente será modificado na performance de improvisação. Quero dizer que, no *live coding*, tais formalizações caminham mais para a invenção do que a utilização de um modelo científico *a priori*. De qualquer forma, a utilização de algoritmos computacionais no *live coding* estão tão enraizadas nas práticas descritas na seção 1.2, seja na dimensão visual ou sonora que, independente do gênero musical apontado, um ou vários modelos matemáticos serão utilizados como materiais musicais; parafraseando Ward et al. (2004), poderia ser possível dizer que no *live coding* “o algoritmo se torna a essência da música”¹⁰⁴

1.4.2 Apropriação da Música Generativa no *live coding*

Foi grifado na páginas 53 deste trabalho um comentário do manifesto de Ward et al.: “Termos como ‘música generativa’ e ‘música de processador’ tem sido inventados e apropriados para descrever esta nova perspectiva de composição.”¹⁰⁵ A Música Generativa (MG) por sua vez é um tema que, segundo Rene Wooler et al. (2005), apresenta perspectivas multidisciplinares para a MA, mais especificamente neste trabalho como CGS (Sons Gerados por Computador) e CGC (Composição Gerada por Computador), incluindo abordagens: *i*) Linguísticas/Estruturais; *ii*) Interativas/Comportamentais; *iii*) Criativas/Procedurais; e *iv*) Biológicas/Emergenciais.

Descreverei nas próximas duas subseções apenas as apropriações Criativas/Procedurais e Linguísticas e Estruturais, tanto pela falta de tempo em falar sobre isso, quanto por uma familiaridade maior com esses assuntos.

1.4.2.1 Música de algoritmos simples

Segundo Wooler et al., a música procedural é uma “Música resultante de um conjunto de processos colocados em movimento pelo compositor, tais como ‘In C’ de Terry Riley e ‘It’s gonna rain’ de Steve Reich” (WOOLER et al., 2005 apud ENO, 1996, p. 1)¹⁰⁶; o que é chamado de “procedural” por Wooler et al., é chamado por Joshua Mailman (2013) em seu artigo “*Agency, Determinism, Focal Time Frames and Processive Minimalist Music*”, música de processos mínimos, ou música minimalista de processos determinísticos.

¹⁰⁴ Cf. WARD et al., 2004, p. 245-246. Trecho da citação já utilizada na p. 24: “Contudo, alguns músicos exploram suas idéias como processos de *software*, muitas vezes ao ponto que o *software* se torna a essência da música”.

¹⁰⁵ WARD et al., 2004, loc. cit.. Tradução nossa de *Terms such as “generative music” and “processor music” have been invented and appropriated to describe this new perspective on composition.*

¹⁰⁶ Tradução nossa de *Music resulting from processes set in motion by the composer “In C” by Terry Riley and “Its gonna rain” by Steve Reich*

O autor faz menção aqui ao compositor Alvin Lucier (1931-) e sua peça *Crossings* (1984). Segundo o autor,

A longa forma de *Crossings* (1984) de Alvin Lucier é especialmente clara; ela surge processivamente, neste caso de um glissando de som senoidal puro que lentamente (mais de 16 minutos) ascende de infra-sons para a ultra-sons. Um processo discreto combina com este glissando, criando uma série de processos contínuos de curto alcance. Uma orquestra dividida alterna no jogo de alturas consecutivas, em uma escala cromática ascendente ascendente que interage com o glissando. Um grupo inicia e sustenta um intervalo acima daquela do glissando de tal modo que a uma dissonância mútua cria batimentos; conforme o glissando aumenta, a velocidade do batimento diminui; quando se alcança um uníssono com os instrumentos, os batimentos momentaneamente cessam; como o glissando ascende acima da nota dos instrumentos, batimentos gradualmente aceleram. Eventualmente, o outro conjunto entra e sustenta um intervalo de um semitom acima, começando esse processo de curto alcance novamente. (MAILMAN, 2013, p. 128)¹⁰⁷

Essa descrição é um breve comentário da peça, mas descreve um simples processo de ação musical, bem como relata de forma clara os resultados sonoros. É possível presumir que existe, na execução de *Crossings*, um simples algoritmo que permeia toda a peça: uma alternância de um ritmo lento de dois grupos instrumentais defasados temporalmente em relação ao som emitido pelo alto-falante¹⁰⁸.

Uma vez que na MP também é possível notar a influência de um algoritmo, é necessário evitar confusão com o uso original da MP, tal qual em compositores como Reich, Lucier e Riley; já o seu uso apropriado é realizado por Adrian Ward, Julian Rohruber, Frederick Olofsson, Alex McLean, Dave Griffiths, Nick Collins e Amy Alexander. Sugiro portanto comparar o fragmento do manifesto citado na página 53 com um fragmento do manifesto *Music as a Gradual Process* do compositor Steve Reich (1968):

O que é distintivo sobre processos musicais é que eles determinam todos detalhes de nota-a-nota (som-a-som) e toda a forma simultaneamente (pense em um *Round* ou um *Canon*). **Eu estou interessado em processos perceptivos. Eu quero ser capaz de ouvir o processo acontecendo através da música que soa.** Para facilitar a descri-

¹⁰⁷ Tradução nossa de *The long-range form of Alvin Lucier's Crossings (1984) is especially clear; it arises processively, in this case from a glissando of a pure sine tone that slowly (over sixteen minutes) ascends from infrasonic to ultrasonic. A discrete process combines with this glissando, creating a series of short-range continuous processes. A divided orchestra alternates in playing the consecutive pitches of a rising chromatic scale that interacts with the glissando. One ensemble initiates and sustains a pitch just higher than that of the glissando such that their mutual dissonance creates beats; as the glissando rises, the speed of the beats slows; when it reaches a unison with the instruments, the beats momentarily cease; as the glissando ascends above the pitch of the instruments, the beats gradually accelerate. Eventually, the other ensemble enters and sustains a pitch a semitone higher, starting this short-range process again.*

¹⁰⁸ Esse tipo de algoritmo pode ser facilmente implementado em uma linguagem de programação musical, como o SuperCollider (ver [seção A.3](#)).

ção detalhada de um processo musical, a escuta deve acontecer muito gradualmente. (REICH, 1968, p. 1).¹⁰⁹

Quanto ao manifesto de Adrian Ward, Julian Rohruber, Frederick Olofsson, Alex McLean, Dave Griffiths, Nick Collins e Amy Alexande, tenho uma primeira impressão que essa transformação gradual do som não está presente da mesma forma que em Reich, Lucier ou Rilley; vejo que o uso a reminiscência do termo “processo” a partir de “procedural” e “processador” se dá mais pela atividade de reprogramação em uma CGS ou CGC, mais especificamente, para fins de entretenimento. Aqui o *processo* é formalizado como algoritmo, colocado na situação de constante modificação.

Já no manifesto de Reich o som não é programado, mas se prevê um comportamento musical em constante modificação a partir de uma ação mínima sobre aquilo que produz o som. Um exemplo interessante é seu *Pendulum Music* (1968), onde deixa oscilar microfones posicionados acima de alto-falantes, com os cabos presos no teto; através de um único tipo de ação, restrita apenas pelas distâncias tomadas por intérpretes antes de colocar os microfones em movimento e pela comprimento do cabo preso ao teto¹¹⁰, ocorrerá um *processo físico* (desaceleração das oscilações que influenciam quando o microfone passará pelo alto-falante) que desencadeará um *processo perceptivo* (escuta de diferentes momentos de quando estes microfones passam pelo alto-falante até um momento estacionário).

Tal técnica faz uso deliberado de um fenômeno conhecido como microfonia, um *processo recursivo* onde os sons projetados pelo alto-falante são retro-alimentados pelo microfone. A retroalimentação já era usada por Pauline Oliveros (1969), como por exemplo em *Beaultiful Soup* (1967), peça para fita de dois canais, elaborada através de um circuito em retroalimentação (a compositora indica uma atividade musical econômica que ao longo do tempo causa "interessantes mudanças timbrísticas em sons sustentados"¹¹¹). Uma proposta posterior também foi realizada por Alvin Lucier (1931-) em *I'm sitting in a room* (1969) quando retroalimenta seu próprio discurso em uma sala, incrementando sinais ao gravador, criando um *drone*, ou um som de duração indefinida¹¹². Esta técnica de retroalimentação também pode ser utilizada no *live coding*, por exemplo, no SuperCollider (ver [seção A.2](#)).

¹⁰⁹ Tradução nossa de: *The distinctive thing about musical processes is that they determine all the note-to-note (sound-to-sound) details and the over all form simultaneously. (Think of a round or infinite canon.) I am interested in perceptible processes. I want to be able to hear the process happening throughout the sounding music. To facilitate closely detailed listening a musical process should happen extremely gradually.*

¹¹⁰ Seria possível incluir aqui também a resistência do ar ao movimento do objeto.

¹¹¹ Cf. OLIVEROS, 1969, p. 43 Tradução nossa de: *interesting timbre changes on sustained sounds*

¹¹² É interessante comentar, nesse contexto, o uso deliberado de *drones* no *livecoding* através de uma apropriação da *Drone Music*.

1.4.2.2 Música de algoritmos complexos

Esta seção discute o discurso da MG a partir de dois pontos: da perspectiva de Brian Eno, próximo à MP, e da perspectiva estrutural, próxima das teorias de gramáticas gerativas. A primeira será abordada na [subseção 1.4.2.2.1](#) e a segunda na [subseção 1.4.2.2.2](#).

1.4.2.2.1 Abordagens procedurais

Como a estética de Brian Eno foi mencionada por [Ward et al. \(2004\)](#), farei um comentário a respeito, a partir de um fragmento de texto de [Eno \(1978\)](#):

O conceito de música projetada especificamente como pano-de-fundo ambiental foi um pioneirismo da indústria Muzak nos anos cinquenta, e tem sido conhecido genericamente como Muzak. As conotações que este termo carrega são particularmente associadas com um tipo de material que as indústria Muzak produz - melodias familiares arranjadas e orquestradas de maneira leve e derivativa. Compreensivelmente, isso leva ouvintes atentos (e compositores) a dispensar inteiramente o conceito de música ambiental como digna de atenção (...) Se as empresas existentes de música enlatada se baseiam em regularizar ambientes cobrindo as idiossincrasias acústicas e atmosféricas, a Música AMbiental intenciona em melhorá-las. **Se o pano-de-fundo convencional é produzido removendo todo o sentido de dúvida e incerteza (e portanto o interesse genuíno) da música, a Música Ambiental retém essas qualidades (ENO, 1978, online)**¹¹³

É interessante que neste contexto podemos fazer referência à *Discreet Music* (1975) e *Ambient 1: Music for Airports* (1978); tais peças fazem uso deliberado de laços de fitas que criam sistemas de atrasos (*delays*). Se tais procedimento criativos não eram novos na época¹¹⁴; geralmente essas abordagens podem criar *drones* que ficam variando em seu espectro.

Um exemplo interessante de *live coding* com *drones*, que utiliza pequenos impulsos sonoros no SuperCollider (ver [Figura 24](#)). Aqui o processo de escuta é contínuo, semelhante à práticas da música eletroacústica, onde algoritmos são pré-concebidos e modificados durante a performance. Cole Ingraham utiliza um objeto que gera impulsos sonoros aperiódicos, e o mantém como um plano sonoro que se mantém, e vai sendo. Paralelamente,

¹¹³ *The concept of music designed specifically as a background feature in the environment was pioneered by Muzak Inc. in the fifties, and has since come to be known generically by the term Muzak. The connotations that this term carries are those particularly associated with the kind of material that Muzak Inc. produces - familiar tunes arranged and orchestrated in a lightweight and derivative manner. Understandably, this has led most discerning listeners (and most composers) to dismiss entirely the concept of environmental music as an idea worthy of attention. (...) Whereas the extant canned music companies proceed from the basis of regularizing environments by blanketing their acoustic and atmospheric idiosyncrasies, Ambient Music is intended to enhance these. Whereas conventional background music is produced by stripping away all sense of doubt and uncertainty (and thus all genuine interest) from the music, Ambient Music retains these qualities.* Grifo nosso.

¹¹⁴ Nesse sentido indicamos as peças de Steve Reich e Alvin Lucier e Pauline Oliveros já citadas.

```

7 sig = Dust.ar(
8 Xline.kr(0.1, 100, dur)).lag(LFNoise0.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.0001, 0.01)
9 );
10
11 sig = RPF.ar(
12 sig,
13 LFOisel.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(100, 10000).clip(50, 20000),
14 LFNoise0.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.1, 1)
15 );
16
17 sig = GVerb.ar(sig) [0];
18
19 Out.ar(out, [sig, DelayC.ar(sig, LFNoise0.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.01, 0.1))]*env)
20 ).add;
21 )
22
23 10.do {Synth("bg", [{"out": 0, "dur": 600})];
24
25 ///////////////////////////////////////////////////////////////////
26
27 /
28 SynthDef("one", {|out=0, dur=1, freq=440, amp=1, pan=0|
29 var sig, env;
30
31 env = EnvGen.kr(Env.now([0, 1, 0], [0, 30, 0]), 'linear', doneAction:0);
32
33 sig = SinOsc.ar(
34 freq,
35 0,
36 LFNoise2.kr(LFNoise0.kr(Rand(0.1, 1)).range(0.1, 1)).range(0.1, 0.0) +
37 SinOsc.kr(Rand(0.1, 0.0));
38 );
39
40 Out.ar(out, Pan2.ar(sig, pan, env*amp))
41 }).add;
42 )
43
44 (5*[14/9, 16/9]).do{|f|Synth("one", [{"freq": f, "pan": 0, "xrand(-1), "amp": xrand(0.1, 0.1)}]);
45
46 ///////////////////////////////////////////////////////////////////
47 }
48
49 
```

Figura 24 – Improviso com *drones* no SuperCollider **Fonte:** <<https://www.youtube.com/watch?v=b8j4umQ2lIE>>.

sons contínuos vai permeando este plano dos impulsos, a partir de senóides com um vibrato. É interessante notar que em alguns momentos, o *live coder* deixa de programar para prestar atenção no resultado sonoro obtido, ao invés de ficar constantemente re-programando o código-fonte. **há uma bela citação de alvin lucier a respeito**

1.4.2.2.2 Abordagens estruturais

Músicas realizadas a partir do **Item i**, na [subseção 1.4.2](#), página 61, podem ser relacionadas com pesquisas de gramática generativa, inaugurada com o artigo “Three Models for the description of language” de Noam Chomsky (1956). Nesta disciplina linguística/estrutural prevalecem as implementações computacionais analíticas que possibilitam usos criativos a partir de um *corpus teórico* já estabelecido¹¹⁵.

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

¹¹⁵ Sobre este tema sugerimos a leitura da tese "Luteria de Algoritmos Pós-Tonais" de Soares (2015-03-13), defendida recentemente nesta mesma instituição onde esta pesquisa está sendo realizada. Também elaborei uma peça eletroacústica nomeada *Impulsos* (<<http://soundcloud.com/opusd/musica-gerativa-9-impulsos>>), onde a partir do primeiro modelo de Chomsky (cadeia Markoviana de ordem 0) e de técnicas de retroalimentação de sinais descritas por Oliveros (1969) e Alvin Lucier, elaborei uma *árvore-drone* a partir de pulsos-finitos, feitos no aplicativo *Wavepot* (<<http://www.wavepot.com>>), semelhantes ao som de um metrônomo.

para cima” me refiro a processos elementares de uma composição, que influencia na sua macro-forma.

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

Uma abordagem estritamente de cima-para-baixo considera a macroestrutura como um plano global pré-concebido ou um modelo cujos detalhes são preenchidos em fases posteriores da composição. Isto corresponde à noção tradicional de forma na música clássica, onde certos esquemas formais tem sido usados por compositores como moldes (Apel 1972). Muitos compositores predeterminam a macroestrutura de suas peças de acordo com um esquema mais-ou-menos formal antes de um único som ser composto. Por contraste, uma abordagem estritamente de baixo-para-cima concebe a forma como resultado de um processo de desenvolvimento interno provocado por interações nos níveis mais baixos da estrutura (...) Para alguns, composição envolve uma mediação entre as abordagens de cima-para-baixo e de baixo-para-cima, entre uma concepção abstrata de alto-nível e os materiais concretos sendo desenvolvidos nos níveis mais baixos da estrutura. Isso implica uma negociação entre o desejo de ordenar a macroestrutura e imperativos que emergem da fonte material (...) Ultimamente, a dicotomia entre forma e processo é uma ilusão, uma falha da linguagem para cegar dois aspectos do mesmo conceito em uma unidade. Nas ciências da computação, o conceito de *restrição* elimina esta dicotomia (Sussman and Steele 1981). Uma forma é construída de acordo com um conjunto de relações. Um conjunto de relações implica um processo de avaliação que resulta em uma forma. ([ROADS, 2001](#), p. 12-14) ¹¹⁶

Restrições são pequenas regras, que se assemelham a gramáticas; tais gramáticas, podem depender métodos e técnicas de composição que restringem possibilidades criativas, afim de gerar novas restrições e possibilidades. Esta categoria de composições dependem de um corpo teórico bem estabelecido, procedendo primeiramente através da análise; tais análises podem partir como a Teoria de níveis estruturais de Heinrich

¹¹⁶ Tradução nossa de: *A strict top-down approach considers macrostructure as a preconceived global plan or template whose details are filled in by later stages of composition. This corresponds to the traditional notion of form in classical music, wherein certain formal schemes have been used by composers as molds (Apel 1972). (...) Many composers predetermine the macrostructure of their pieces according to a more-or-less formal scheme before a single sound is composed. By contrast, a strict bottom-up approach conceives of form as the result of a process of internal development provoked by interactions on lower levels of musical structure (...) For some, composition involves a mediation between the top-down and bottom-up approaches, between an abstract high-level conception and the concrete materials being developed on lower levels of musical structure. This implies negotiation between a desire for orderly macrostructure and imperatives that emerge from source material (...) Ultimately, the dichotomy between form and process is an illusion, a failure of language to bind two aspects of the same concept into a unit. In computer science, the concept of constraints does away with this dichotomy (Sussman and Steele 1981). A form is constructed according to a set of relationships. A set of relationships implies a process of evaluation that results in form*

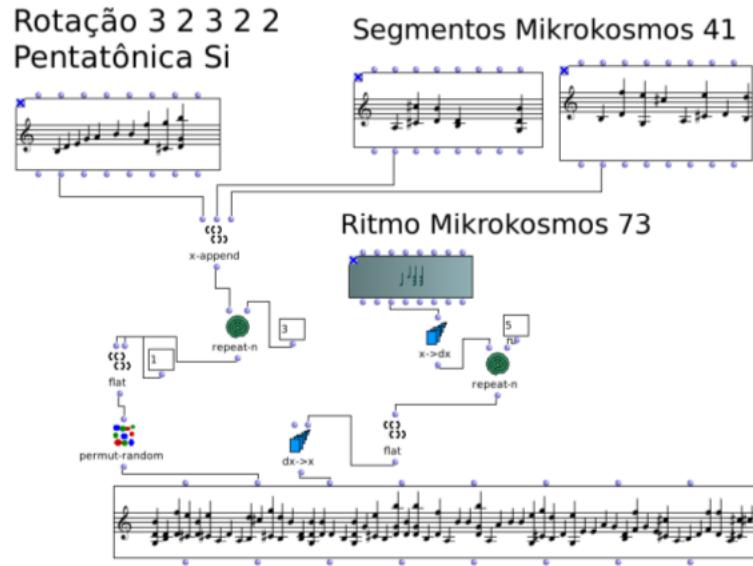


Figura 25 – Processo analítico da gramática musical dos *Mikrokosmos* 41 e 73 de Bartók, e recombinação dos seus elementos melódicos e rítmicos para elaboração de um novo material musical, como um processo próprio da CAC. **Fonte:** ([SOARES, 2015-03-13](#), p. 113)

Schenker, da Teoria Generativa da Música Tonal de Lerdhal e Jackendoff (GTTM) e a Teoria de Implicação-Realização de Eugene Narmour, para depois agir criativamente; tais análises indicariam comportamentos direcionais, melódicos, harmônicos e rítmicos de uma composição, colocando este aspecto como relativo a algumas produções da música tonal europeia ¹¹⁷.

Um exemplo interessante, fora do contexto do *live coding*, mas próprio da abordagem acima, foi realizado por [Soares \(2015-03-13\)](#), ao mapear clichês de ritmos (entendidos aqui como padrões musicais, no âmbito do ritmo, um elemento caro à GTTM) no *Mikrokosmos* 25 de Béla Bartók (1881-1945), e transpô-los para um aplicativo em OpenMusic, baseado na linguagem LISP (ver Figura 25), gerando assim novos materiais musicais que possuem uma memória bartokiana.

No *live coding*, essa abordagem pode ser realizada com o *Tidal* de [McLean e Wiggins \(2010\)](#)¹¹⁸. Em alguns vídeos observados, como na Figura 26, observamos o seguinte uso: de 0'05" a 0'22", Mark Derricutt define um pulso básico como um bumbo; de 0'22" a 0'38", define um outro som, semelhante ao bumbo, mas com mais graves no ataque que o primeiro, em contra-tempo com o pulso básico (o que por si só caracteriza uma performance de

¹¹⁷ Cf. [ROWE, 1991](#), p. 29–43

¹¹⁸ “*Tidal* é uma lingagem de padrões embebida na linguagem de programação Haskell, consistindo de representações de padrões, uma biblioteca de geradores de padrões” ([McLean; WIGGINS, 2010](#), p. 2). Tradução nossa de *Tidal is a pattern language embedded in the Haskell programming language, consisting of pattern representation, a library of pattern generators and combinators, an event scheduler and programmer's live coding interface*.

```

File Edit Options Buffers Tools Haskell-Tidal Haskell Help
cps (30/50)
d1 $ sound "[bd sn bd bd, ht sn ht cp sn cp]"
d2 $ sound "gabba industrial" |+| vowel "a e"
d3 @ every 2 (slow 16) $ sound "arp:0 arpy:3 arpy:1 arpy:2" |+| pan sinewave1

-UUU:---F1 test.tidal All L6 (Haskell Tidal) ---
haskell-mode-hook is a variable defined in 'haskell-mode.el'.
Its value is nil

  This variable may be risky if used as a file-local variable.

Documentation:
Hook run after entering Haskell mode.
No problems result if this variable is not bound.
'add-hook' automatically binds it. (This is true for all hook variables.)

You can customize this variable.

[back]

-UUU:---F1 *Help* All L1 (Help) ---

```

Figura 26 – *Live coding* com Tidal **Fonte:** <<https://www.youtube.com/watch?v=ZQqg9Ghxruo>>.

??

algorave). De 0'38" a 1'05", define quatro semicolcheias. De 1'05" a 1'16", executa uma permutação com esses materiais. De 1'16" a 1'26", o ritmo se torna mais complexo, mas com base no bumbo inicial. De 1'26" a 1'51", é alterado o pulso original, tornando o ritmo anterior mais lento. De 1'51" a 2'02", entra um “baixo”, em contra-tempo com o 3º tempo da célula rítmica; do ponto de vista de gênero musical, o nome do som utilizado para o baixo é o “gabba”, um tipo de “Hardcore dance” surgido em 1993 na Holanda, e também conhecido como *Rotterdam techno*¹¹⁹. A partir de 2'02" a 3'25" o *live coder* improvisa com um material chamado *industrial* (outro possível gênero musical para a performance), manipulando-o através de DSPs que se assemelham a filtros de vogais (aeiou). De 3'25" a 4'25" um novo material melódico, sincopado, é incluído (*arp*, provavelmente semelhante aos sintetizadores ARP); é interessante notar que as alturas dessa melodia são permutadas. De 4'25" a 4'57", essa melodia é panoramizada, isto é, uma onda senoidal controla a posição estereofônica da melodia, como se ela girasse em torno da cabeça do ouvinte. Em 4'57" a 5'22" é aplicado um *delay* nesta melodia, isto é, as notas serão repetidas, criando uma espécie de *canon*. Em 5'22" o mesmo procedimento de panoramização é aplicado ao baixo. Em 5'58" a 6'26", a melodia em *canon* é alterada para tocar com pausas entre algumas notas, sem o *delay* que gerava o *canon* que se mantém até o final em 7'04"

¹¹⁹ Conferir <<http://corehistory.blogspot.com.br/2009/12/gabber.html>>.

