INSTITUTO DE
ARTES E DESIGN
IAD • UFJF

Termpot: criação, edição de funções no navegador em tempo de execução.

Guilherme Lunhani[1]

Instituto de Artes e Design – Universidade Federal de Juiz de Fora
Juiz de Fora, MG <gcravista@gmail.com>

31 de março de 2016

Sumário

Preâmbulo

O tema da improvisação de códigos, como um universo de conceitos, surgiu de uma experiência com um código em linguagem *python* (<https://www.python.org>), útil para gerar um mapa de termos, como ilustrado na figura abaixo.

Figura : Nuvem de palavras do McLean et al. (2015), 1^o Congresso Internacional de Live

Metáfora:
Livecoding como venda de filosofias → Mercado emergente do conhecimento das artes computacionais.

Metáfora:
Livecoding (improvisação de códigos) como venda de filosofias → Mercado emergente do conhecimento das artes computacionais.

Introdução

Abordagem do conhecimento das artes computacionais no *Livecoding*:

• Conhecimento ortopédico → Universo de conceitos, bricolagem

• Razão indolente → metáforas e imagens mentais de uma improvisação de códigos são reduzidos como um objeto com diversas propriedades.

• Pensamento abissal → divisão dos que praticam o *Livecoding* do ambiente europeu, e dos que não praticam fora do contexto europeu.

Introdução

Abordagem do conhecimento das artes computacionais no *Livecoding*: • Universo de conceitos

Objetivo

- Investigar um Universo de Conceitos sobre a *improvisação de códigos* (*live coding*);
- Investigar um Espaço de Conceitos, historicamente restrito, sobre a improvisação de códigos;
- Investigar um método de análise/criação para uma improvisação de códigos;
- Investigar um Espaço Conceitual de uma improvisação de códigos, *A Study in Keith* (2009) de Andrew Sorensen, e seu algoritmo musical em um ciclo de transformação.

Definições de base da Improvisação de códigos

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

Definições de base da Improvisação de códigos

A definição denota a aplicação em qualquer outra área, não apenas como metáfora, mas como estratégia de gerenciamento de fluxos criativos, assistidos por computador

Tecelagem

O grupo *Weaving codes* foi formado para a *(. . .) investigação de padrões a partir das perspectivas de tecelagem e música, e através do desenvolvimento de uma linguagem de computador e código para descrever a construção de tecidos* (GRIFFTHS, 2015b)

Tecelagem

Uma das idéias originais era combinar tecelagem e codificação em um cenário de atuação, ambos para fornecer uma forma de tornar a codificação ao vivo mais inclusiva com a tecelagem, e ao mesmo tempo esclarecer os processos de pensamentos digitais envolvolvidos na tecelagem (. . .) Nossa audiência consistiu de pesquisadores de artesanato, biólogos antropológicos, arquitetos, designers de jogos e tecnólogos – foi mais do que antecipamos! Alex e eu disponibilizamos alguns códigos de música do slub para tecer, e minha parte favorita foi projetar a tecelagem ao vivo (GRIFFTHS, 2015b).