1.4.3 Entre a música de vanguarda e a música de pista

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

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

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

Poderia ser questionado se o intercruzamento de gêneros musicais no *livecoding* depende dos *softwares*. Iazzetta responderia não, mas que dependem de uma articulação entre produções musicais e seus compositores, que carregam diferentes formações teóricas. Esse problema é colocada da seguinte forma:

Embora seja possível vislumbrar que a maioria dos programas disponíveis no mercado possam ser agrupados num número razoavelmente pequeno de categorias, é certo também que a disseminação destes nas

duas últimas décadas do século XX veio acompanhada de uma proliferação de microteorias, pequenas gramáticas musicais que passam a circular entre os músicos. (...) É notável como há um alargamento no número de segmentos musicais distintos que passam a se utilizar da mesma tecnologia – e, portanto, em algum nível, das mesmas teorias e traços estéticos – para realizar produções tão distantes (...) **A possibilidade de ocorrência desse cruzamento entre ferramentas tecnológicas e gêneros estéticos acontece mais por uma sinergia entre produções, que em sua diversidade compartilham dos mesmos elementos sociais e culturais, do que por uma intenção explícita dos artistas envolvidos no sentido de promover uma aproximação** (IAZZETTA, 2009, p. 152)

Embora seja possível considerar uma "comunidade Max/MSP", ela está dentro de uma população contendo as comunidades "SuperCollider", "PureData", "ChucK", indicando apenas alguns.

Dentro dessa população, surgem as pequenas comunidades de *softwares* de *live coding*, pela utilização e invenção de mini-linguagens pouco usadas, se comparadas com as do parágrafo acima. No contexto anglófono, essas pequenas comunidades apropriam um termo, segundo Collins e McLean (2014), *algorave*.

Algorave é um tipo de música eletrônica de pista, que se utiliza de algoritmos, processos, e teorias generativas; não se configura como uma música de pista normal, em seu processo de criação, mas reproduz alguns regimes de escuta, como *dance*, *drum'n'bass*, *cyberpunk*, etc. Nesse processo de criação, é comum utilizar alguns *softwares* já comentados, como o *iXiLang* e o *Tidal*, o que leva a crer na emancipação de “comunidades iXiLang” e “comunidades Tidal”.

No entanto, *algorave* é um termo anterior ao advento do *live coding* e ao uso dos *softwares* supracitados:

Algorave não é sustentado exclusivamente por *live coders*, mas estes têm mantido uma forte presença em todos os eventos até agora. É assim talvez, porque a tradição do *live coding* de projetar telas motiva todo o esforço; onde algoritmos não estão visíveis por períodos de tempo durante uma algorave, se corre o risco das coisas parecerem muito como um evento de música eletrônica padrão. (COLLINS; McLean, 2014, p. 356) ¹²⁰

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

¹²⁰ Tradução nossa de *Algorave is not exclusively a preserve of live coders, but they have maintained a strong presence at every event thus far. This is perhaps because the live coding tradition of projecting screens help motivates the whole endeavour; where algorithms are not made visible for periods during an algorave, we run the risk of things feeling much like a standard electronic music event.*

techno, com abordagem *gabba*; é interessante aqui o uso do termo *club live coding*. Em 2001 é identificado a utilização de redes neurais para composição de padrões semelhantes ao *drum'n'bass*. Em 2004 é fundado o TOPLAP, organização internacional de *live coding*, em uma casa noturna de Hamburgo.¹²¹

¹²¹ [COLLINS; McLean, 2014](#), loc. cit..

Conclusão

A pesquisa tomou um rumo diferente da proposta inicial, : um estudo que discutiria a questão da aparente intimidação experenciada por compositores no uso de linguagens de programação para composição musical, especificamente em ambientes de redes de computadores, bem como a elaboração de um *software* original com base nessa reflexão.

Para chegar neste aplicativo, tomei conhecimento de uma cena emergente no que é conhecido hoje como *live coding* em navegadores de internet¹²², tais como *Gibber* (ROBERTS; KUCHERA-MORIN, 2012), *Vivace* (VIEIRA et al., 2015), *Wavepot*¹²³, *Html5Bytebeat*¹²⁴. Uma reflexão a respeito das linguagens nestes aplicativos possibilitou o desenvolvimento de um aplicativo *web* em conjunto com Luiz Schiavonni, professor da UFSJ (Universidade Federal de São João del-Rei), que foi chamado de *Termpot*¹²⁵.

No entanto, o Programa de Pós-graduação em Artes, Cultura e Linguagens (PPG-ACL/UFJF) proporcionou o contato com obras de autores como Kuhn (1970), Feyerabend (1975) e Santos (2008) que foram significativas para aprimorar a metodologia; arrisquei-me a uma pesquisa orientada a paradigmas, aqui, paradigmas do *livecoding*, considerando a existência cooperativa/competitiva de um conjunto deles em ambientes de produção de conhecimento (incluímos aqui a Universidade). Adicionalmente, a vivência com uma botânica dedicada em problemas de plano de manejo de uma espécie vegetal em Ubatuba/SP e um comunicador debruçado em problemas de gênero em Juiz de Fora/JF, me colocou na posição de refletir sobre uma questão, pessoalmente mais fundamental que a proposta inicial, de uma ecologia de gêneros musicais utilizando um fragmento de produções do *livecoding*¹²⁶. Autores citados acima não foram utilizados neste trabalho para discutir o *livecoding*, mas foram importantes para a concepção do seguinte triangulamento metodológico realizado: levantamento qualitativo do *livecoding*, a partir de uma bibliografia básica disponível¹²⁷ e verificação quantitativa, através de levantamento de dados seletivo na internet que confirmasse o levantamento qualitativo em um nicho específico¹²⁸.

A partir de diferentes textos, derivei palavras-chave que carregam significados do objeto de pesquisa: *live coding*, *live-coding*, *livecoding*, *live code*, *conversational programming*,

¹²² Apple Safari, Google Chrome, Mozilla Firefox.

¹²³ <<http://www.wavepot.com>>, acessado em 27 de junho de 2015

¹²⁴ <<https://github.com/greggman/html5bytebeat>>, acessado em 27 de junho de 2015.

¹²⁵ <<http://jahpd.github.io/termpot>>, acessado em 27 de junho de 2015.

¹²⁶ Em comunicação pessoal com Tiago Rubini, “a primeira menção à palavra ‘gênero’ se refere à questão de identidade de gênero. A segunda, sobre dinâmicas de gêneros musicais”. Infiro que uma dinâmica de gêneros musicais pressupõe um conjunto limitado de conhecimentos, que exige ser discutido para oferecer uma idéia das próprias limitações do conhecimento discutido.

¹²⁷ <http://toplak.org/wiki/Videos,_Articles_and_Papers>. Acessado em 27 de junho de 2015.

¹²⁸ Utilizando um SDK disponibilizado pelo Soundcloud.

on-the-fly programming, live algorithm programming; estas palavras-chaves apontam para uma prática de improvisação utilizando o computador, articulada de maneira colaborativa ou competitiva entre *performers*, mediada por projetores visuais e acústicos (projetores e alto-falantes), e direcionadas para um público parcialmente passivo em ato de ouvir música e ver imagens (codificadas durante a improvisação). Este aspecto de improvisação com *scripts* criados e editados no computador, segundo Cox, McLean e Ward (2004), reside em uma capacidade de predizer (de forma aproximada) resultados complexos antes de codificar, isto é, o executante deve ser capaz de realizar em um curto espaço de tempo uma formalização lógica de um comportamento audiovisual antes de sua programação em um *script*:

Um programador é, portanto, capaz de predizer e especular sobre como o seu código irá se comportar em circunstâncias mais usuais. Como com qualquer coisa que é de autoria, a questão da subjetividade é inevitável, uma vez que qualquer resultado particular pode ser alcançado em diferentes (e muitas vezes concorrentes) maneiras. Nesse sentido, qualquer senso de improvisação depende de um entendimento preditivo de sistemas complexos e geradores. (COX; McLean; WARD, 2004, p. 169)¹²⁹

Colocado de outra forma por Ruthmann et al. (2010)

Executar de maneira efetiva uma manipulação em tempo-real do código (live coding musical) para criar e formalizar música gerada requer ambos entendimentos musicais e computacionais. De uma perspectiva musical, é necessário entender o fluxo de como uma música generativa deve soar. Da perspectiva computacional, é preciso entender como o código deve ser ajustado e manipulado em tempo real para atingir mudanças aurais e musicais desejadas. (RUTHMANN et al., 2010, p. 3-4).¹³⁰

Esses conhecimentos pré-concebidos antes de codificar podem ter origem no conhecimento de mundo da Música daquele que realiza a improvisação; de diferentes maneiras, músicos-programadores adequam programação-partituras para determinados contextos do fazer musical. Expressa-se uma teoria musical que regula os algoritmos, esta por sua vez regulada por uma ecologia de gostos musicais adequadas para cada contexto. Interessado na dinâmica dos gêneros musicais que emergem de diferentes contextos de improvisação, coloquei-me a tarefa de discutir o que diferentes autores debruçados no assunto *livecoding* entendem por música, como realizam, e seus discursos da respeito de realizações em público.

¹²⁹ Tradução nossa de: *A programmer is therefore able to predict and speculate upon how their code will behave in most usual circumstances. As with anything that is authored, the issue of subjectivity is unavoidable, since any particular result can be achieved in different (and often competing) ways. In this, any sense of improvisation relies on a predictive understanding of complex and generative systems.*

¹³⁰ Tradução nossa de *Performing effective real-time manipulation of code (musical live coding) to create and shape generated music requires both musical and computational understanding. From a musical perspective, one needs to understand how the ongoing, generative music should sound. From a computational perspective, one needs to understand how the code can be adjusted and manipulated in real time to achieve the aural and musical changes and outcomes one desires.*

Decidi recortar o assunto a partir de comparações limitadas ao âmbito musical por julgar-me incapaz de abordar diferentes linguagens artísticas como a literatura generativa e práticas derivadas de exibição simultânea da imagem renderizada e seu respectivo código; algumas menções foram feitas apenas para delimitar o que está fora e o que está dentro de uma janela de discussão. Uma bibliografia básica foi levantada como forma de esclarecer origens e práticas musicais do *livecoding*: Cox, McLean e Ward (2000), Cox, McLean e Ward (2004), Collins et al. (2003a), McLean (2004), Wang e Cook (2004), Ward et al. (2004), Collins (2007), Rohrhuber e Campo (2009), Ruthmann et al. (2010), McLean et al. (2010), Magnusson (2011), Magnusson (2014a), Magnusson (2014b) .

O *livecoding* pode existir em contextos tradicionalmente reservados para concertos ou em contextos informais que estimulam a sociabilização através da dança (os *Night Clubs* de McLean (2004)); noto que, em um período de aproximadamente dez anos, ocorre um movimento de cooperação entre espaços acadêmicos e de entretenimento, onde emerge o termo *algorave*¹³¹. Por outro lado, estes mesmos espaços estimularam o desenvolvimento de novas áreas de pesquisa, entre elas, o *livecoding* de ambientes virtuais, isto é, a prática sendo aplicada em pequenas ou grandes redes de computadores; ao mesmo tempo, o advento da biblioteca *WebAudio API* possibilitou a experimentação do *livecoding* na internet¹³².

Destes dois modos busquei fazer um mapeamento de práticas musicais mencionadas por autores de manifestos e artigos mencionados no segundo parágrafo desta conclusão, bem como a localização geográfica e que práticas musicais são colocadas no plano de discussão em relação ao *livecoding*: 1) Música Algorítmica (MA), 2) Música Processual (MP), 3) Música Generativa (MG), 4) e Música de Pista (ou o que que denominamos a partir do termo simplificado *Disk Jockey*, (DJ) . Busquei averiguar até que ponto o uso dos termos referidos estão de acordo com definições compartilhadas, respaldados em um embate com autores como Reich (1968), Eno (1978), Kramer et al. (1999), Roads (2001), Wooler et al. (2005) Malt (2006), Walker e Kramer (2006), Essl (2007), Cope (2008), Iazzetta (2009), Mailman (2013) e Collins e McLean (2014) Castelões, Oliveira e Franco (2015).

Em um segundo momento, levantamos dados pertinentes ao tema na rede social *Soundcloud* como maneira de justificar o primeiro mapeamento das comunidades de gosto.

Realizando uma retrospectiva geopolítica dos autores, isto é, observando as lo-

¹³¹ “Nenhuma conferência acadêmica está completa sem uma *algorave*, uma chance de dançar algoritmos com velhos ou novos amigos. Teremos pelo menos um clube de noite, no excelente co-op Wharf Chambers [<http://www.wharfchambers.org/>]. em <<http://iclc.livedcodenetwork.org/cfp.html#performance>>. Tradução nossa de: *No academic conference is complete without an algorave, a chance to dance to algorithms with friends new and old. We will have at least one club night, at the excellent Wharf Chambers co-op. More details to follow.*

¹³² Como os aplicativos *Gibber*, *Wavepot* e *Html5Bytebeat*, *Vivace*. Para mais informações a respeito, sugiro a leitura de "The Web Browser As Synthesizer And Interface" de Roberts, Wakefield e Wright (2013) e "The Viability of the Web Browser as a Computer Music Platform" Wyse e Subramanian (2014).

calizações do globo terrestre em que foram escritos os textos supracitados no primeiro parágrafo deste capítulo, outros em *passim*, mais um conjunto de dados que podem ser observados no [Apêndice D](#) – que descrevem essas regiões de maneira quantitativa em um período de tempo entre 2008 e 2015–, é possível confirmar uma tese que foi sendo construída no decorrer da pesquisa, mas que somente pode ser verbalizada no segundo parágrafo de “A genealogia da moral” de Nietzsche, “o privilégio senhorial de dar nomes permitem-nos conceber a origem da linguagem ela mesma como uma manifestação de poder dos governantes”¹³³, no sentido de que o *live coding* possue esse nome, como *programa de pesquisa*, que integra todo uma Epistemologia do Norte que que [Santos](#) tanto comenta.

Na Inglaterra identifico os autores como Alex McLean, Nick Collins, Adrian Ward, e Dave Grifths. Nos EUA identifico as *Laptop Orchestras*, em Stanford e Princeton, bem como compositores como James Harkins e Joshua Parmenter e Ge Wang; na Austrália, tem sido notável o papel de Andrew Sorensen na *Queensland University of Technology* utilizando o piano; no Brasil identifico os trabalhos de Bernardo Barros, André Damião, Antônio Goulart, Vilson Vieira, Geraldo Magela de Castro Rocha Junior, Caleb Mascarenhas Luporini, Daniel Penalva, Ricardo Fabbri, Renato Fabbri, Ricardo Brasileiro e Daniel Penalva e Flávio Luiz Schiavonni.

No levantamento de dados do Soundcloud pude confirmar alguns dos pontos levantados no levantamento bibliográfico e questionar algumas afirmações feitas: por exemplo, na questão de qual país produz grande quantidade de livecoding confirmei posições de países falantes da língua inglesa, como Inglaterra e EUA, mas ao mesmo tempo, notei uma grande quantidade de produções anônimas, isto é, com localizações não identificadas, desacreditando em uma centralização de produção; ademais foi possível perceber uma miríade de produções em países como Alemanha, México e Japão. Na questão de gênero musical, articulado através de *hashtags*¹³⁴, pude confirmar uma hibridização de práticas musicais consideradas anteriormente distintas, e ao mesmo tempo, questionar uma centralização conceitual em um ou outro termo, como *a) algorithmic music b) algorave, c) algopop, d) bytebeat, e) drone f) electronic music, g) electroacoustic, h) glitch, i) noise, j) whistling*.

O seguinte ponto-de-vista foi desenvolvido: cada termo utilizado no *livecoding* carrega uma teoria musical e, por outro lado, costumes distintos do fazer musical no *livecoding* estão relacionados com gêneros musicais, a partir daquilo que discute comunidades de gosto; tais comunidades de gosto foram investigadas em um nicho musical virtual específico, a saber, a rede social Soundcloud.

¹³³ Cf. *On the Genealogy of Morality* Edited by Keith Ansell-Pearson. Translated by Carol Diethe. Cambridge. 2006. Tradução nossa do segundo parágrafo do primeiro ensaio *The seigneurial privilege of giving names even allow us to conceive of the origin of a language itself as a manifestation of the power of the rulers*.

¹³⁴ Para mais informações, sugiro a leitura de <<https://en.wikipedia.org/wiki/Hashtag>> e <[https://en.wikipedia.org/wiki/Tag_\(metadata\)](https://en.wikipedia.org/wiki/Tag_(metadata))>, acessados em 27 de junho de 2015.

Percebi o *livecoding* como um campo de estudos em transição entre Música e Ciências da Computação; esse campo tem sido estimulado em países considerados como centros de referência, por exemplo, Inglaterra e EUA. Esse estímulo tem sido acompanhado por colaborações interdisciplinares formalizadas (isto é, reconhecidas institucionalmente); não coincidentemente, isso foi pressuposto por [Mathews \(1963\)](#) no desenvolvimento da família MUSIC N como única maneira de avançar neste campo de estudo¹³⁵. No Brasil, no entanto, existem barreiras institucionais que impedem a colaboração interdisciplinar formal, porém estas mesmas barreiras forçam músicos interessados na área, a aprender a programar, mesmo com algumas barreiras do jargão técnico (muitas vezes intimidadoras); de maneira semelhante, cientistas (da computação e muitas vezes físicos) e não-acadêmicos brasileiros, com seu conhecimento de programação, exploraram comportamentos musicais em *softwares* como forma de ultrapassar uma barreira impostas pelo jargão técnico dos músicos.

¹³⁵ Lembro que o desenvolvimento de estudos musicais com CSIRAC (na Austrália), antes mesmo do advento do MUSIC N, tem como um dos fatores de seu fracasso, segundo [Nunzio \(2010\)](#), devido a uma falta de cooperação entre músicos e cientistas da computação.

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 na página 17.
- 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 5 vezes nas páginas 15, 16, 20, 22 e 23.
- 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 45.
- BROWN, C.; BISCHOF, J. *INDIGENOUS TO THE NET: Early Network Music Bands in the San Francisco Bay Area*. 2013. Disponível em: <<http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html>>. Citado 2 vezes nas páginas 20 e 23.
- BUDAPEST, K. *Dancecoding experiments 2011*. Vimeo, 2011. Disponível em: <<https://vimeo.com/30133155>>. Citado 2 vezes nas páginas 31 e 32.
- CASTELÓES, L. E.; OLIVEIRA, T. d.; FRANCO, Y. Conversores de parámetros del color a parámetros sonoros cuantificables usando los sistemas RGB, HSV y CMYK. v. 7, n. 14, 2015. Disponível em: <https://www.academia.edu/12472704/_Conversores_de_par%C3%A1metros_del_color_a_par%C3%A1metros_sonoros_cuantificables_usando_los_sistemas_RGB_HSV_y_CMYK_2015_2013_>. Citado 3 vezes nas páginas 54, 57 e 87.
- CHADABE, J. Remarks on computer music culture. v. 24, n. 4, p. 9–11, 2000. Disponível em: <<http://www.jstor.org/stable/3681550>>. Citado na página 79.
- CHOMSKY, N. Three models for the description of language. MIT, p. 12, 1956. Citado na página 63.
- COLLINS, N. Generative music and laptop performance. v. 22, n. 4, p. 67–79, 2003. Disponível em: <<https://wiki.brown.edu/confluence/download/attachments/74258672/Collins,+Generative+Music.pdf>>. Citado na página 15.
- COLLINS, N. Live coding practice. 2007. Disponível em: <<http://www.composerprogrammer.com/research/livecodingpractice.pdf>>. Citado 2 vezes nas páginas 15 e 87.
- COLLINS, N.; McLean, A. Algorave: Live performance of algorithmic electronic dance music. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. [S.l.: s.n.], 2014. p. 355–358. Citado 6 vezes nas páginas 15, 27, 29, 53, 54 e 87.
- COLLINS, N. et al. Live coding in laptop performance. v. 8, n. 3, p. 321–330, 2003. Citado 3 vezes nas páginas 15, 24 e 87.

COLLINS, N. et al. Live coding techniques for laptop performance. v. 8, n. 3, p. 321–330, 2003. Citado 2 vezes nas páginas 15 e 29.

COPE, E. D. Prefacio a OM composer's book vol. 2. In: *OM Composer's Book*. [S.l.]: Editions Delatour, 2008. v. 2, p. ix–xv. Citado 3 vezes nas páginas 54, 57 e 87.

COX, G.; McLean, A.; WARD, A. collaborative collection of artworks, research and experiments by artists and academics, *The Aesthetics of Generative Code*. 2000. Disponível em: <<http://generative.net/papers/aesthetics/>>. Citado 3 vezes nas páginas 3, 30 e 87.

COX, G.; McLean, A.; WARD, A. Coding praxis: reconsidering aesthetics of code. In: *Read_me: Software Art & Cultures*. Olga Goriunova and Alexei Shulgin, 2004, (Digital Aesthetics Research Centre, 2004). p. 161–174. Disponível em: <<http://art.runme.org/1107862958-4046-0/cox.pdf>>. Citado 4 vezes nas páginas 22, 30, 86 e 87.

DOORNBUSCH, P. Computer sound synthesis in 1951: The music of CSIRAC. v. 28, n. 1, p. 10–25, 2004. Citado na página 30.

ENO, B. Base de dados, *Music for Airports liner notes*. 1978. Disponível em: <http://music.hyperreal.org/artists/brian_eno/MFA-txt.html>. Citado 3 vezes nas páginas 54, 62 e 87.

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 59.

ESSL, K. Algorithmic composition. In: *The Cambridge Companion to Electronic Music* (Nick COLLINS, Julio D'ESCRIVAN, org.). [S.l.]: Cambridge University Press, 2007. p. 107–125. ISBN 978-0521688659. Citado 4 vezes nas páginas 54, 55, 56 e 87.

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

FEYERABEND, P. *Against Method*. Third edition published by verso 1993. [S.l.]: Verso, 1975. ISBN 0-86091-481-X. Citado 2 vezes nas páginas 22 e 85.

FUMAROLA, M. A. Editor's note in Music Computer Journal, Volume 21, Number 1, *Computer Music - The Next 20 Years*. 1997. Disponível em: <<http://www.jstor.org/stable/3681101>>. Citado 2 vezes nas páginas 79 e 80.

GRIFFITHS, D. Fluxus: Scheme livecoding. 2008. Disponível em: <<http://www.pawfal.org/dave/files/scheme-uk/scheme-uk-fluxus.pdf>>. Citado 3 vezes nas páginas 15, 25 e 31.

HAOULI, J. E.; MANNIS, J. A. Abertura para o silêncio. In: . [S.l.: s.n.], 2011. p. 8. Citado na página 48.

HEIKKILÄ, V.-M. Discovering novel computer music techniques by exploring the space of short computer programs. p. 8, 2011. Disponível em: <<http://arxiv.org/abs/1112.1368>>. Citado 3 vezes nas páginas 37, 38 e 113.

- IAZZETTA, F. A IMPORTÂNCIA DOS DEDOS PARA a MÚSICA FEITA NAS COXAS. p. 8, 2005. Disponível em: <http://www.anppom.com.br/anais/anaiscongresso_anppom_2005/sessao21/fernando_iazzeta.pdf>. Citado na página 76.
- IAZZETTA, F. *Música e mediação tecnológica*. [S.l.]: Ed. Perspectiva-Fapesp, 2009. ISBN 9.788527308724E12. Citado 7 vezes nas páginas 11, 50, 54, 56, 57, 67 e 87.
- KRAMER, G. et al. *Sonification Report: Status of the Field and Research Agenda*. 1999. Disponível em: <<http://www.icad.org/websiteV2.0/References/nsf.html>>. Citado 4 vezes nas páginas 54, 57, 58 e 87.
- KUHN, T. *The Structure of Scientific Revolutions*. INTERNATIONAL ENCYCLOPEDIA of UNIFIED SCIENCE. [S.l.]: University of Chicago Press, 1970. ISBN 0-226-45803-2. Citado 2 vezes nas páginas 22 e 85.
- LEROUX, P. The model of the model in VOI(DEX). In: *The OM Composer's Book*. [S.l.]: Delatour, 2008. v. 2, p. 149–164. Citado na página 56.
- MAGNUSSON, T. Algorithms as scores: Coding live music. v. 21, p. 19–23, 2011. Citado 3 vezes nas páginas 15, 46 e 87.
- MAGNUSSON, T. *iXiLang - a live coding language interpreter - version 4*. Github, 2012. Disponível em: <<https://www.github.com/thormagnusson/ixilang>>. Citado na página 40.
- MAGNUSSON, T. Herding cats: Observing live coding in the wild. v. 38, n. 1, p. 8–16, 2014. Citado 3 vezes nas páginas 15, 28 e 87.
- MAGNUSSON, T. Scoring with code: Composing with algorithmic notation. p. 268–275, 2014. Disponível em: <http://www.ixi-software.net/thor/Magnusson_OS2014.pdf>. Citado 3 vezes nas páginas 12, 15 e 87.
- 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 4 vezes nas páginas 54, 59, 60 e 87.
- MALT, M. Concepts et modèles, de l'imaginaire à l'écriture dans la composition assistée par ordinateur. In: *Musique, instruments, machines. Autour des musiques électroacoustiques*. [S.l.]: Bruno Bossis, Anne Veitl et Marc Battier, 2006, (Musique et nouvelles technologies, v. 2). p. 213–234. Citado 3 vezes nas páginas 54, 55 e 87.
- MATHEWS, M. V. The digital computer as a musical instrument. v. 142, n. 3592, p. 553–557, 1963. Disponível em: <<http://www.jstor.org/stable/1712380>>. Citado 2 vezes nas páginas 12 e 89.
- MATHEWS, M. V. et al. *The technology of computer music*. 2a, 1974. ed. [S.l.]: MIT press, 1969. ISBN 0 26213050 5. Citado na página 12.
- MATHEWS, M. V.; MOORE, F. GROOVE a program to compose, store, and edit functions of time. p. 7, 1970. Citado 3 vezes nas páginas 3, 12 e 24.

- McCallum, L.; SMITH, D. *Show Us Your Screens*. Vimeo, 2011. Disponível em: <<https://vimeo.com/20241649>>. Citado 2 vezes nas páginas 15 e 25.
- McLean, A. *Hacking Perl in Nightclubs*. 2004. Disponível em: <<http://www.perl.com/pub/2004/08/31/livecode.html>>. Citado na página 87.
- McLean, A. Music improvisation and creative systems. online, p. 6, 2006. Disponível em: <https://www.academia.edu/467101/Music_improvisation_and_creative_systems>. Citado 3 vezes nas páginas 20, 45 e 52.
- McLean, A. et al. *Visualisation of Live Code*. Alex McLean; making music with a text, 2010. Disponível em: <<http://yaxu.org/visualisation-of-live-code/>>. Citado 5 vezes nas páginas 15, 16, 20, 21 e 87.
- 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 5 vezes nas páginas 15, 20, 22, 23 e 24.
- McLean, A.; WIGGINS, G. *TIDAL – PATTERN LANGUAGE FOR LIVE CODING OF MUSIC*. [S.l.]: Centre for Cognition, Computation and Culture; Department of Computing Goldsmiths, University of London, 2010. Citado na página 65.
- 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. Disponível em: <http://www.tesonline.com/__PDF/37726/37726p.pdf>. Citado 4 vezes nas páginas 12, 30, 49 e 89.
- NUNZIO, A. D. *Musica informatica*. 2014. Disponível em: <<http://www.musicainformatica.org/>>. Citado na página 12.
- OLIVEROS, P. Tape delay techniques for electronic composers. In: *Software for people: collected writings 1963-80*. 1. ed. Smith Publications, 1969. p. 11. Disponível em: <http://monoskop.org/images/2/29/Oliveros_Pauline_Software_for_People_Collected_Writings_1963-80.pdf>. Citado 2 vezes nas páginas 61 e 63.
- PARK, T. H.; MATHEWS, M. An interview with max mathews. v. 33, n. 3, p. 9–22, 2009. Disponível em: <<http://www.jstor.org/stable/40301041>>. Citado 2 vezes nas páginas 12 e 79.
- PUCKETTE, M. *The Theory and Technique of Electronic Music*. v. december 30, 2006. [S.l.]: World Scientific Publishing Co. Pte.Ltd., 2006. Citado 2 vezes nas páginas 52 e 107.
- RANDALL, J. A report from princeton. p. 84–92, 1965. Disponível em: <<http://www.jstor.org/stable/832506>>. Citado na página 12.
- RAYMOND, E. S. Minilanguages, finding a notation that sings. In: *The Art of Unix Programming*. Eric Steven Raymond, 2003, (<http://www.faqs.org/faqs/>). Disponível em: <<http://www.faqs.org/docs/artu/minilanguageschapter.html>>. Citado na página 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 4 vezes nas páginas 54, 60, 61 e 87.

- ROADS, C. Times scales of music. In: *Microsound*. First MIT paperback edition 2004. [S.l.]: MIT press, 2001. p. 1–41. ISBN 978-0-262-18215-7. Citado 4 vezes nas páginas 54, 63, 64 e 87.
- ROADS, C.; MATHEWS, M. Interview with max mathews. v. 4, n. 4, p. 15–22, 1980. Disponível em: <<http://www.jstor.org/stable/3679463>>. Citado na página 12.
- ROBERTS, C.; KUCHERA-MORIN, J. *Gibber: live-coding audio in the browser*. [S.l.]: University of California at Santa Barbara: Media Arts & Technology Program, 2012. Citado 2 vezes nas páginas 32 e 85.
- ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013. Citado na página 87.
- ROHRHUBER, J.; CAMPO, A. d. Improvising formalization - conversational programming and live coding. In: *New Computational Paradigms for Computer Music*, Assayag, G. and Gerzso, A., eds., Delatour/IRCAM, 2009. p. 13. Disponível em: <http://albertodecampo.net/uploads/ImprovisingFormalisation_NewLangs_IRCAM_jrh_adc.pdf>. Citado 2 vezes nas páginas 15 e 87.
- ROWE, R. J. *Machine Listening and Composing: Making Sense of Music with Cooperating Real-Time Agents*. Media Arts and Sciences — MIT, 1991. Disponível em: <<http://r.duckduckgo.com/l/?kh=-1&uddg=http%3A%2F%2Fdspace.mit.edu%2Fbitstream%2Fhandle%2F1721.1%2F29083%2F32147968.pdf>>. Citado na página 65.
- RUTHMANN, A. et al. Teaching computational thinking through musical live coding in scratch. In: *SIGCSE'10*. [s.n.], 2010. p. 6. Disponível em: <<https://teaching.cs.uml.edu/~heines/academic/papers/2010sigcse/SoundThinking-SIGCSE-2010.pdf>>. Citado 2 vezes nas páginas 86 e 87.
- 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 6 vezes nas páginas 20, 21, 25, 46, 85 e 88.
- 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 34.
- 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 5 vezes nas páginas 4, 11, 63, 65 e 99.
- SORENSEN, A.; SWIFT, B.; RIDDEL, A. Livecoding's many meanings. v. 38, n. 1, p. 65–76, 2014. Disponível em: <http://muse.jhu.edu/journals/computer_music_journal/v038/38.1.sorensen.pdf>. Citado 2 vezes nas páginas 15 e 43.
- SuperCollider.ORG. *SuperCollider Overviews: JITLib - An overview of the Just In Time library*. 2014. Citado na página 11.
- 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 19 e 76.

TRUAX, B. The aesthetics of computer music: a questionable concept reconsidered. v. 5, n. 3, p. 119–126, 2000. Citado 2 vezes nas páginas 50 e 79.

VIEIRA, V. et al. Vivace: A collaborative live coding language. arXiv, n. 1502, p. 16, 2015. Disponível em: <<http://arxiv.org/abs/1502.01312>>. Citado 3 vezes nas páginas 3, 18 e 85.

WALKER, B. N.; KRAMER, G. Auditory displays article xx: Sonification. In: *International Encyclopedia of Ergonomics and Human Factors*. 2. ed. [S.l.]: W. Karwowski; CRC Press, 2006. p. 1254–1256. Citado 3 vezes nas páginas 54, 57 e 87.

WANG, G.; COOK, P. R. On-the-fly programming: using code as an expressive musical instrument. In: *Proceedings of the 2004 International Conference on New interfaces for Musical Expression (NIME)*. [s.n.], 2004. p. 6. Disponível em: <http://soundlab.cs.princeton.edu/publications/on-the-fly_nime2004.pdf>. Citado 2 vezes nas páginas 15 e 87.

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 11 vezes nas páginas 15, 16, 21, 22, 28, 53, 54, 59, 62, 67 e 87.

WOOLER, R. et al. A framework for comparasion of process in algorithmic music systems. Generative Arts Practice, p. 109–124, 2005. Disponível em: <<http://eprints.qut.edu.au/6544/1/6544.pdf>>. Citado 3 vezes nas páginas 54, 59 e 87.

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 87.

Appendices

APÊNDICE A – Técnicas de recursividade

As chamadas técnicas recursivas são funções matemáticas que definem, entre suas variáveis, uma versão diferente dela mesma; um caso clássico é a definição da *Série de Fibonacci*, série numérica infinita idealizada por Fibonacci (c. 1170 – c. 1250) para resolver um problema geral de crescimento populacional, embora diferentes fontes citem o uso por matemáticos indianos antes do séc. VI¹. A partir dela, é possível obter a proporção áurea, número de ouro ou ϕ .

A.1 Fibonacci

Esta série e esta proporção são largamente usadas na obra de Béla Bartók (1881-1945) como ferramenta de formalizações melódicas, harmônicas e estruturais². O programa da Figura 30 resulta nas listas numéricas apresentadas na Figura 31.

No livecoding as técnicas algébricas recursivas podem ser usadas para, por exemplo, definir um grupo de alturas e durações de uma sequência de notas. Na Figura 32 temos um exemplo no SuperCollider com essa abordagem:

A.2 Delays

Outra abordagem pode se basear apenas na característica "processual", ou seja, não se reduz ao processo algébrico demonstrado acima; a recursividade pode ser usada como técnica de DSP, onde o sinal de entrada de um módulo pode ser ele mesmo levemente alterado. O uso prático mais direto disso, é com *Delays*, como é o caso da Figura 34. No *CSound Book* (p.422), isso pode ser descrito a partir do design de *resposta de impulso* e *filtros de resposta de impulso infinita* (IIR).

A.3 Batimentos

Este código exemplifica um modelo de improvisação segundo *Crossings* de Alvin Lucier.

¹ <https://en.wikipedia.org/wiki/Fibonacci#Fibonacci_sequence>, acessado em 27 de junho de 2015.

² Para mais detalhes, sugerimos a leitura de uma tese defendida recentemente por Guilherme Soares (2015-03-13)

```

1 // Define função de fibonacci
2 function fib(n){
3     if(n<0){
4         throw new Error("De um valor maior que 0");
5     }
6     else if(n === 0 || n === 1){
7         return 1;
8     }
9     else{
10        return fib(n-2) + fib(n-1);
11    }
12}
13
14 // Define uma lista de valores de acordo
15 // com uma função abstrata
16 function imprime_lista(fn, n){
17     var lista = []
18     for(var i=0; i<n; i++){
19         lista.push(fn(i));
20     }
21     return lista;
22}
23
24 // Define a razão aurea
25 function phi(n){
26     var lista = imprime_lista(fib, n);
27     return lista[lista.length-2]/lista[lista.length-1];
28}
29
30
31 // imprime uma lista de valores de
32 // acordo com a função de fibonacci
33 var lista = imprime_lista(fib, 15);
34 console.log("Sequencia numerica de fibonacci:")
35 console.log(lista);
36
37 // imprime uma lista de razões aureas
38 var phis = imprime_lista(phi, 15);
39 console.log("Sequencia numerica de razoes aureas:");
40 console.log(phis);

```

Figura 27 – Programa escrito em javascript que define uma função de fibonacci e cria uma série a partir dela, bem como uma lista de razões áureas

```
guilherme@R410-L-BP12P1:~/Dropbox/Mestrado/monografia/codigos$ node fibonacci.js  
Sequência numérica de fibonacci:  
[ 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610 ]  
Sequência numérica de razões áureas:  
[ NaN,  
  NaN,  
  1,  
  0.5,  
  0.6666666666666666,  
  0.6,  
  0.625,  
  0.6153846153846154,  
  0.6190476190476191,  
  0.6176470588235294,  
  0.6181818181818182,  
  0.6179775280898876,  
  0.6180555555555556,  
  0.6180257510729614,  
  0.6180371352785146 ]  
guilherme@R410-L-BP12P1:~/Dropbox/Mestrado/monografia/codigos$ █
```

Figura 28 – Resultado do programa da Figura 30

```

1 // Liga servidor do SuperCollider
2 s.boot
3
4 // define Funcao fibonacci
5 ~fib = { |n|
6     var result;
7     var rec = { |k| ~fib.value(k-2) + ~fib.value(k-1)};
8     if(n==0,
9         {result=1},
10        {if(n==1, {result=1}, {result=rec.value(n)})})
11    );
12    result;
13 }
14
15 // Define serie numerica
16 ~fiblist = { |n| var lista = { |i| ~fib.value(i)}!n; lista}
17
18 // Define ritmos e melodias
19 ~phi = { |n|
20     var lista = ~fiblist.value(n);
21     lista[lista.size-2]/lista[lista.size-1]
22 }
23 ~lista = { |n, k|
24     { |i|
25         ~fiblist.value(n).choose * ~phi.value(n) / k
26     }!n
27 }
28
29 // A melodia; a melodia, com as proporcoes
30 // de fibonnaci, mudam de instante em instante
31 Ndef('fib_melodia').fadeTime = 5
32 Ndef('fib_melodia', Pdef(\melodia)).play
33 Pdef('melodia',
34     Pbind(
35         'dur',
36         Prand(~lista.value(9.rand+2, Pdef('melodia')), inf),
37         'degree',
38         Prand(~lista.value(9.rand+2, 15.rand+1), inf)
39     )
40 );

```

Figura 29 – Programa escrito no SuperCollider que cria uma sequencia infinita de notas baseada na razão áurea.

```

1 s.boot
2
3 Ndef(\impulso, {Impulse.ar(1)})
4 Ndef(\delay, {
5     AllpassC.ar(
6         \in.ar,
7         1.0,
8         \time.kr,
9         1.0,
10        1.0,
11        \in.ar * Ndef(\delay).ar * \feedback.kr(0.071)
12    )
13 }).play
14
15 Ndef(\delay) <>> .in Ndef(\impulso)

```

Figura 30 – Programa escrito no SuperCollider que cria uma rede de delays infinita.

```

1 s.boot
2
3 //lowest A at 27.5 Hz.
4 //to the highest C at 4186 Hz.
5 ~minuto = 60
6 ~total = ~minuto * 16
7 Ndef(\senoide, {
8     SinOSc.ar(XLine.kr(27.5, 4186, ~total), mul: 0.71)
9 })
10
11 // Um som de guitarra
12 // deve ser tocado uma nota
13 // a cada 10 segundos
14 // (Uma oitava a cada 2 minutos)
15 Ndef(\nota, {|note=27.5.freqtomidi|
16     Pluck.ar(Saw.ar(note, 0, 1), Impulse.kr(0.1), 0.1, 10, 0.5, 1)
17 }).play
18
19 Ndef(\senoide).play
20
21 //Modificar
22 //Tocar isso denovo quando acabar
23 Ndef(\nota, 1).play

```

Figura 31 – Programa escrito no SuperCollider que cria uma rede de delays infinita.

APÊNDICE B – Webaudio API e Wavepot-runtime

B.1 Wavepot

No ambiente virtual *Wavepot*¹, podemos usar o paradigma de dois arquivos, semelhante ao instrumento e orquestra do *CSound*; no primeiro podemos definir os instrumentos, e no segundo o processamento geral. Uma nota: os comentários iniciais do arquivo dos instrumentos são usados para definir o autor do script, o nome do script e o que faz para ser salvo como metadado. Para definir um instrumento, deve-se usar o token **export**.

O programa escrito pode ser salvo e compartilhado na rede social *Github*²:

```

1  /*
2   * @name ruido
3   * @author anonimo
4   */
5  export noise(a){
6      return a*(Math.random() * 2 - 1);
7 }
```

Figura 32 – Algoritmo de um ruído branco sendo salvo no Wavepot

Em seguida, pode-se usá-lo invocando o token **import**; em seguida deve ser usada a função *dsp*, chamando novamente **export**:

```

1 import "anonimo/ruido"
2
3 export dsp(t){
4     return noise(0.71);
5 }
```

Figura 33 – Algoritmo de um ruído branco sendo salvo no Wavepot

É possível também salvar vários módulos em um único arquivo, como na Figura 40.

Em outro arquivo, na mesma janela é possível usar a função *dsp* para processar, como na Figura 38.

¹ <<http://www.wavepot.com>>, acessado em 27 de junho de 2015.
² <<https://www.github.com>>, acessado em 27 de junho de 2015.

```

1  /*
2   * @name osc
3   * @author anonimo
4   */
5  var TAU = 2 * Math.PI;
6  export noise(a){
7      return a*(Math.random() * 2 - 1);
8 }
9
10 export sin(f, a, t){
11     return a*(Math.sin(TAU*f*t))
12 }
```

Figura 34 – Duas funções básicas em um mesmo arquivo; um ruído e uma senóide.

```

1  export dsp(t){
2      noise(sin(0.001));
3 }
```

Figura 35 – Síntese sonora com o *wavepot*. monoaural

Para criar uma diferença entre canais (2), ver [Figura 39](#).

```

1  var stereo = [];
2
3  export dsp(t){
4      stereo[0] = noise(sin(0.001));
5      stereo[1] = noise(sin(0.002));
6      stereo;
7 }
```

Figura 36 – Síntese sonora com o *wavepot*. estereofônica.

B.2 Coffeepot

Em um ambiente virtual, em linguagem *coffeescript*³, um usuário pode usar os mesmos procedimentos que o *Wavepot*. Definindo uma função "dsp", que será compilada para *javascript* e então atualizado na máquina virtual demonstrada na [Figura 43](#):

³ <<http://www.coffeescript.org>>, acessado em 27 de junho de 2015.

```

1 # Processamento de Sinal Digital
2 # calculo de um valor randomico
3 # entre -1 e 1 a cada janela
4 # temporal (t).
5 # Som resultante: um ruido branco
6 dsp = (t) -> Math.random() * 2 - 1

```

Figura 37 – Algoritmos de processamento e gravação de uma função que pode ser chamada dinamicamente (`this.dsp`) em linguagem *coffeescript*. A função compilada em javascript está demonstrada na [Figura 41](#).

```

1 TAU = 2*PI
2 noise = (a) -> a * Math.random() * 2 - 1
3 sin = (f,a,t) -> a*Math.sin(TAU*f*t)
4 dsp = (t) -> noise sin sin(1,sin(2,1,t),t), sin(2,1,t), t

```

Figura 38 – Função `dsp` compilada; esta será a função utilizada na figura [Figura 43](#).

Da mesma forma que a seção anterior, podemos concatenar diferentes funções, pensando que, devido a uma visível simplificação no uso de *tokens*, processos mais complexos podem ser realizados em uma unica linha:

B.3 Termpot

O algoritmo a seguir demonstra como o Prof. Dr. Luiz Schiavonni e eu desenvolvemos uma máquina de DSP, *Termpot*⁴, inspirado no original *Wavepot*⁵, a partir de um outro projeto-clone conhecido como *AudioToy*, ao qual aplicamos algumas customizações (ver [Figura 42](#)).

Desde que é utilizado a biblioteca WebAudioAPI, é necessário inicializar apropriadamente o Contexto de Áudio de cada navegador de internet; nesse contexto foi criado um escopo de processamento; assim o escopo pode ser constantemente modificado, e o processador propriamente dito, semelhantemente à técnica *wavetable synthesis* abordada de forma sucinta em [Puckette \(2006\)](#). Adicionamos também um gravador através da biblioteca *recorder.js*:

Colocar
uma
seção a
respeito
como
primeira.

Figura 39 – Máquina virtual clone do Wavepot

A seguir, é utilizada a técnica de processamento *amostra por amostra*. Pressupondo que estamos lidando com o tempo discretizado, em janelas de 512, 1024, 2048 ou 4096

⁴ <<https://www.github.com/jahpd/coffeepot>>

⁵ <<https://www.wavepot.com>>

amostras por segundo, com uma resolução de áudio de 44.1kHz e 16 bits; ou seja, a cada janela temporal de 0.023219954648526078 segundo (43 vezes em um ciclo completo de 1 segundo) é calculado um valor numérico de ponto-flutuante entre 65536 possíveis.

Figura 40 – Algoritmos de processamento e gravação de uma função que pode ser chamada dinamicamente (`this.dsp`)

A função DSP e um ambiente comum devem ser definidos. Neste trecho devo dizer que sou muito agradecido ao Schiavonni, que além de solucionar problemas de *clicks* entre eventos do usuário, propôs o código que será mostrado abaixo, otimizando o processamento ([Figura 47](#)) e sugerindo a possibilidade de se programar uma *GUI* (Graphical User Interface), que foram alguns *sliders* para controle de áudio.

Uma menção ao comando `def`: ele pode ser utilizado da seguinte forma no aplicativo:

E para criar uma GUI:

Figura 41 – Compilador de códigos em áudio, contendo o ambiente de áudio de funções pré-definidas pelo usuário.

```

1 // registra um novo modulo
2 // e adiciona uma gui
3 function addSlider(name, min, max){
4     min = parseFloat(min)
5     max = parseFloat(max)
6     if(control_vars.indexOf(name) === -1){
7         control_vars.push(name)
8         runtime.addControl(name, {value: 0})
9         control_vars.push(name)
10        $('<div>' + name + '</div>').attr('id', name + '_slider').addClass("GUI").slider({
11            range: "min",
12            animate: true,
13            orientation: "vertical",
14            min: min,
15            max: max,
16            slide: function(event, ui){
17                var name = $(this).attr('id').split('_')[0]
18                runtime.setControl(name, {value: map(ui.value, min, max, 0, 1)})
19            }
20        }).appendTo("#control_container")
21    return {
22        type: 'print',
23        out: register_module("def "+name+"() if controls["+name+
24        "\"] then controls["+name+
25        "\"][" + "value\"] else new Error '"+name+
26        "' not available' #returns a value between "
27        +min+
28        " & "+
29        max+
30        " from "+name+
31        "' slider ")
32    }
33 }
34 else{
35     $('#'+name+'_slider').slider({
36         range: "min",
37         animate: true,
38         orientation: "vertical",
39         min: min,
40         max: max,
41         slide: function(event, ui){
42             var name = $(this).attr('id').split('_')[0]
43             runtime.setControl(name, {value: map(ui.value, min, max, 0, 1)})
44         }
45     });
46 }
47 return {
48     type: 'print',
49     out: name+ ' redefined'
50 }
51 }
52

```

```
1 \$> wavepot
2 wavepot> def ruido_branco(amp) Math.random() * 2 - 1
3 true
4 wavepot> ruido_branco(0.7)
5 true
6 wavepot> play
```

Figura 43 – Sequencia de comandos e a utilização do comando *def* para criar um ruído branco

```
1 wavepot> slider("ruido_amp", 0, 1024)
2 wavepot> ruido_branco ruido_amp()
```

Figura 44 – Criando um ruido branco e controlando a amplitude com um slider

APÊNDICE C – Peças realizadas pelo autor da tese utilizando live coding como processo criativo

Rhythms for Why the Lucky Stiff

Essa peça realizei com um *software* desenvolvido inicialmente como experimentos diversos¹, possuindo uma primeira versão² e uma segunda versão³. Essa peça é baseada na geração randômica de parâmetros timbrísticos de um instrumento de cordas utilizando a síntese Karplus-Strong, de uma sala virtual e de um “embaralhador de buffers de áudio”. Do ponto de vista da estrutura de notas e ritmos, ela é parcialmente determinística. O cruzamento dessas duas abordagens possibilitou a criação de uma peça que repete estruturas musicais de maneira bastante variada. A cada loop das sequências, o tempo irá acelerar (primeira versão) ou desacelerar (segunda versão) em 1ms.

Pod binário para luquinhas ate que o pod quebre

Coincidindo com a data de testes práticos do desenvolvimento, um amigo fez aniversário. Fiz uma improvisação (*Pod*) explorando sínteses básicas de AM e saturação e compressão de sinais, possuindo envelopes diversos. Uma primeira sessão contém sons contínuos, a segunda é caracterizada por envelopes percussivos, caracterizando o termo *binário*. Como estava testando continha muitos *bugs*, é possível ouvir o *cooler* do computador em alta-rotação, caracterizando um excesso de processamento, que porventura causou um *crash* no momento de execução (*até que o pod quebre.*)

Livecoding coffeepot #test1

Teste e improvisação testando uma abordagem de sínteses complexas através de técnicas do *wavepot*⁴, com *coffeepot* já descrito. O teste acusa alguns problemas técnicos de interpolação, isto é, quando um novo código é requisitado, uma função de DSP substitue outra, o que causa mudanças de ondas sonoras ouvidas; neste espaço de tempo é possível ouvir em alguns momentos *clicks*.

#coffeepot-test 1426976254118

¹ <https://github.com/jahpd/gibberish_experiments>.

² <https://jahpd.github.io/gibberish_experiments>.

³ <<https://www.soundcloud.com/opusd/rythms-for-why-the-lucky-stiff>>.

⁴ <<https://soundcloud.com/opusd/livecoding-coffeepot-test1>>.

Segundo teste com o *software* mencionado. Aqui corrigi alguns *bugs* explícitos no teste anterior, bem como implementei a síntese multicanal (2 canais, L, R); a peça também é caracterizada por uma reflexão a respeito do *live coding*: o processo é o processo de ouvir ou o processo de codificar? Num primeiro momento, executo um código de síntese complexa e fico sem editar o código por 1'43"; vou então diminuindo e aumentando os intervalos de tempo de intervenção no código. Em alguns momentos a escuta acusa ações intensas de edição. Tentei também nessa peça realizar algumas técnicas de *feedback*, isto é, sinais se retroalimentando.

[Ruídos de Linguagens]

São peças realizadas com a técnica utilizando a placa de som do computador. Disponíveis em <<https://www.soundcloud.com/opusd/ruidos-de-lnguagens>>. Peguei arquivos binários de linguagens como C++, Perl, Python e Node (Javascript).

APÊNDICE D – Plotagem de dados do Soundcloud

D.1 Em formato de torta

Os dados abaixo são parciais de um levantamento de músicas no *Soundcloud* utilizando palavras-chave¹ relativas à produção da prática discutida neste trabalho organizadas em categorias de informação.

Palavras-chave são os termos que apresentaram maior consistência de dados (considerados pelo ponto de vista abstrato do autor do trabalho): *a) algorave/algopop*: parte considerável da produção do *livecoding* realizada em ambientes noturnos, informais ou de entretenimento (possue relação com o elemento dança); *b) livecoding/live-coding*: apresentaram dados coerentes com a bibliografia pesquisada²; *c) bytebeat*: parte considerável de uma técnica de programação musical descrita pela primeira vez por Heikkilä (2011) e aplicada no *livecoding*, isto é, apenas um fragmento dessa produção pode se encaixar como *livecoding* (um desses programas é o *Wavepot*); *d) algorithmic music*: utilizei o termo para apontar o quanto da música algorítmica publicada no *Soundcloud* poderia ser classificado como *livecoding*, em conjunto com outros termos que não foram abordados nesta pesquisa; *e) wavepot*: performances feitas com o aplicativo *Wavepot*, são por definição *livecoding*;

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

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

**como e por quem os generos sao classificados no soundcloud?
uma discussao a esse respeito poderia ser proveitosa.**

¹ No sentido de um parâmetro de procura em *Query-strings*; para mais informações, ver <https://en.wikipedia.org/wiki/Query_string>

² A exclusão do termo *live code/live coding* foi feita pois a separação criava uma ambiguidade de busca no *Soundcloud*, isto é, *live* poderia não se referir ao que pesquisamos por *livecoding*.

created_at distribution of 319 works published on soundcloud

Distribution of tag soundcloud_livecoding

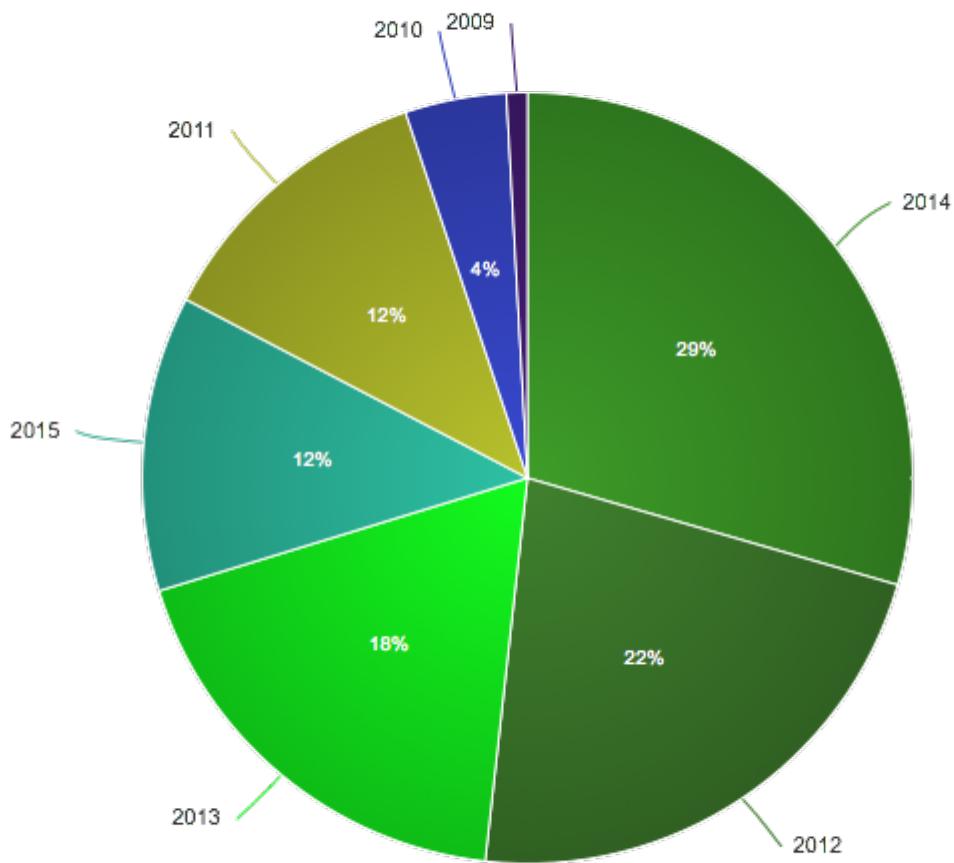


Figura 45 – Termo livecoding, analisado a partir da categoria created_at.

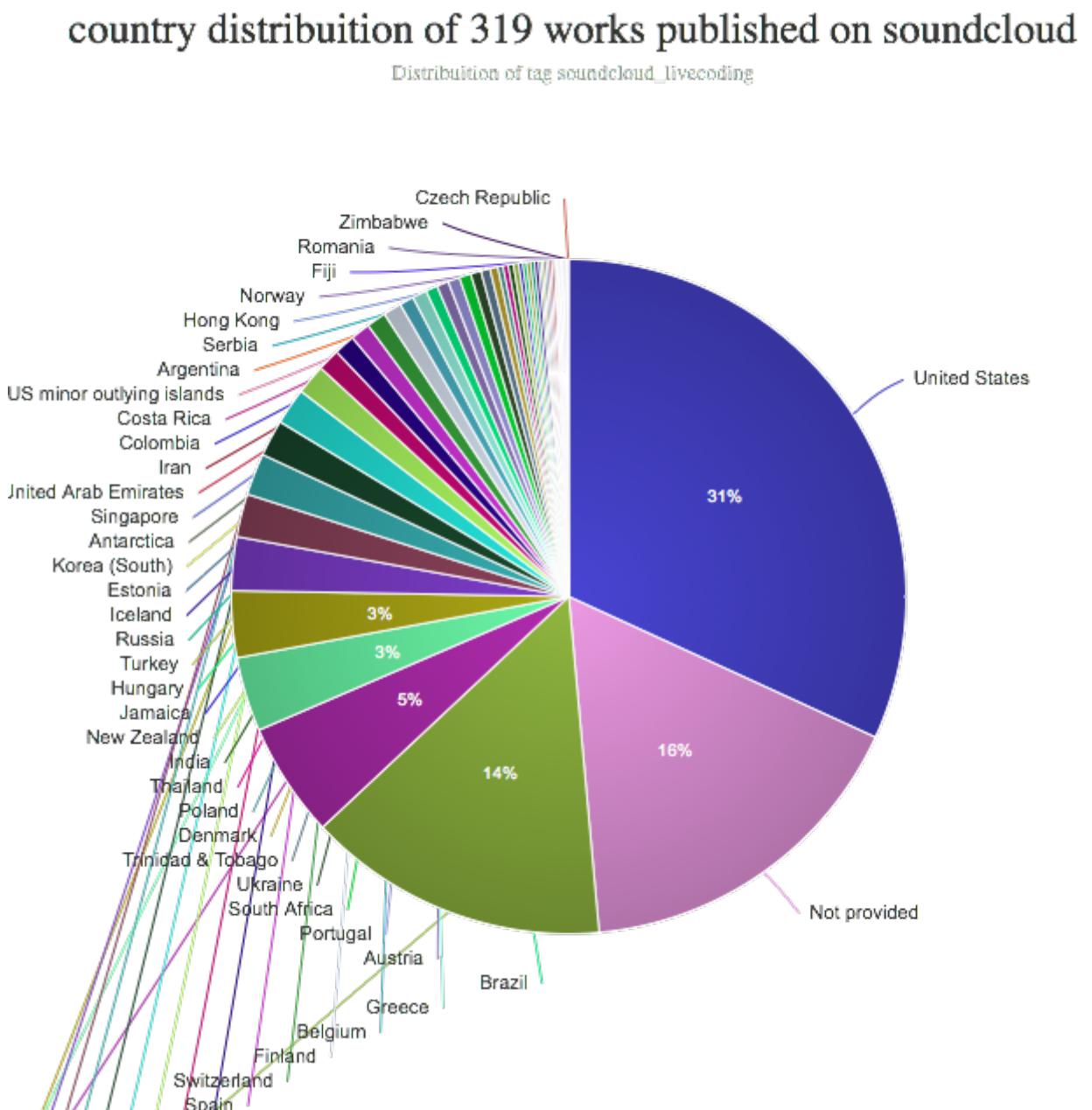


Figura 46 – Termo livecoding, analisado a partir da categoria country.

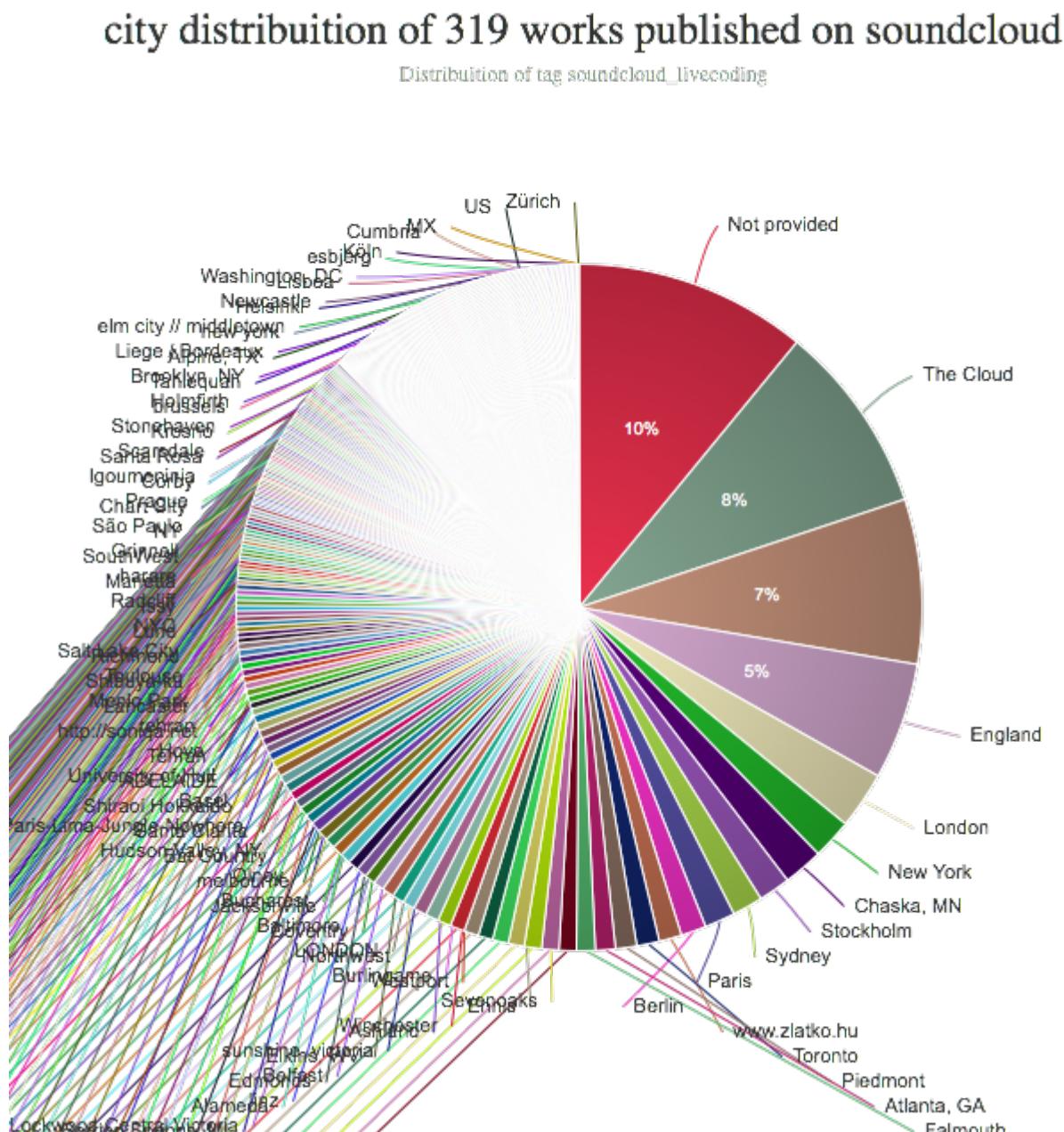


Figura 47 – Termo **livecoding**, analisado a partir da categoria **city**.

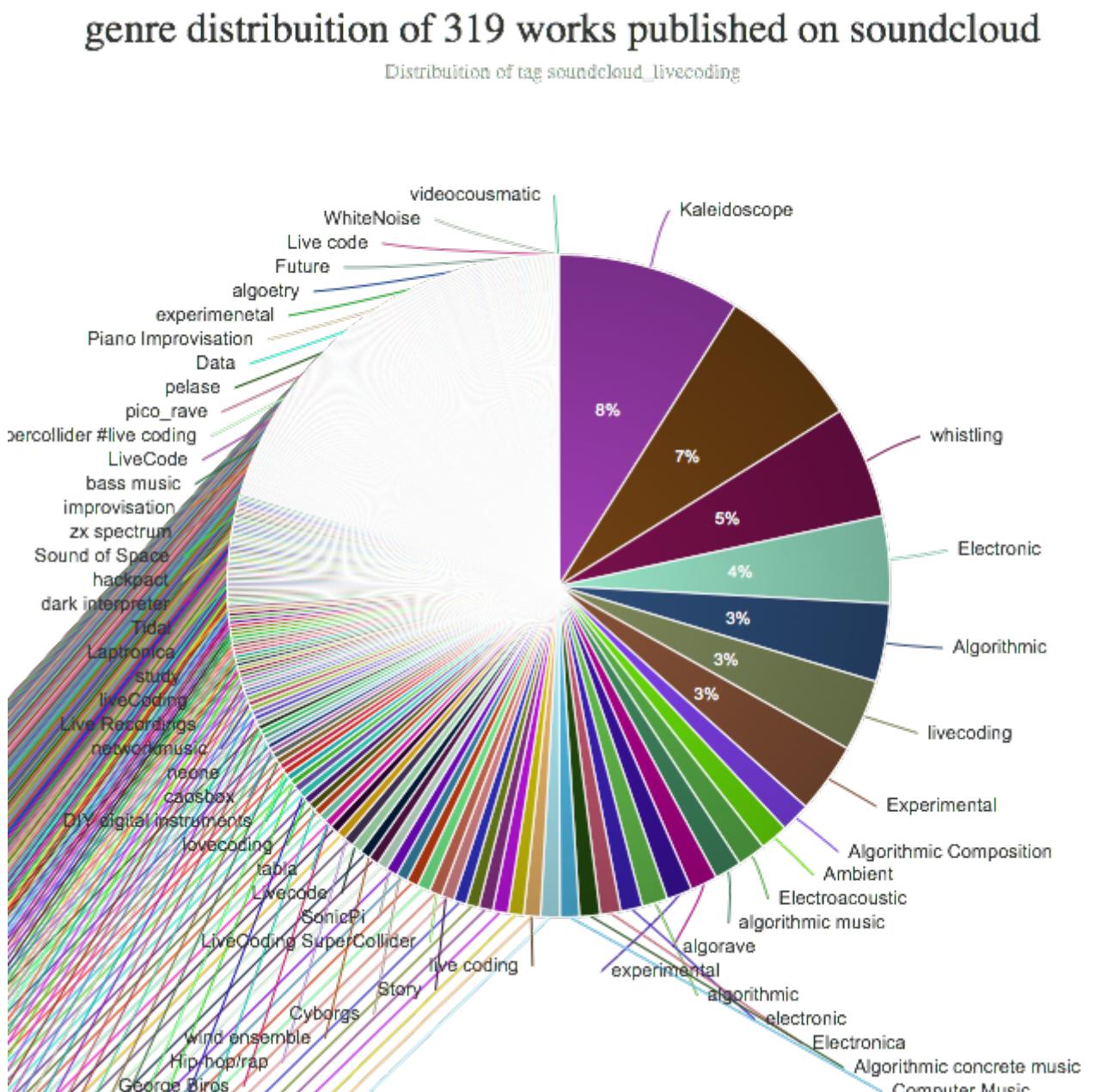


Figura 48 – Termo **livecoding**, analisado a partir da categoria **genre**.

license distribution of 319 works published on soundcloud

Distribution of tag soundcloud_livecoding

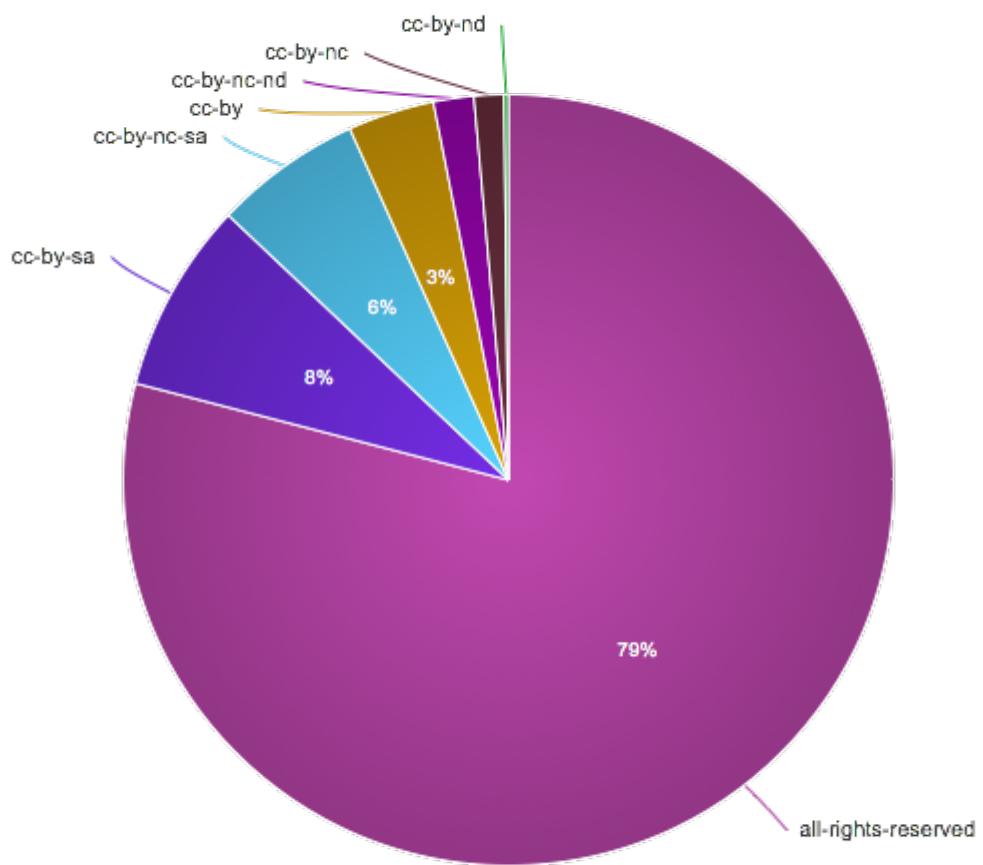


Figura 49 – Termo **livecoding**, analisado a partir da categoria **license**.

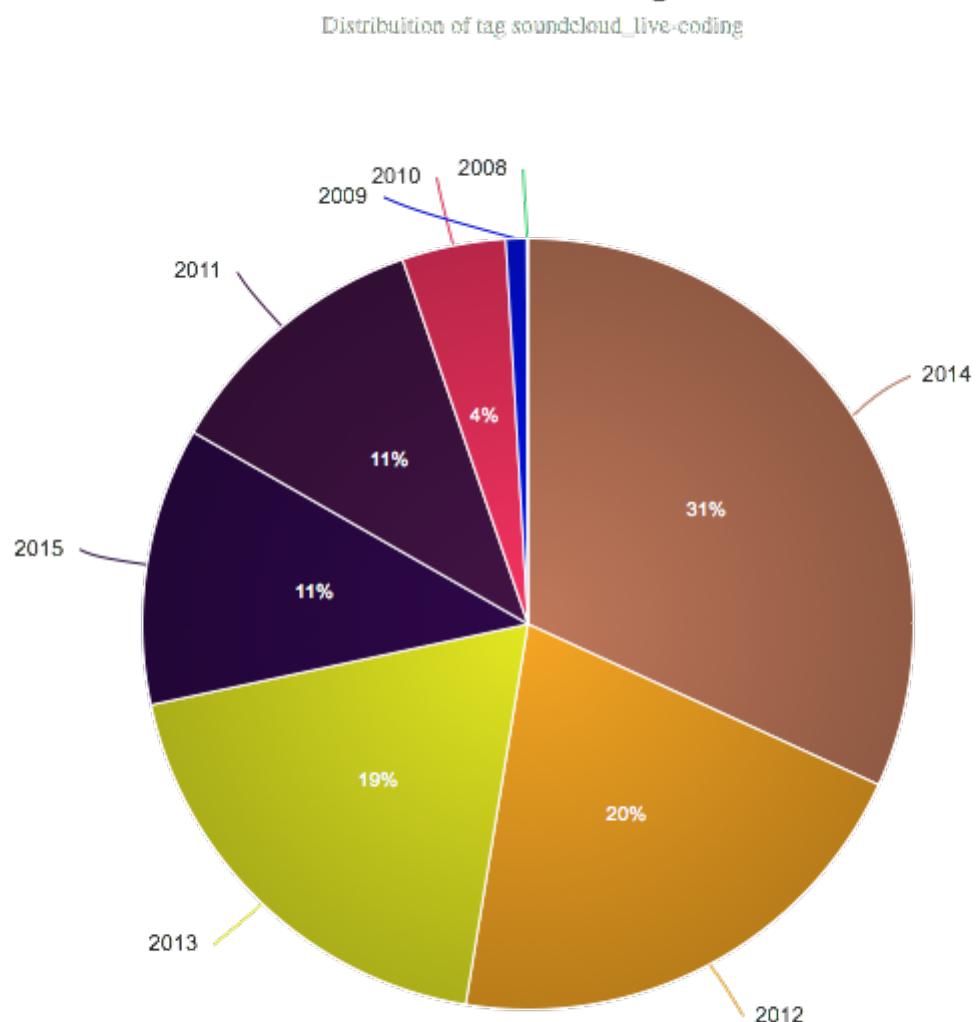
created_at distribution of 289 works published on soundcloud

Figura 50 – Termo live-coding, analisado a partir da categoria **created_at**.

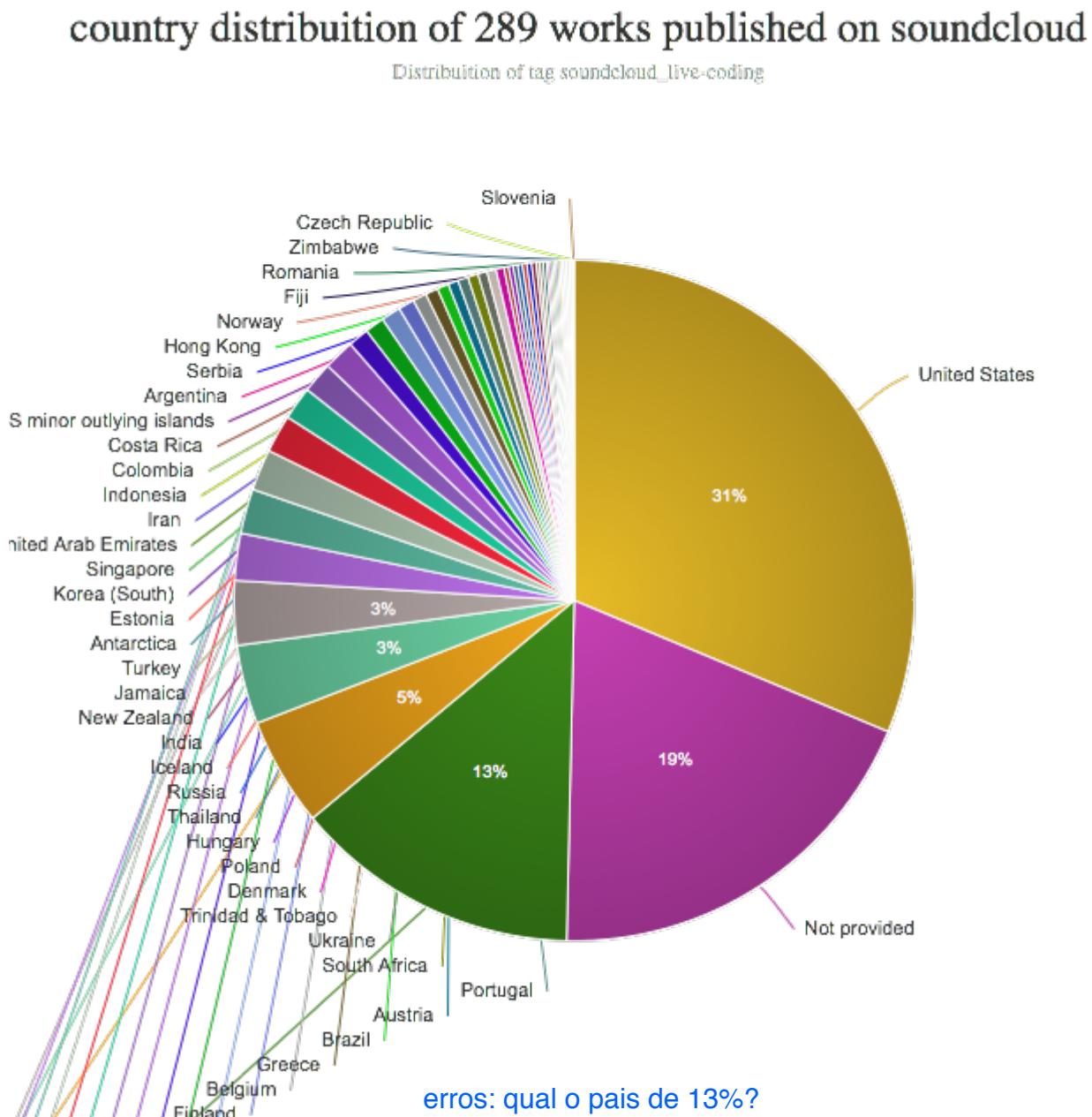


Figura 51 – Termo **live-coding**, analisado a partir da categoria **country**.

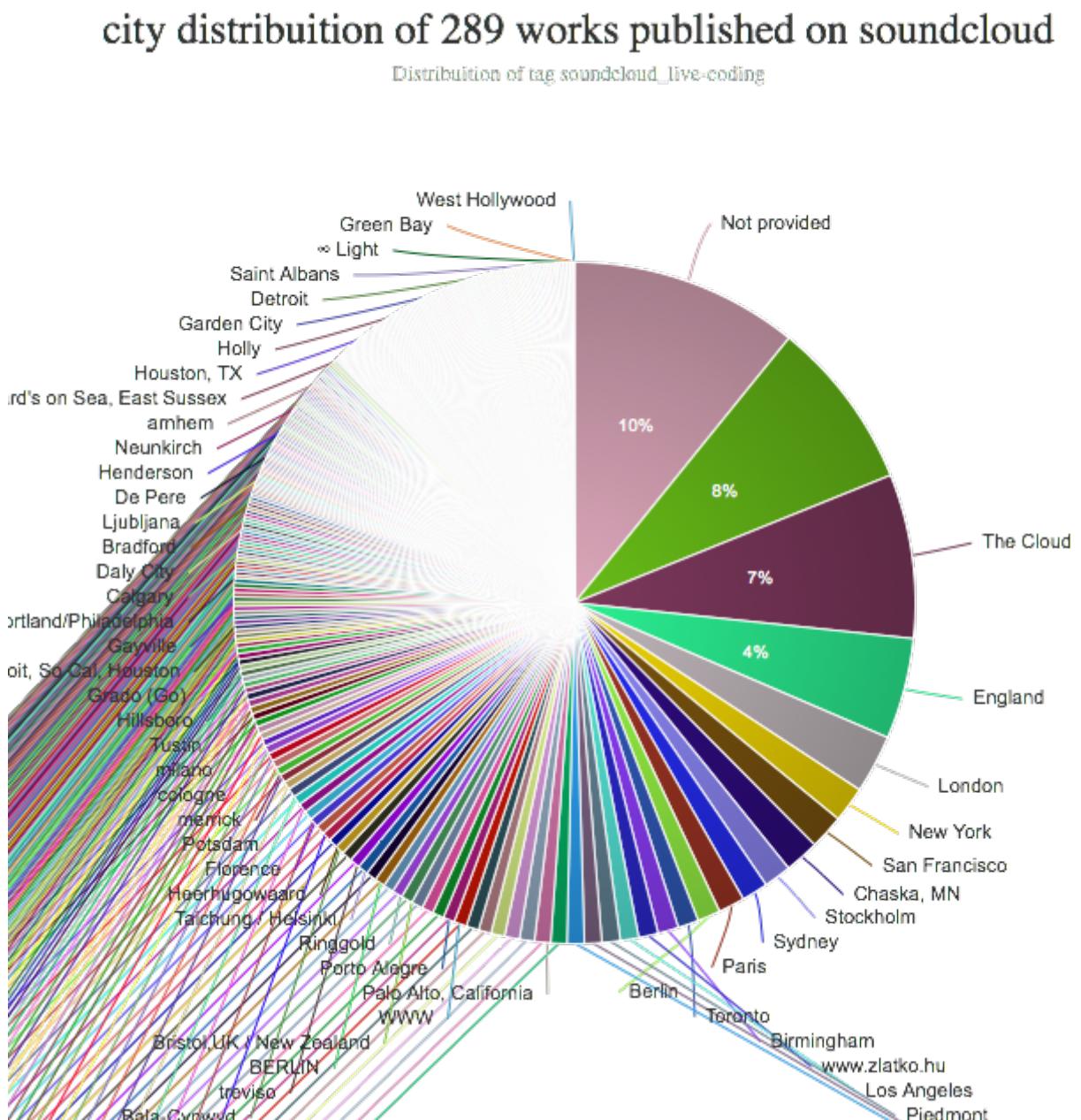


Figura 52 – Termo live-coding, analisado a partir da categoria city.

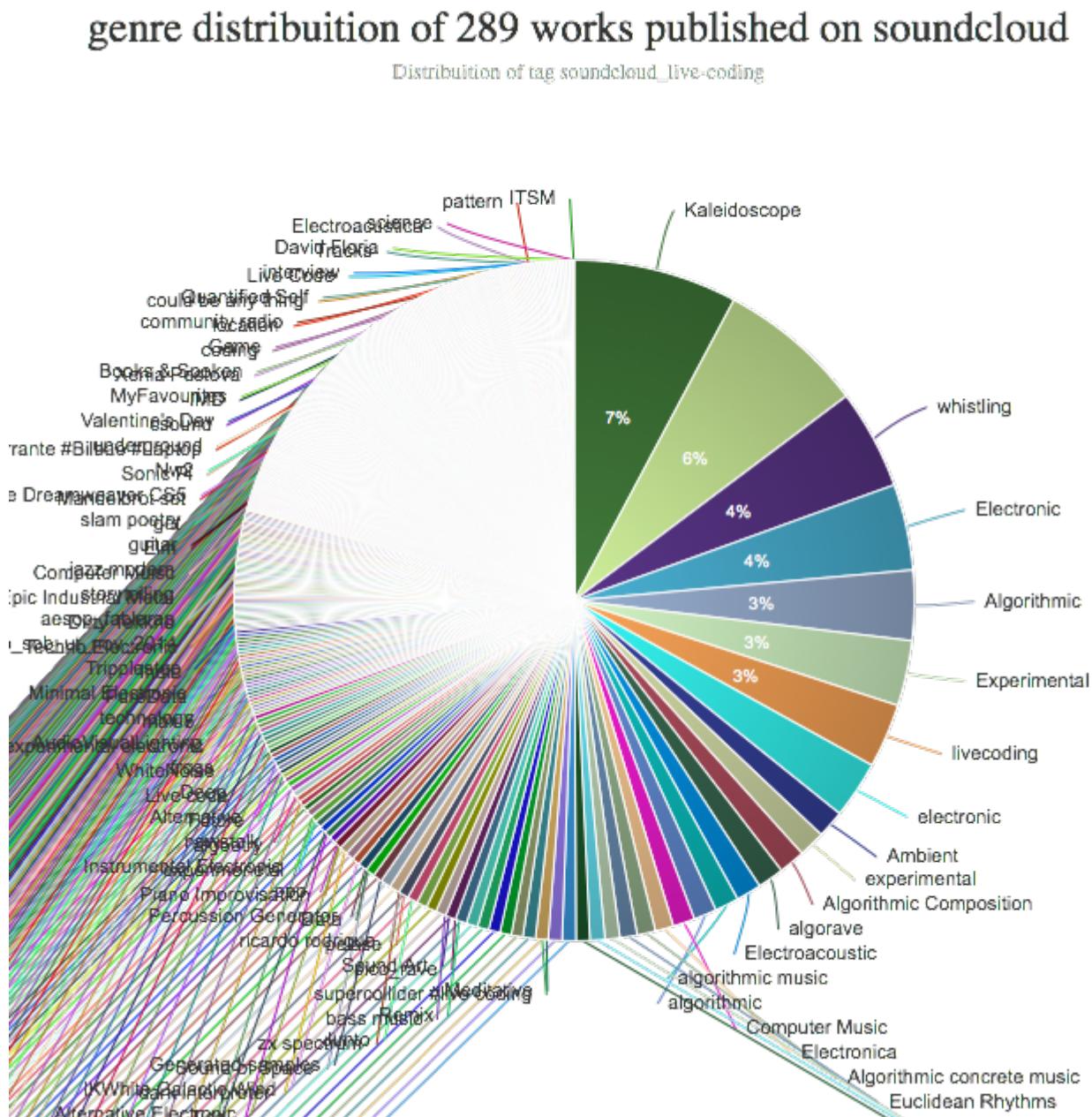


Figura 53 – Termo **live-coding**, analisado a partir da categoria **genre**.

license distribution of 289 works published on soundcloud

Distribution of tag soundcloud_live-coding

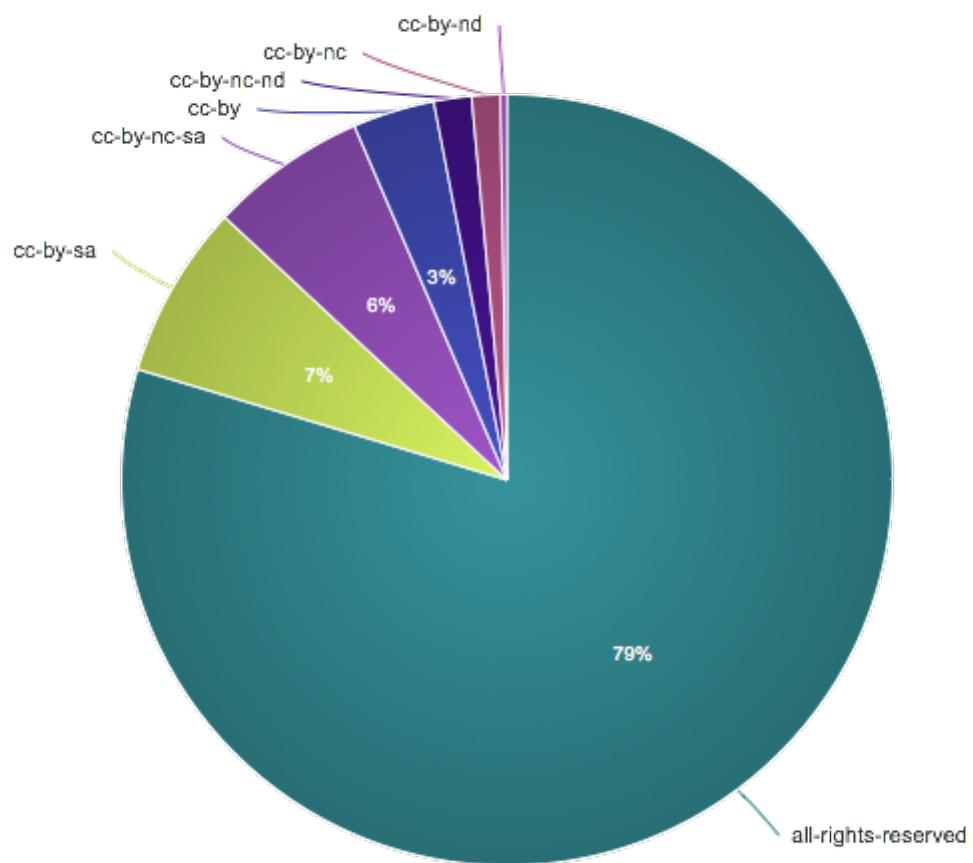


Figura 54 – Termo live-coding, analisado a partir da categoria license.

created_at distribution of 189 works published on soundcloud

Distribution of tag soundcloud_algorave

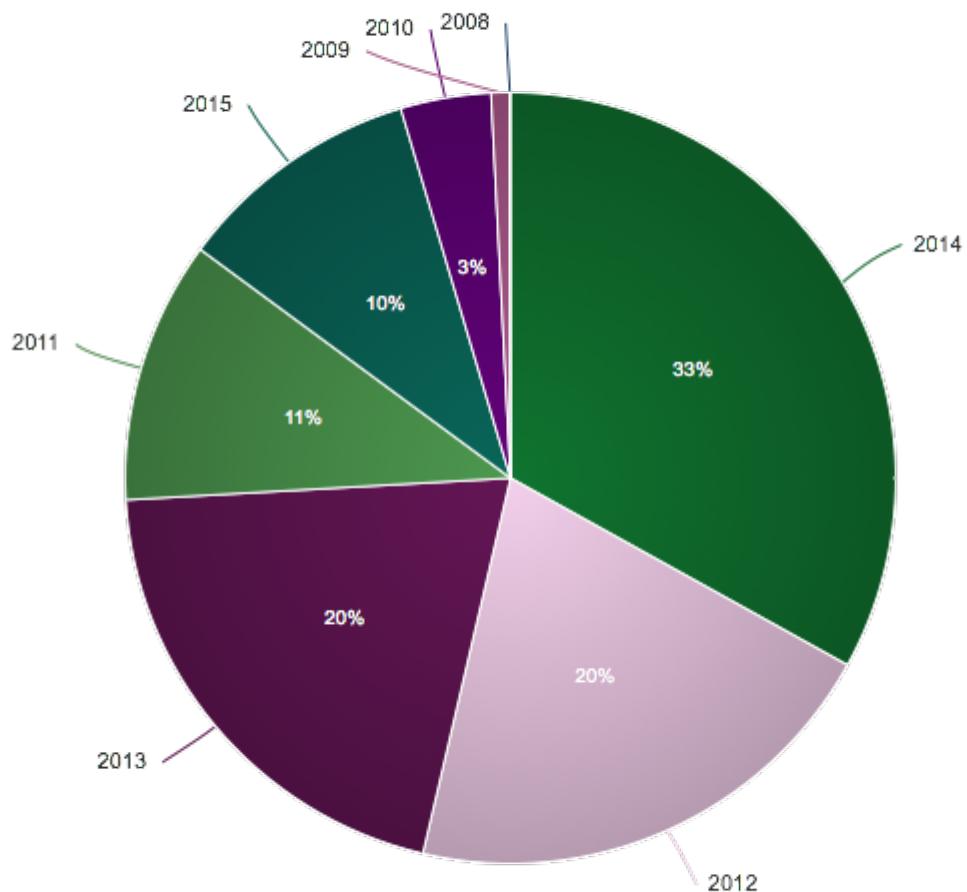


Figura 55 – Termo **algorave**, analisado a partir da categoria **created_at**.

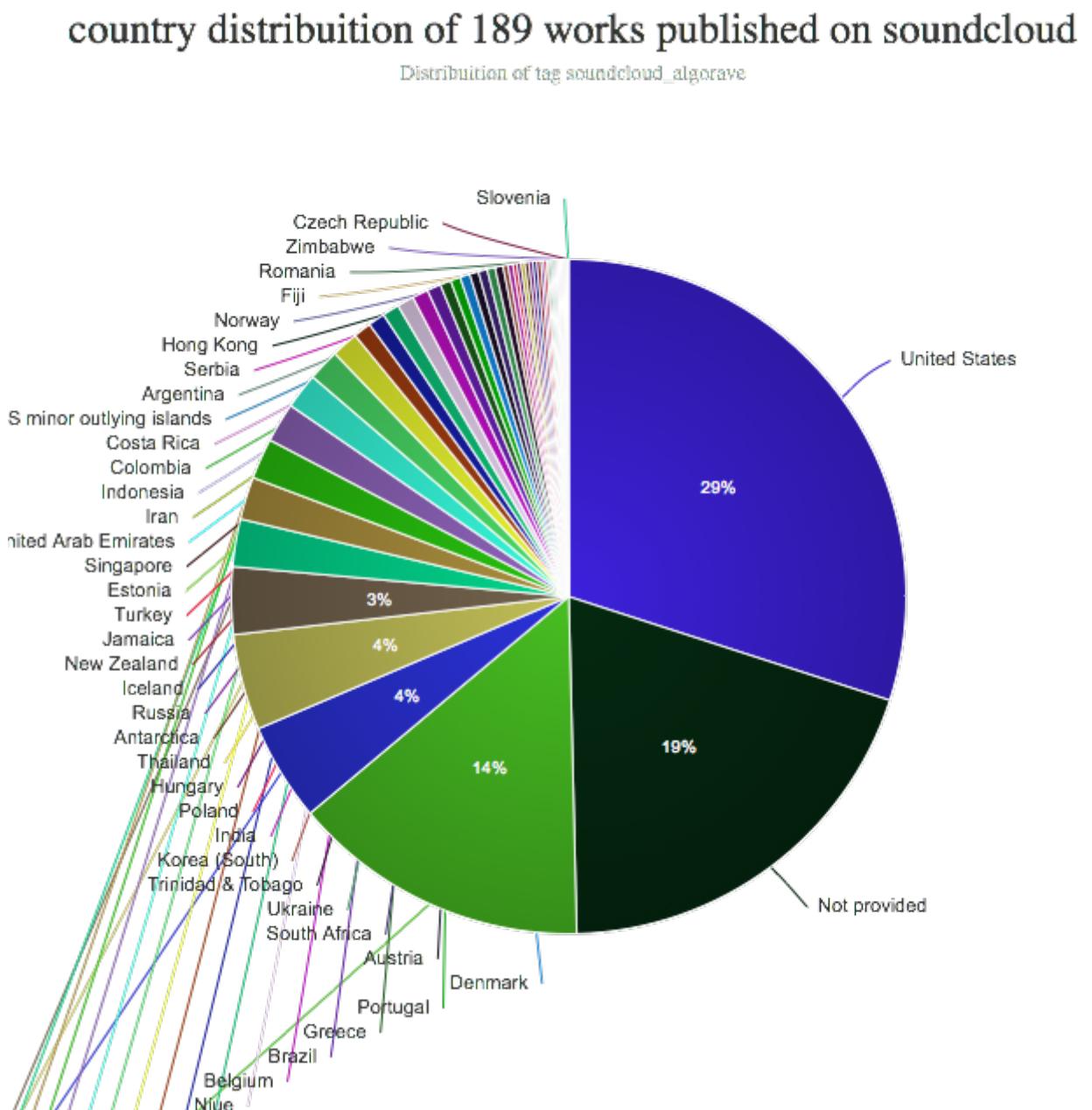


Figura 56 – Termo **algorave**, analisado a partir da categoria **country**.

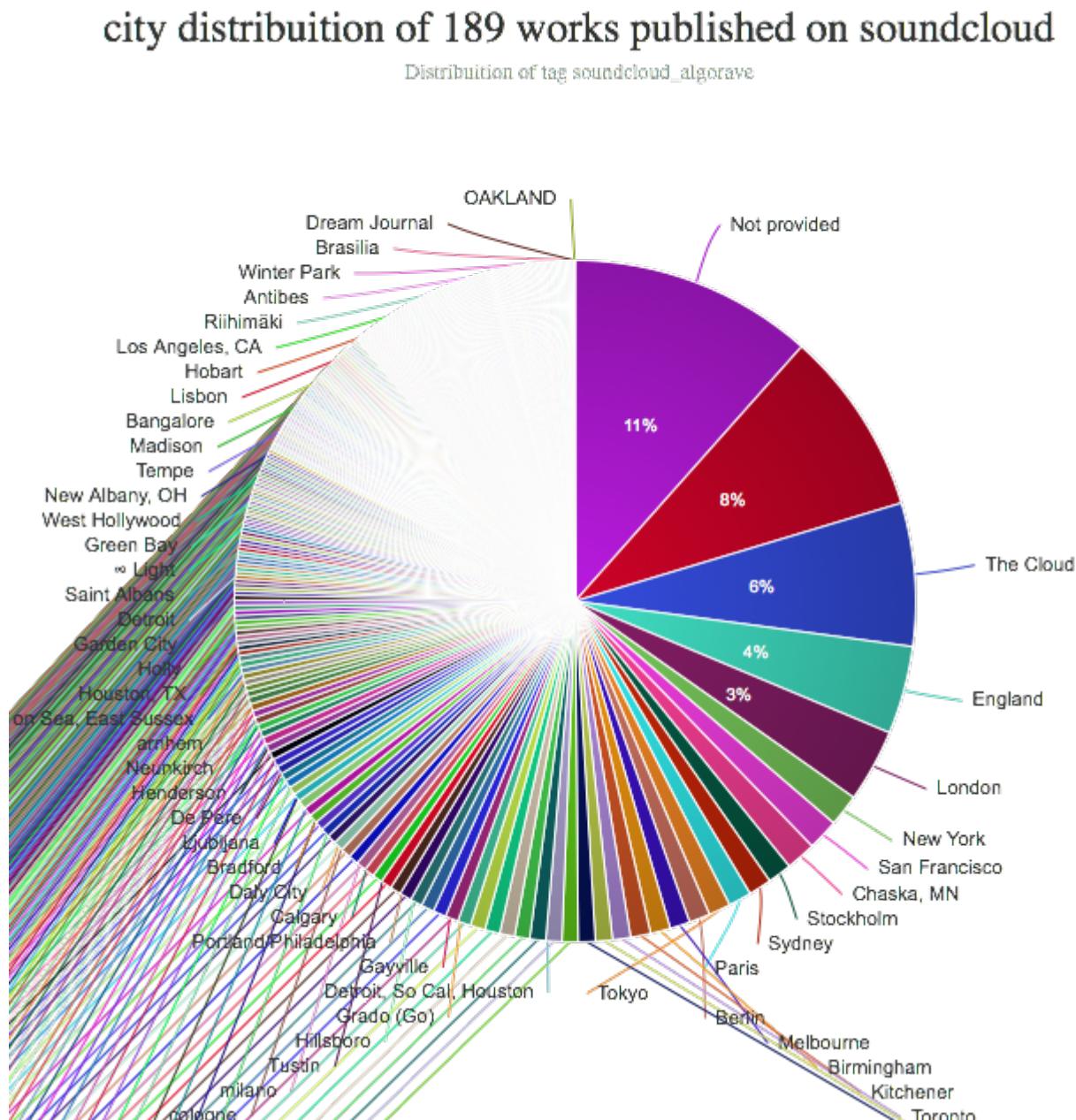
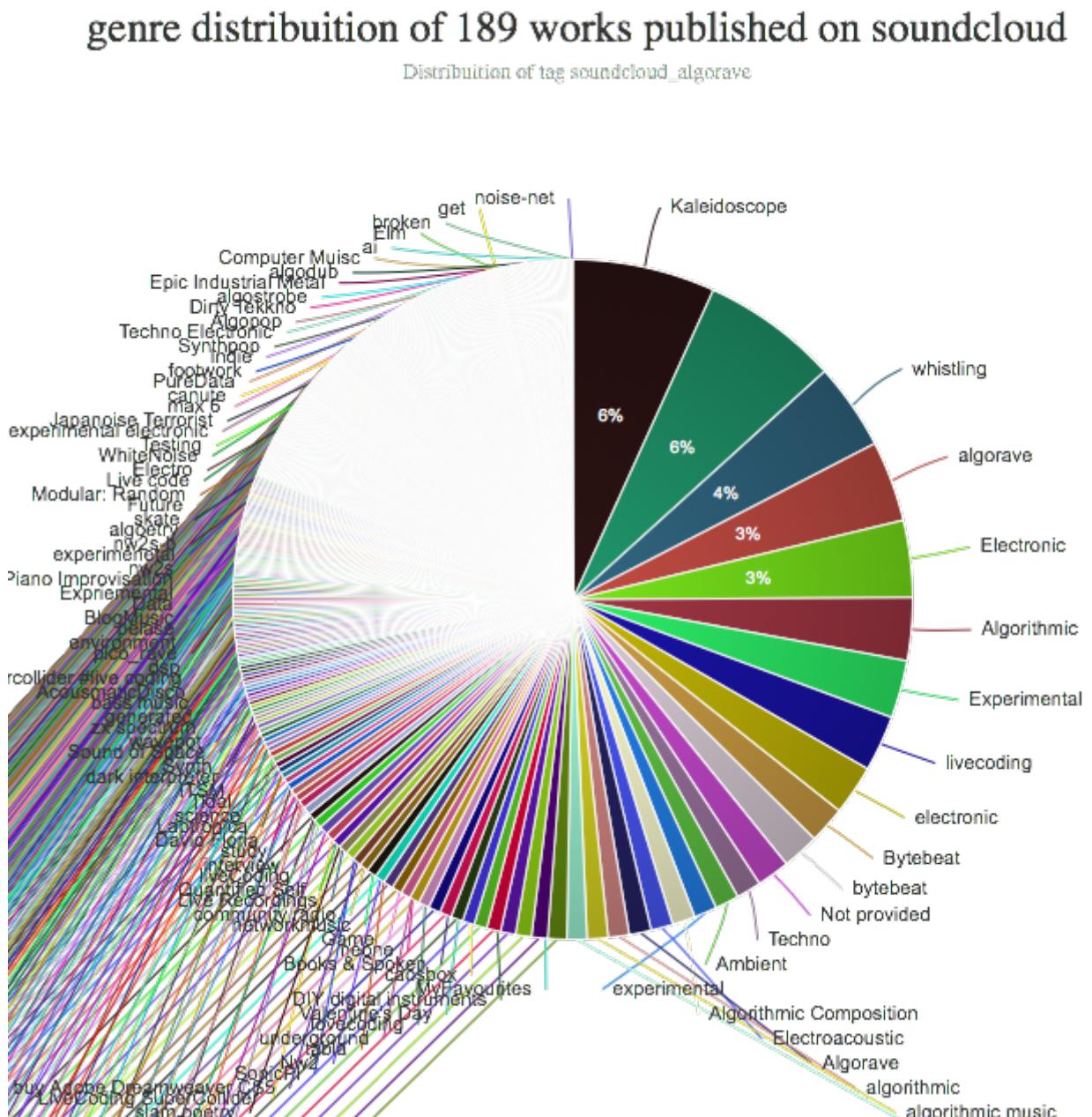


Figura 57 – Termo algorave, analisado a partir da categoria city.

Figura 58 – Termo **algorave**, analisado a partir da categoria **genre**.

license distribution of 189 works published on soundcloud

Distribuition of tag soundcloud_algorave

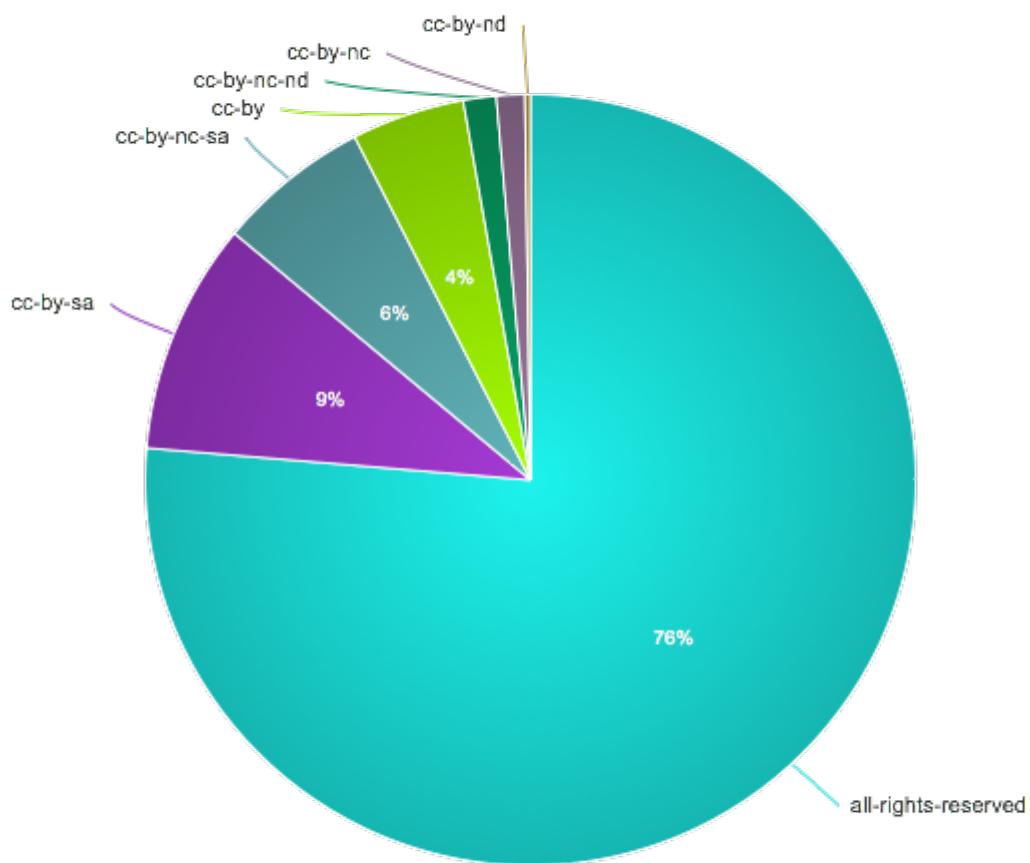


Figura 59 – Termo **algorave**, analisado a partir da categoria **license**.

created_at distribution of 8 works published on soundcloud
Distribution of tag soundcloud_algopop

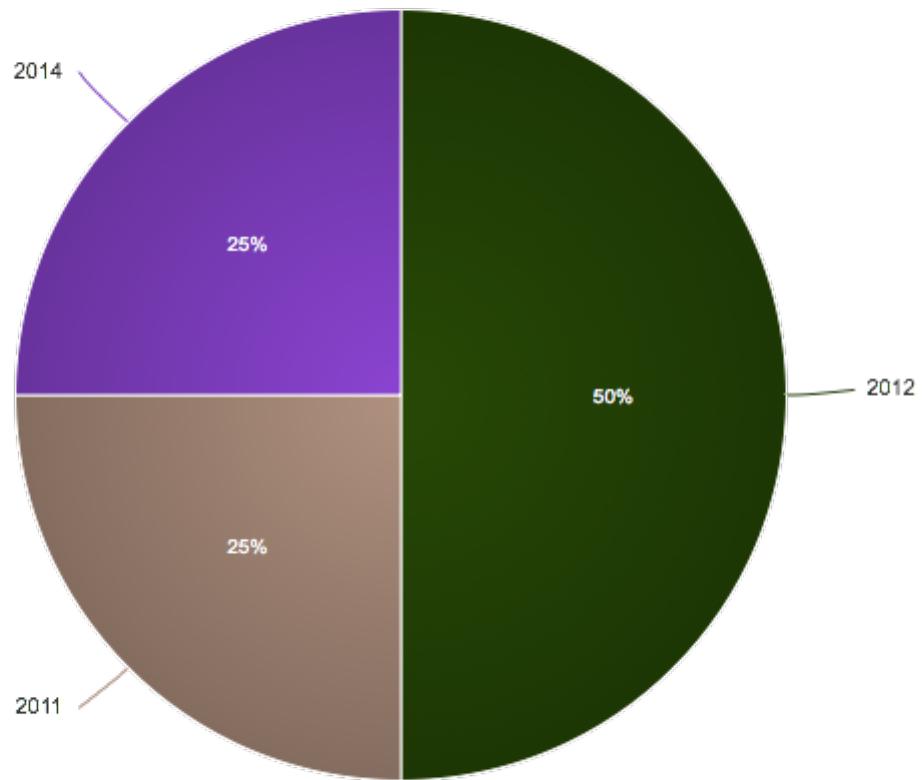


Figura 60 – Termo **algopop**, analisado a partir da categoria **created_at**.

country distribution of 8 works published on soundcloud

Distribution of tag soundcloud_algopop

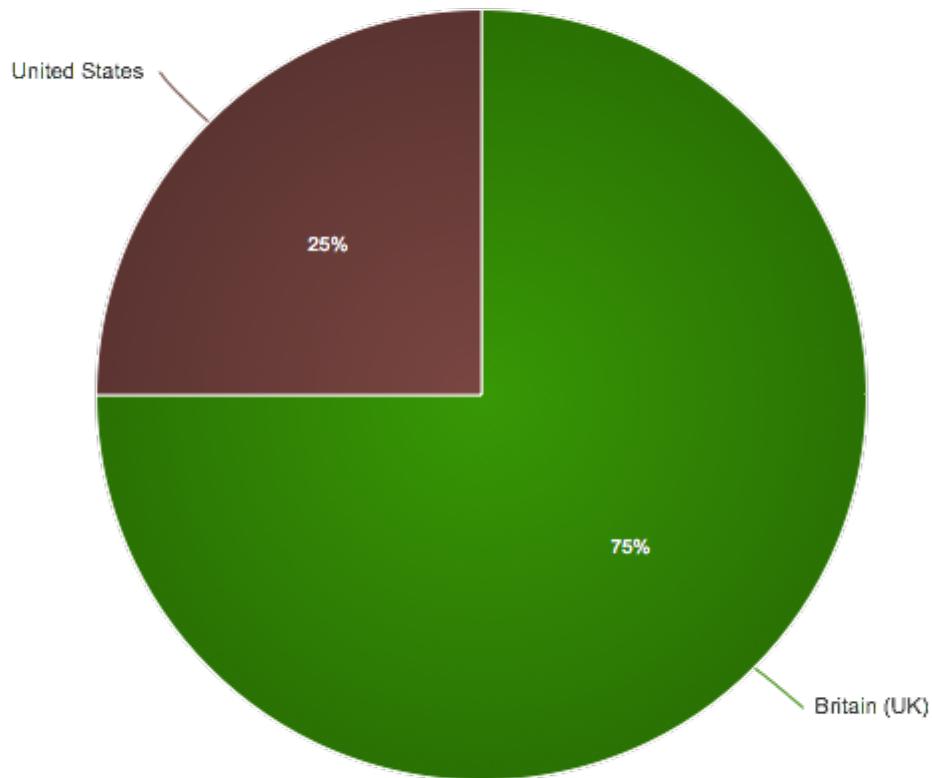


Figura 61 – Termo **algopop**, analisado a partir da categoria **country**.

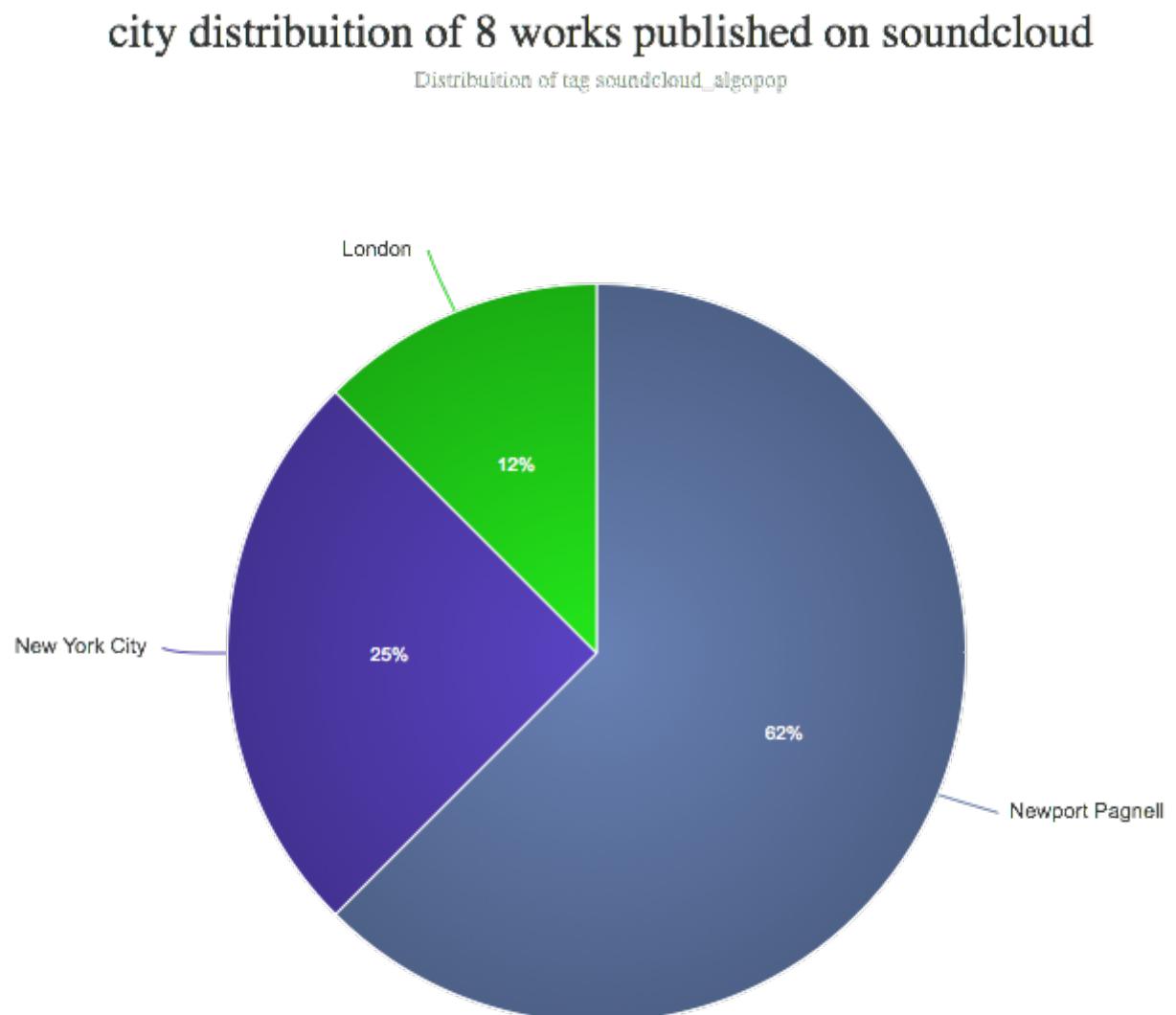


Figura 62 – Termo **algopop**, analisado a partir da categoria **city**.

genre distribution of 8 works published on soundcloud
Distribution of tag soundcloud_algopop

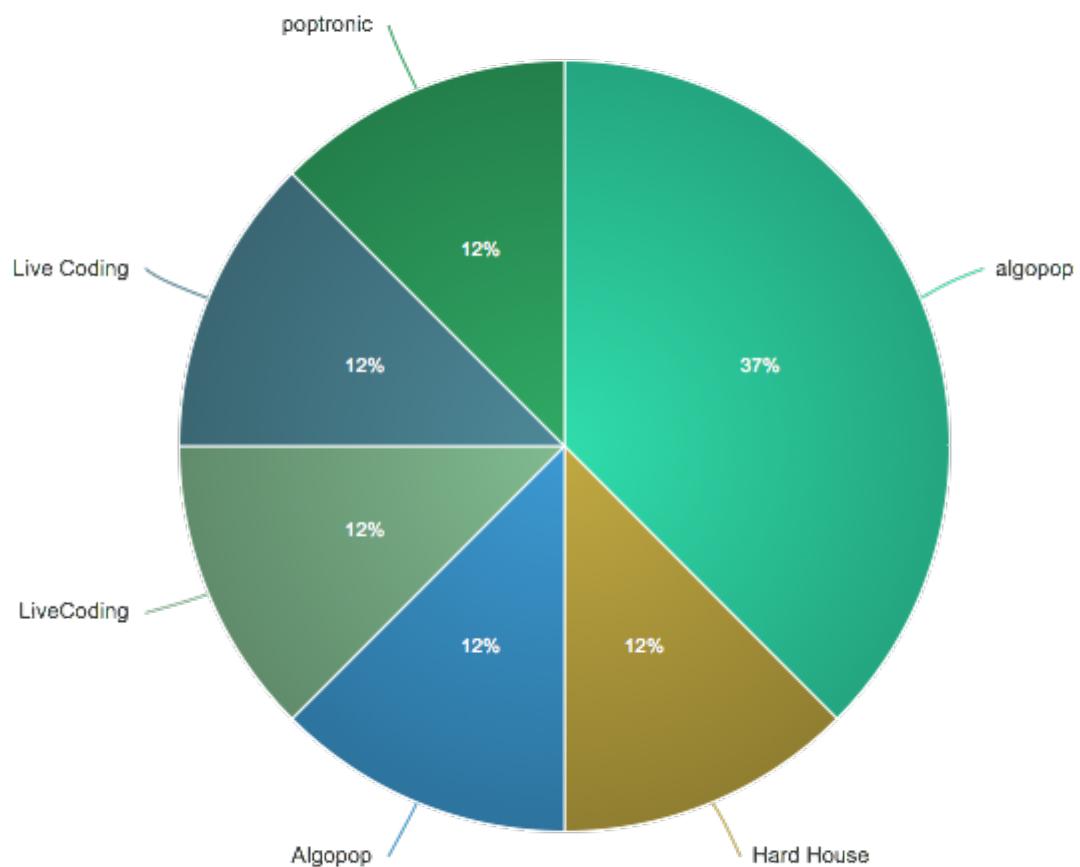


Figura 63 – Termo algopop, analisado a partir da categoria genre.

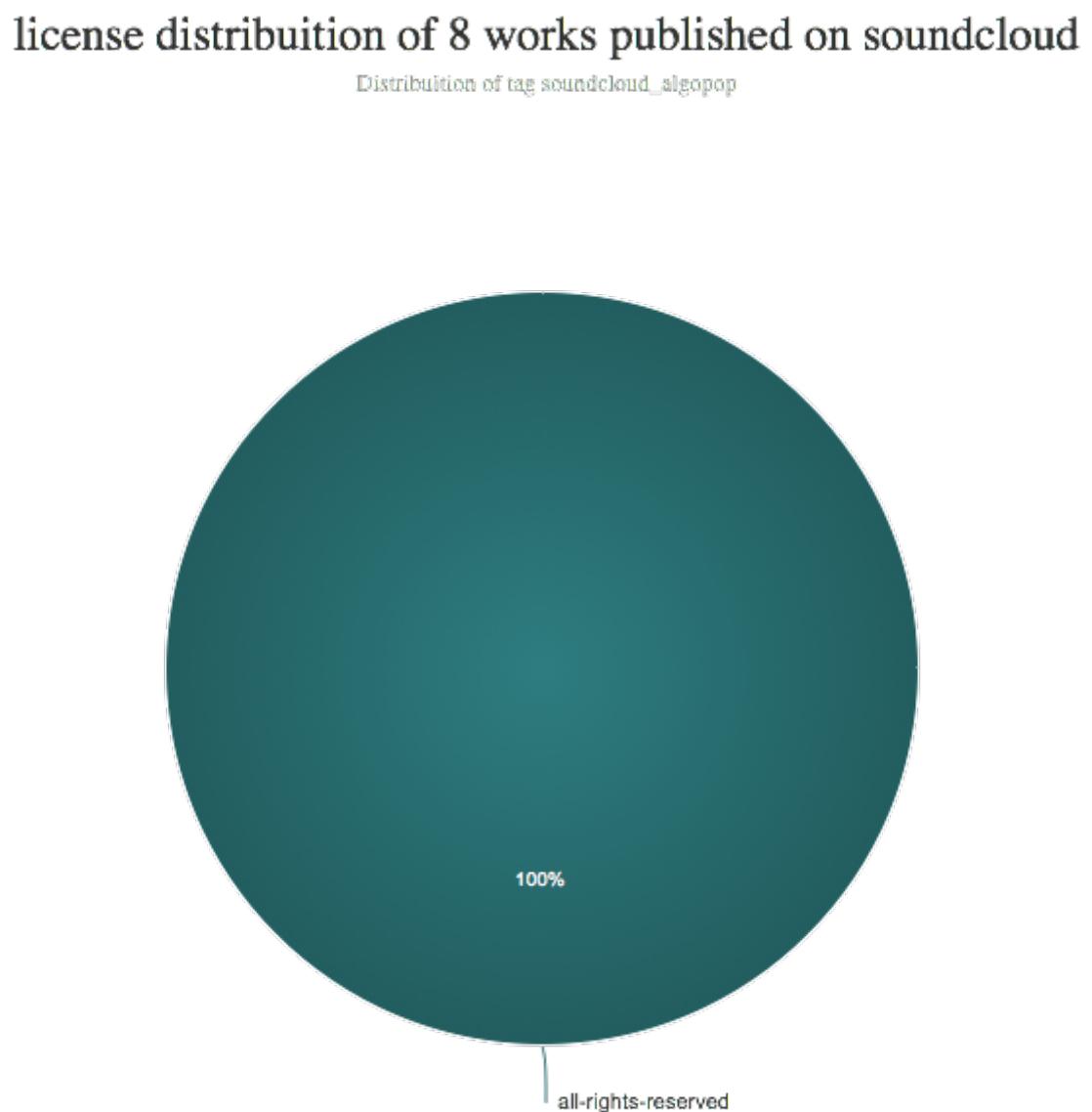


Figura 64 – Termo **algopop**, analisado a partir da categoria **license**.

created_at distribution of 1280 works published on soundcloud

Distribution of tag soundcloud_algorithmic_music

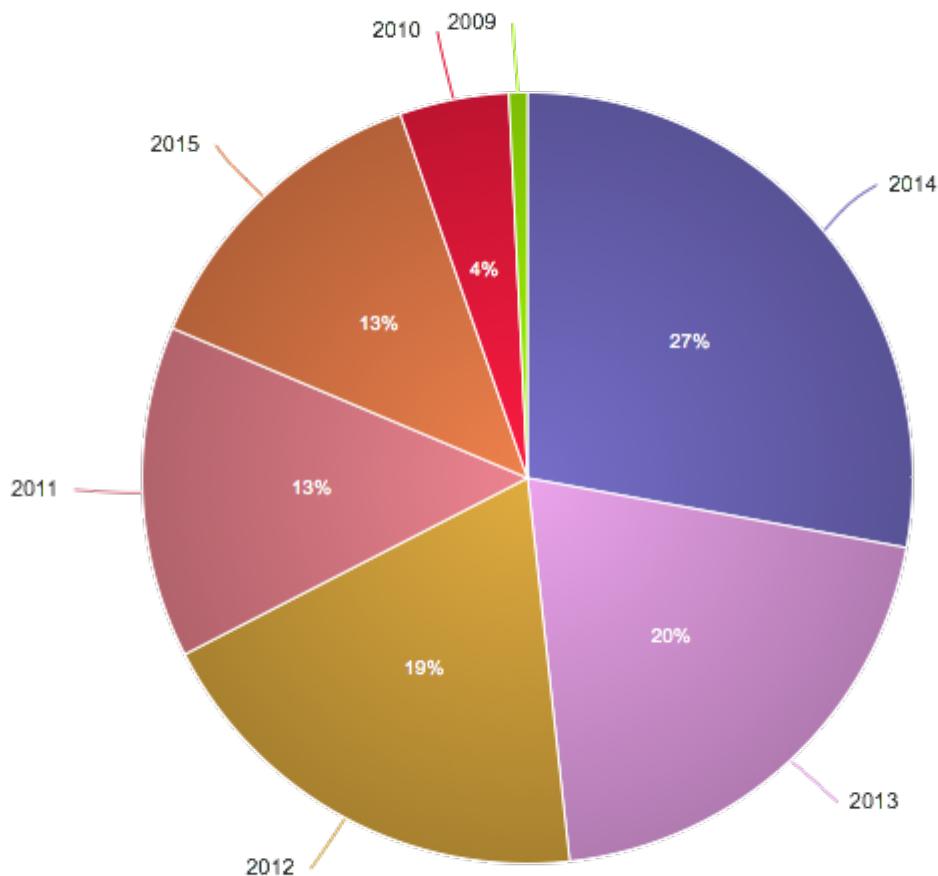


Figura 65 – Termo **algorithmic_music**, analisado a partir da categoria **created_at**.

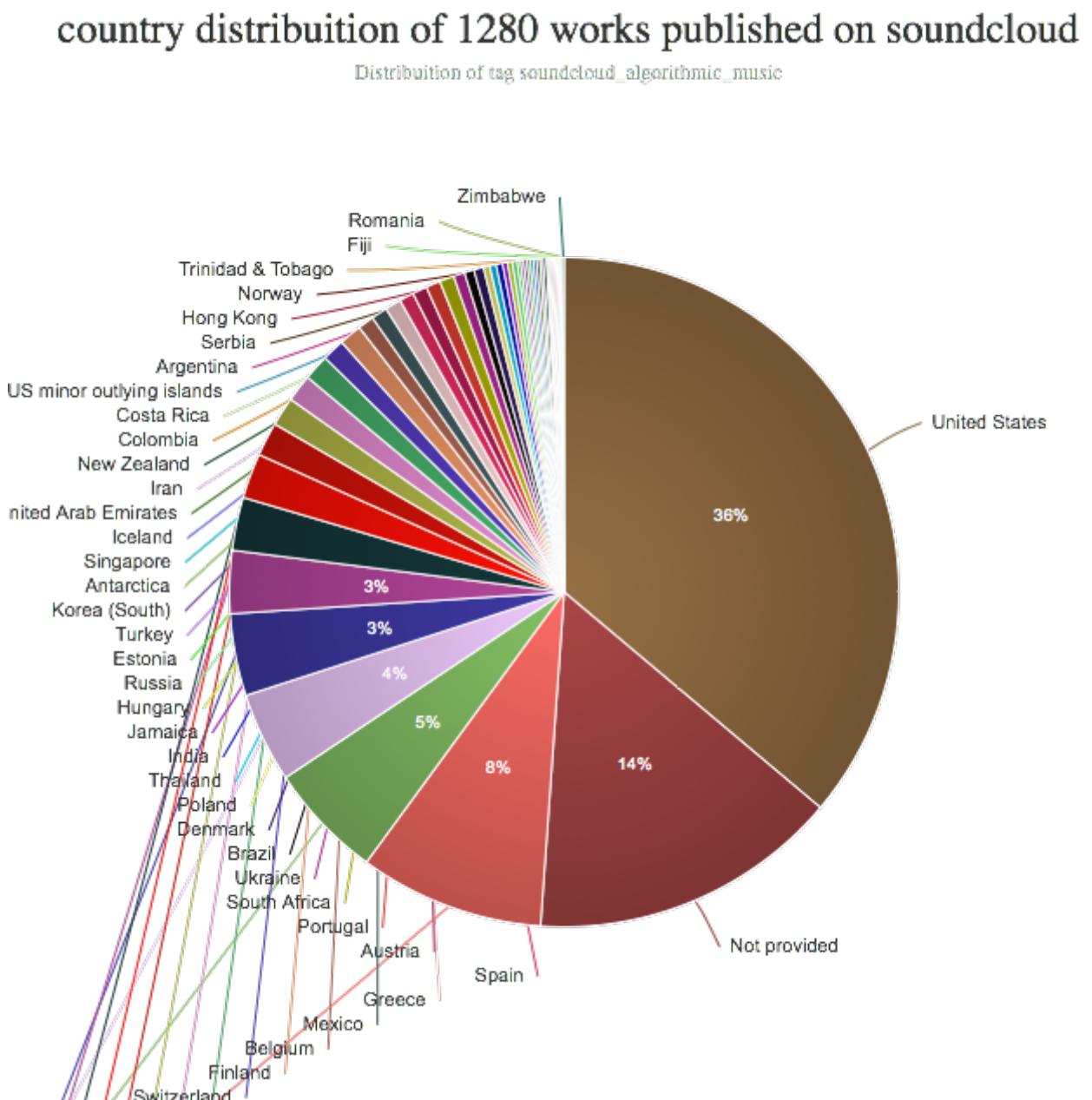


Figura 66 – Termo **algorithmic_music**, analisado a partir da categoria **country**.

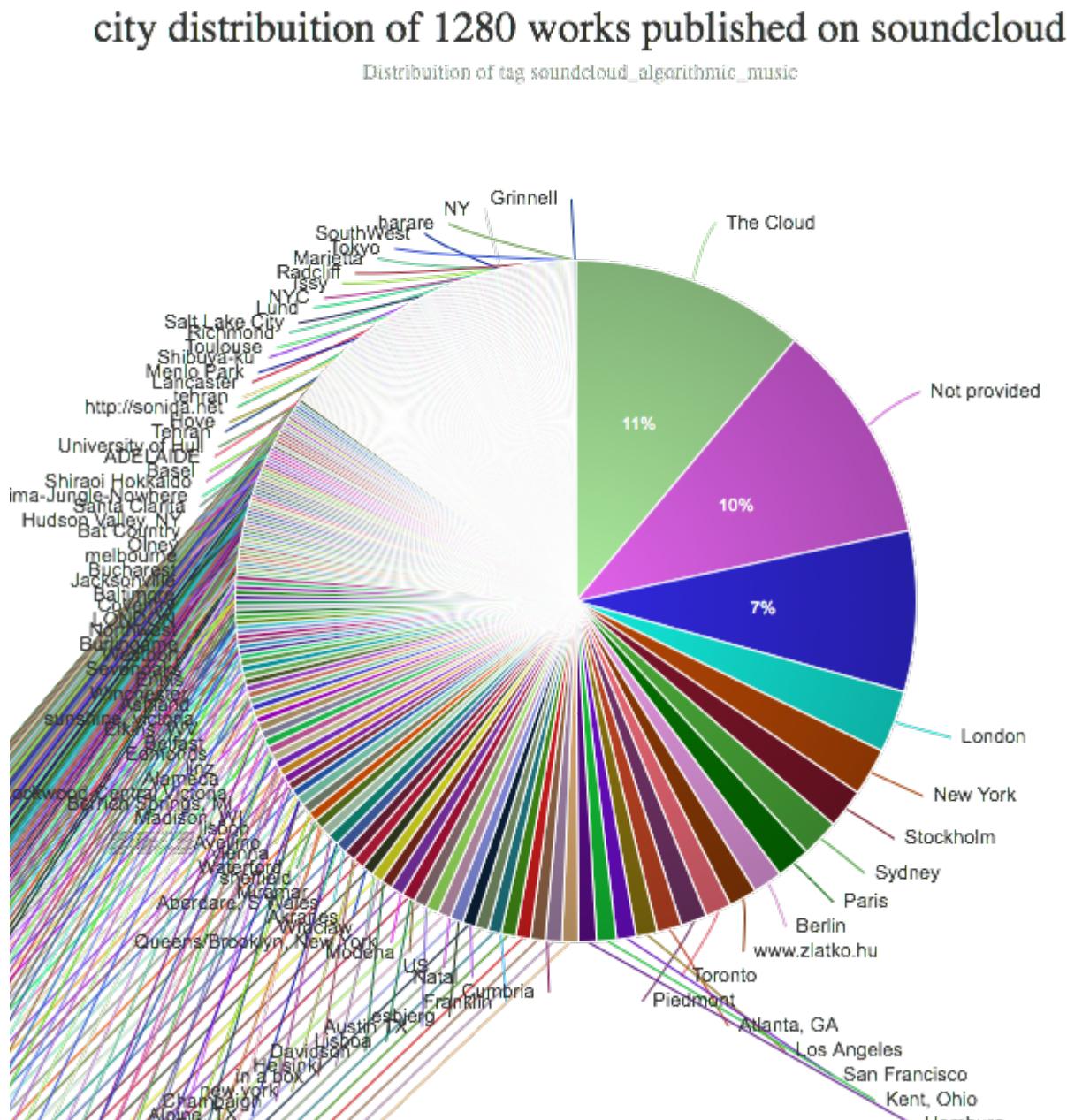


Figura 67 – Termo **algorithmic_music**, analisado a partir da categoria **city**.

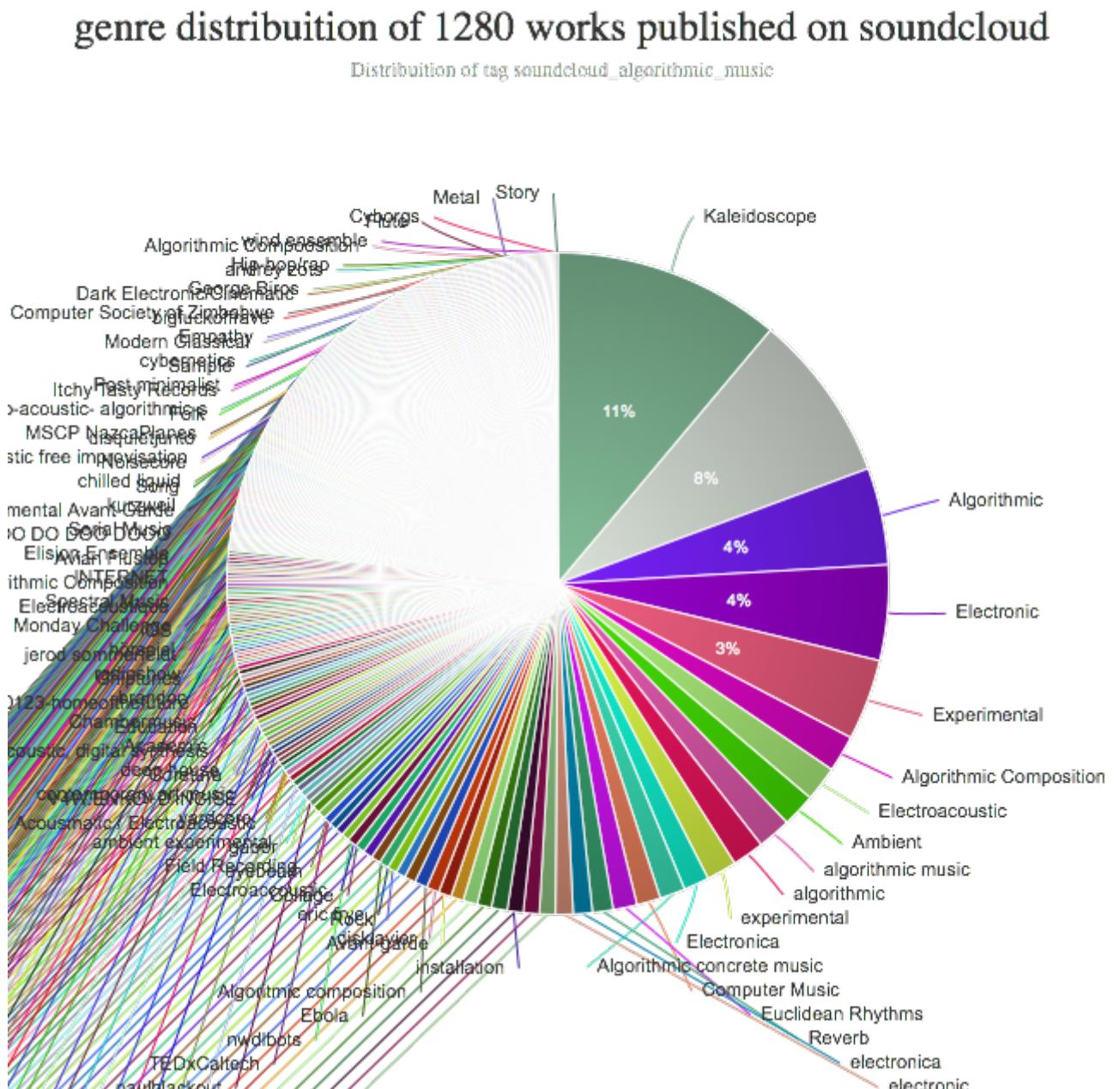


Figura 68 – Termo **algorithmic_music**, analisado a partir da categoria **genre**.

license distribuition of 1280 works published on soundcloud

Distribuition of tag soundcloud_algorithmic_music

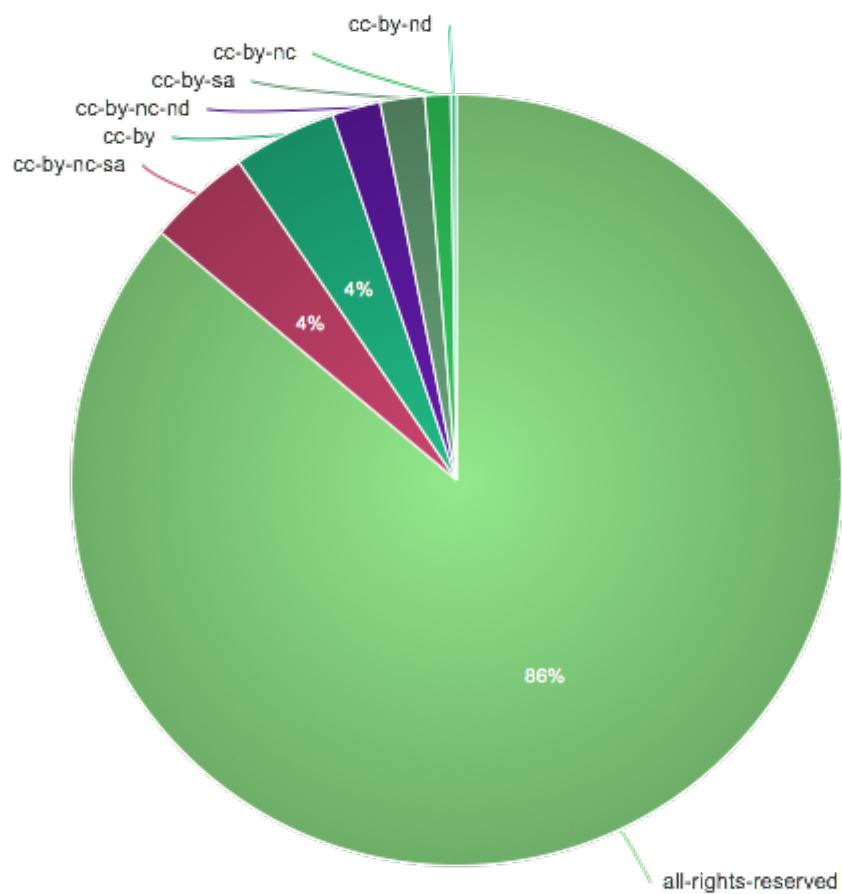


Figura 69 – Termo **algorithmic_music**, analisado a partir da categoria **license**.

created_at distribution of 117 works published on soundcloud
Distribution of tag soundcloud_bytebeat

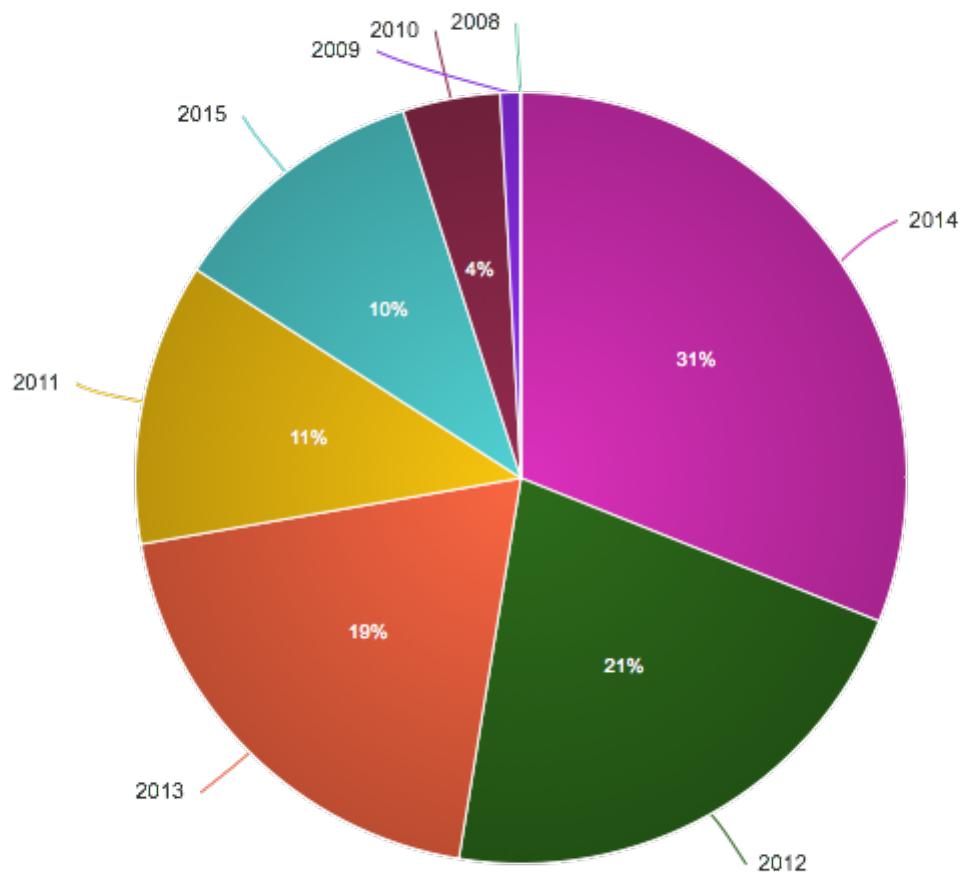


Figura 70 – Termo **bytebeat**, analisado a partir da categoria **created_at**.

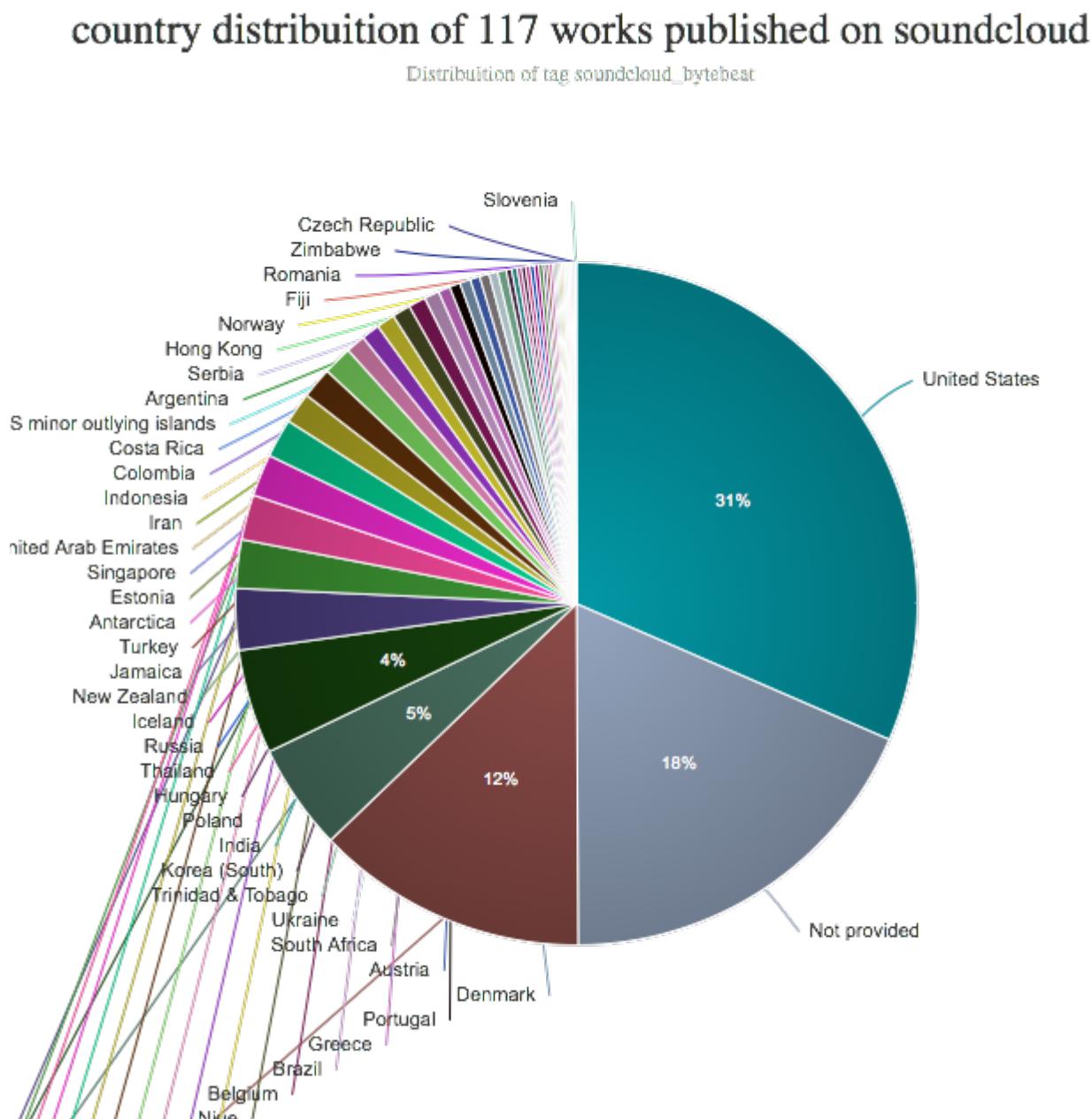


Figura 71 – Termo **bytebeat**, analisado a partir da categoria **country**.

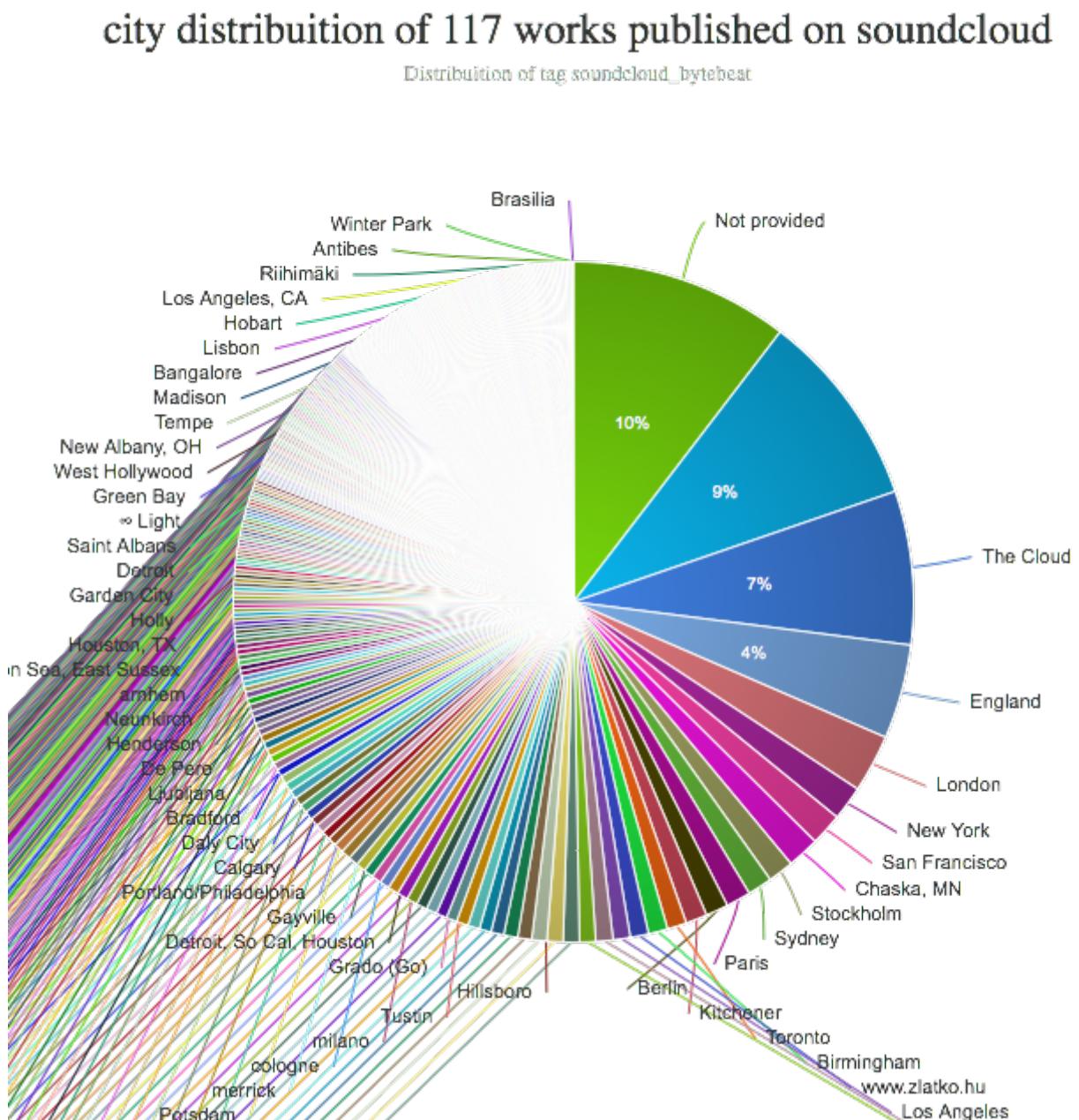


Figura 72 – Termo **bytebeat**, analisado a partir da categoria **city**.

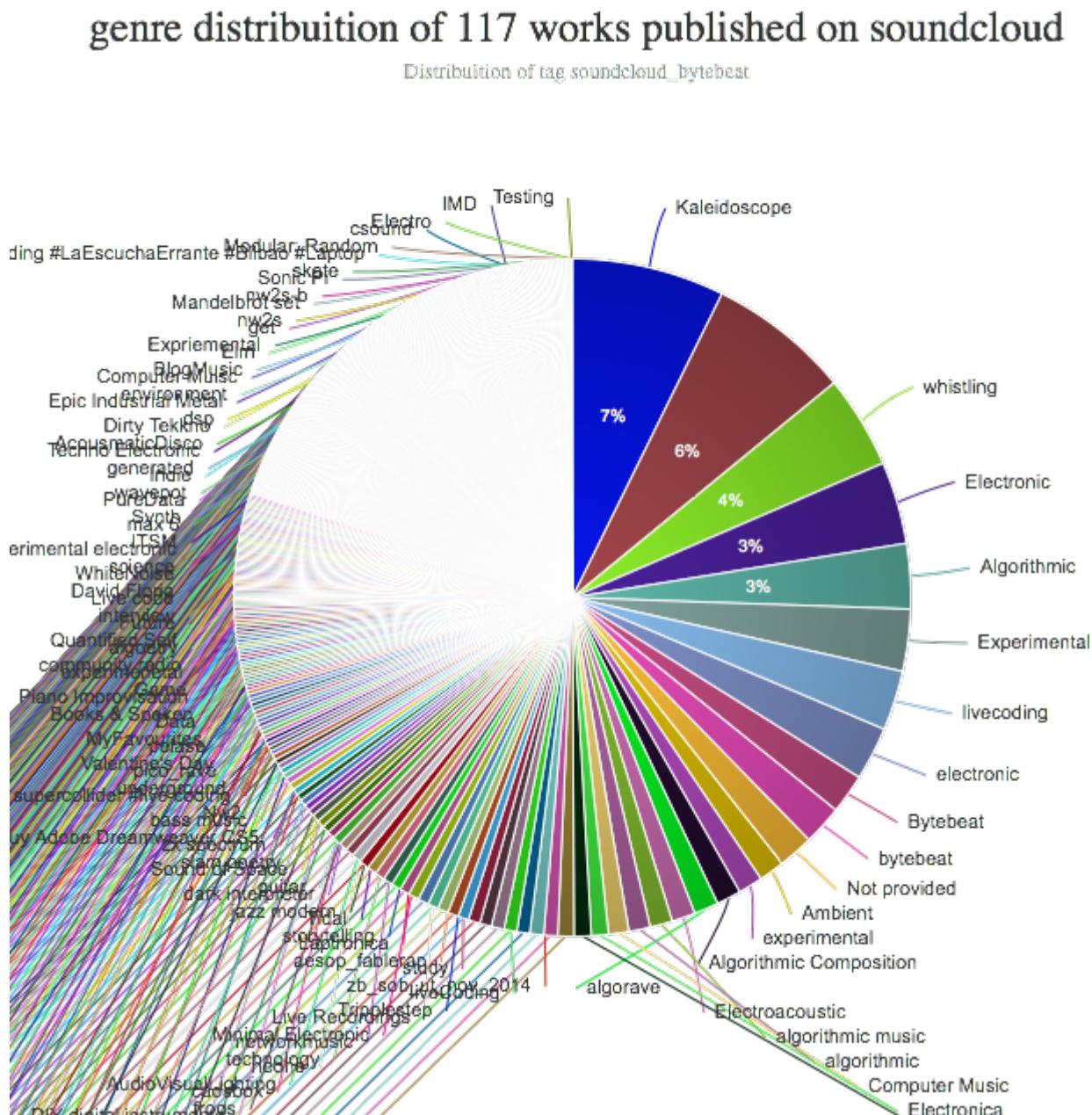


Figura 73 – Termo bytebeat, analisado a partir da categoria genre.

license distribution of 117 works published on soundcloud

Distribution of tag soundcloud_bytebeat

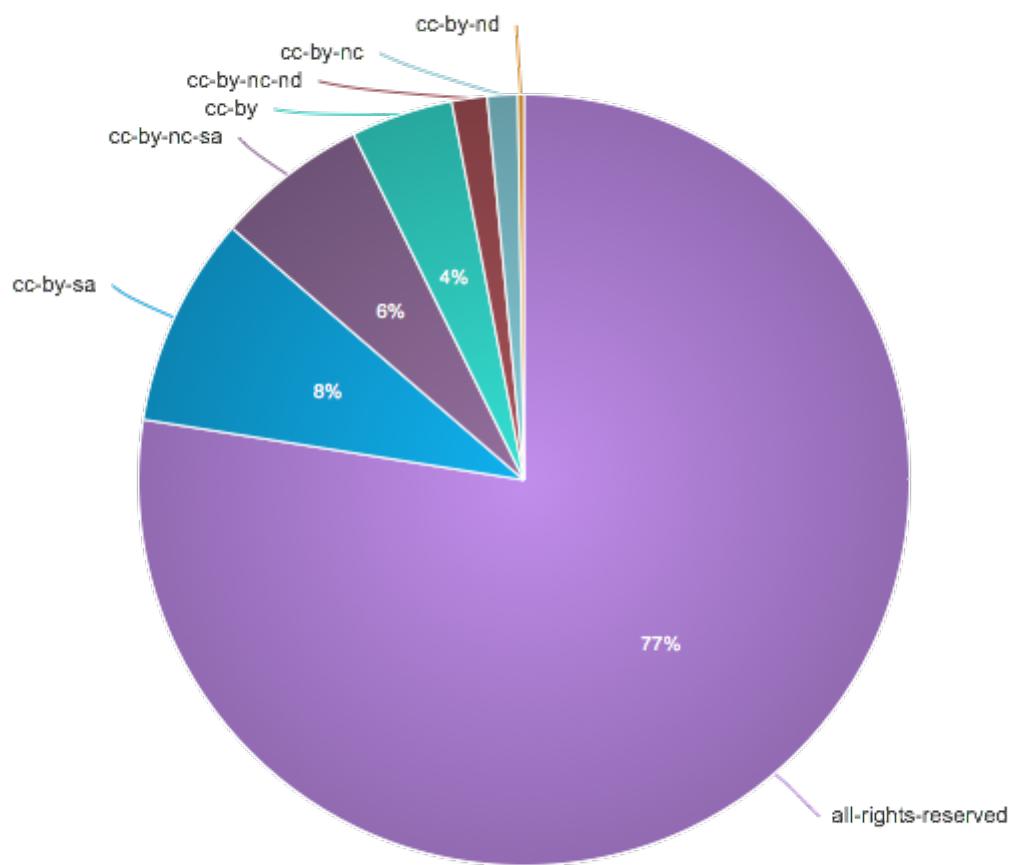


Figura 74 – Termo **bytebeat**, analisado a partir da categoria **license**.

created_at distribuition of 7 works published on soundcloud
Distribution of tag soundcloud_wavepot

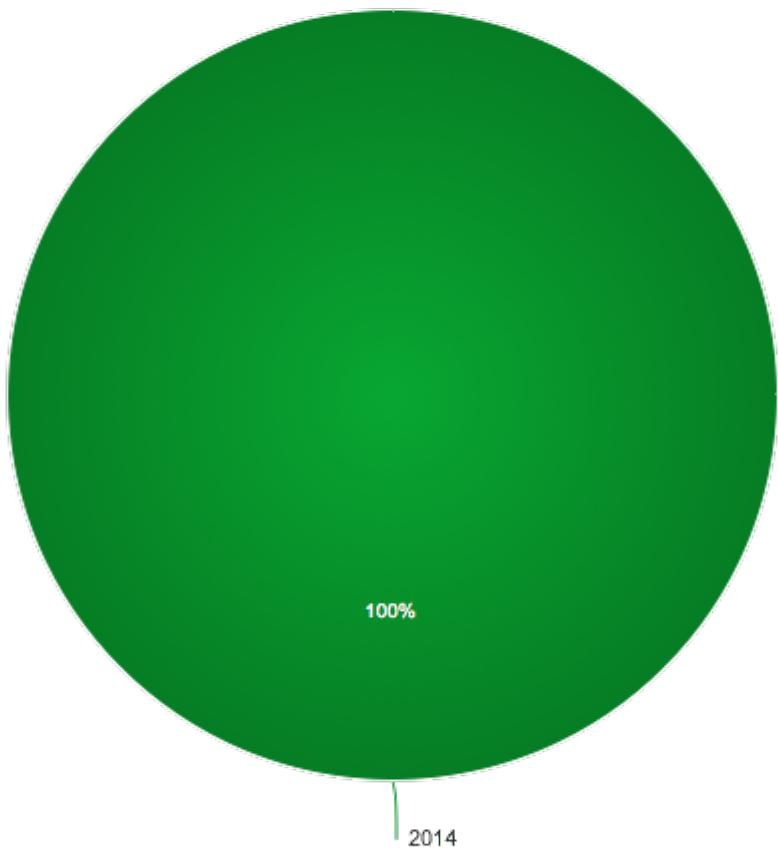
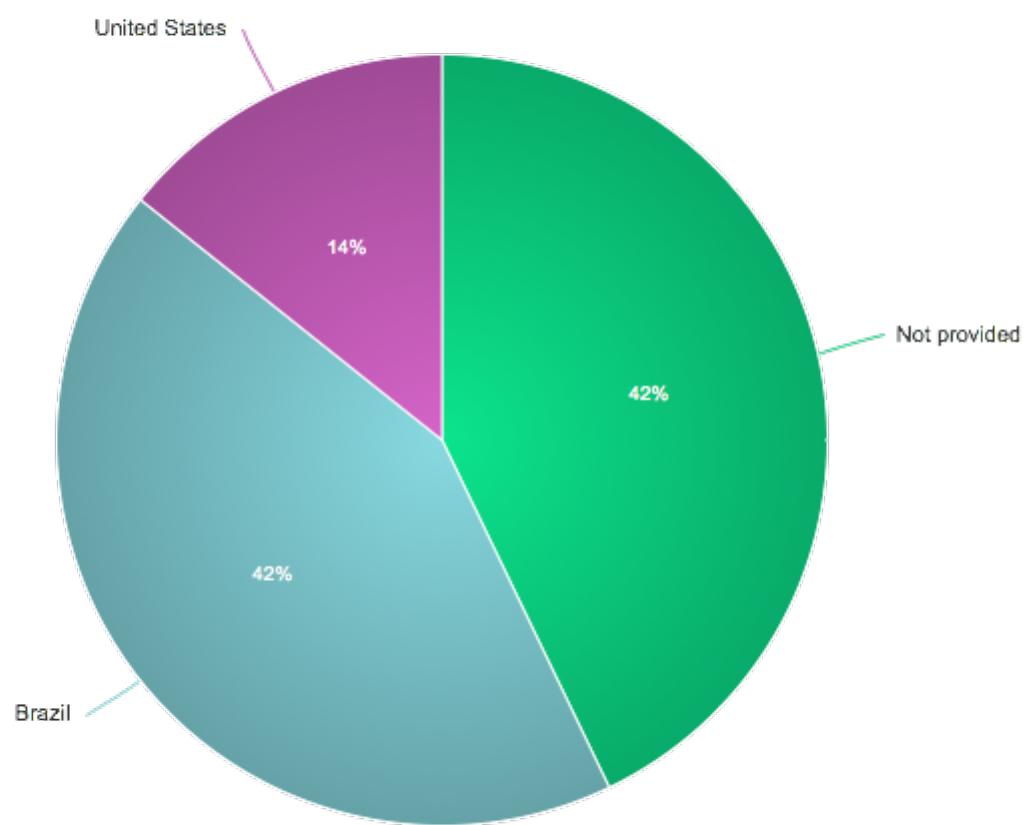


Figura 75 – Termo **wavepot**, analisado a partir da categoria **created_at**.

country distribution of 7 works published on soundcloud

Distribution of tag soundcloud_wavepot

Figura 76 – Termo **wavepot**, analisado a partir da categoria **country**.

city distribution of 7 works published on soundcloud

Distribution of tag soundcloud_wavepot

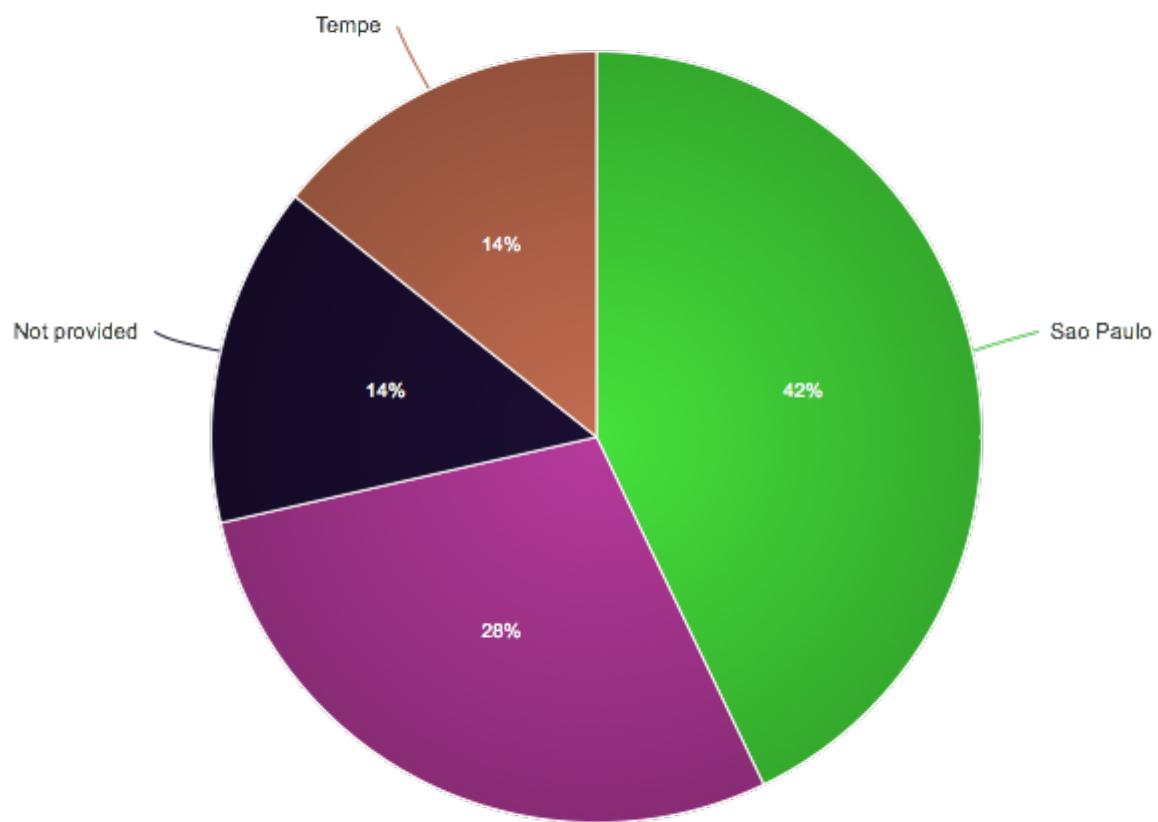


Figura 77 – Termo **wavepot**, analisado a partir da categoria **city**.

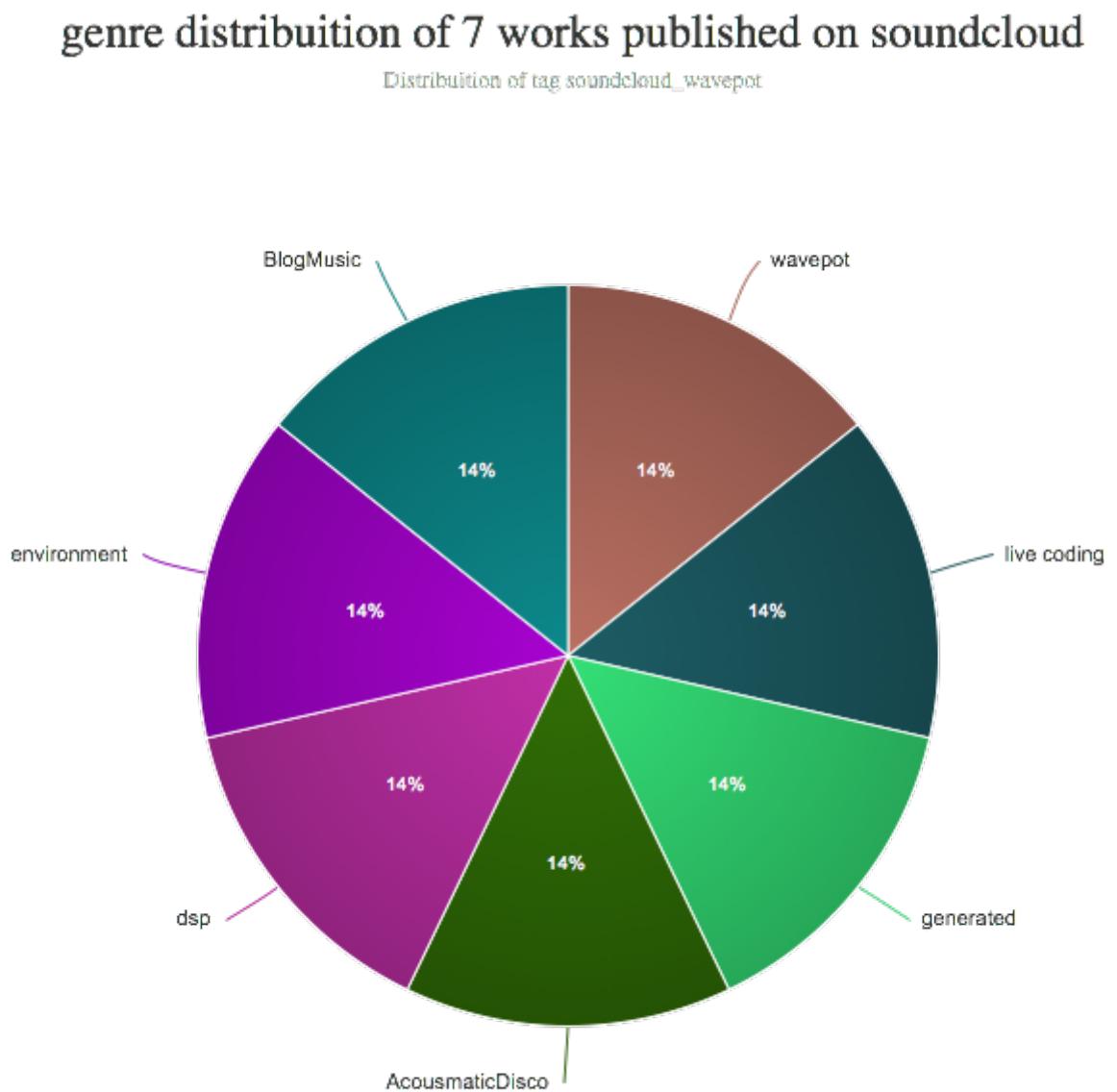


Figura 78 – Termo **wavepot**, analisado a partir da categoria **genre**.

license distribution of 7 works published on soundcloud

Distribution of tag soundcloud_wavepot

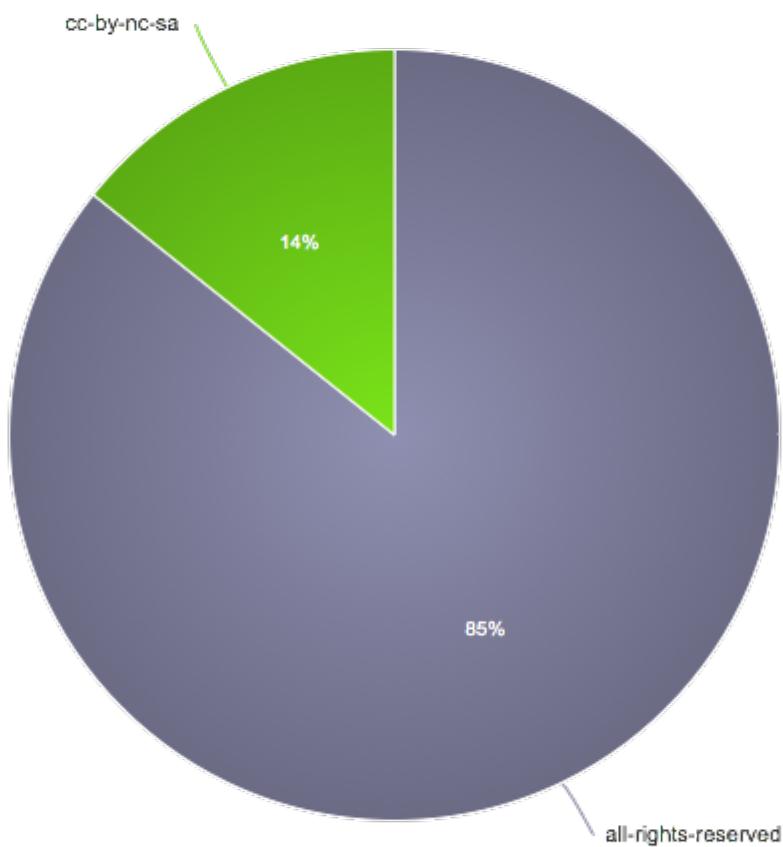


Figura 79 – Termo **wavepot**, analisado a partir da categoria **license**.