Tecelagem

```
(twist 3 4 5 14 15 16)
(weave-forward 3)
(twist 4 15)
(weave-forward 1)
(twist 4 8 11 15)

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

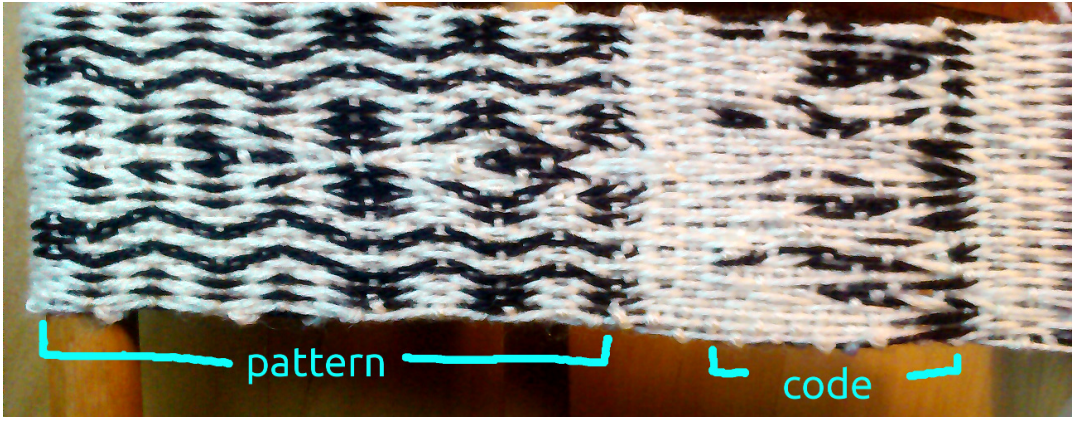


Figura : Tecido resultante da prática *Weaving code*. **Fonte**: Griffths (2015a).

Tecelagem

• Apresentação de dois importantes personagens para o *Livecoding*: Alex McLean e Dave Griffths (*Slub*)
• Observação de possíveis rituais de *livecoding*, como apresentação artística informal/formal.

Tecelagem

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

Slub

A banda *Slub* começou em 2000, como uma colaboração entre Adrian Ward e Alex McLean. A premissa do duo era utilizar a atividade de programação para realização de uma Música Eletrônica de Dança[1]. Sua primeira reunião foi em 2001, no *Paradiso club* em Amsterdã, durante o festival *Sonic Arts*. Em 2005 Griffths se juntou ao duo durante o festival *Sonar*, o que abriu espaço para o desenvolvimento de uma estética de videogames (MCLEAN, 2011, p. 138–140).

Live coding como dança

- Dança como fim de uma improvisação de códigos → Algorave
- Dança como meio (coreografia) de uma improvisação de códigos → Kate Sichio

Algorave

Algorave 'comecou como uma piada', de acordo com Alex McLean, um pesquisador de música computacional e um dos três de uma banda chamada Slub, que têm improvisado códigos por 13 anos. Ele veio com um termo enquanto conduzia uma gig em Nottingham com seu amigo Nick Collins (que tocava "datapop" sob o nome Sick Lincoln) no final de 2011. 'Nós sintonizamos em uma estação pirata tocando happy hardcore, e nós pensamos que seria bom programar alguma música rave.' Deste então, McLean organizou oito algoraves informais no mundo. (CHESIRE, 2013)

Algorave

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)

Algorave

Recorte histórico da Música Eletrônica para Dançar (COLLINS; McLean, 2014):
• 1992: Charles Ames disponibiliza o *Cybernetic Composer*;
• 1994: o duo *Koan*, formado pelos DJs Daniel Roeth e William Grey, realizam adaptações para entretenimento com base no *ambient music* de Brian Eno (1978);
• 1997: *Aphex Twin* (Richard David James) cria em o termo *live club algorithm*;
• 1999: o protocolo para edição audiovisual ao vivo *bbcut* (COLLINS; OLOFSSON, 2003) é incluído nos *opcodes* do *CSound*;
• 2000: o então duo *Slub* realizam performances, autodenominadas *generative techno*;
• 2001 é identificada a utilização de redes neurais para composição de padrões semelhantes ao *drum'n'bass*;
• 2004: é fundado o TOPLAP em uma casa noturna de Hamburgo.

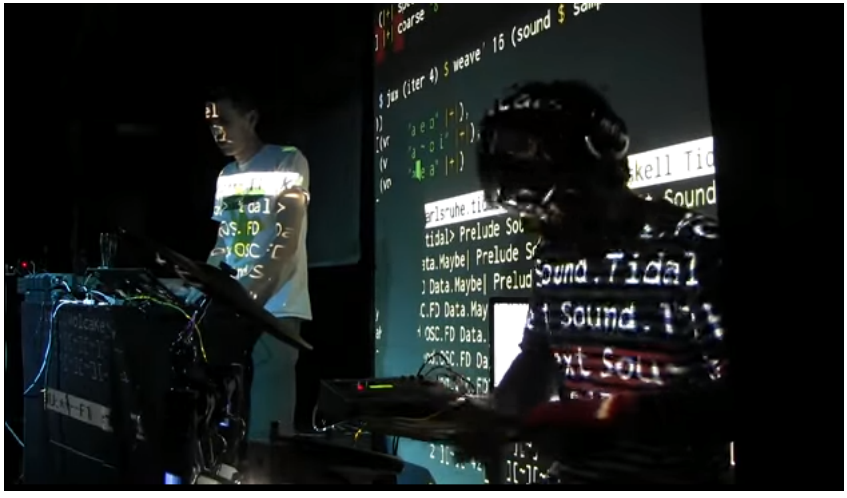


Figura : Performance do duo Canute (Karlesruhe, 2015) **Fonte**: Canute. . . (2015-27-01).

Algorave

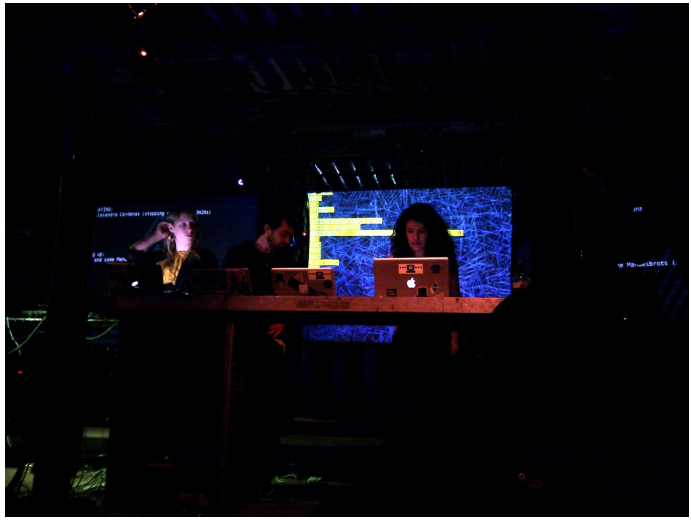


Figura : Performance de Alexandra Cárdenas (Londres, 2013) **Fonte**: Griffths (2013).

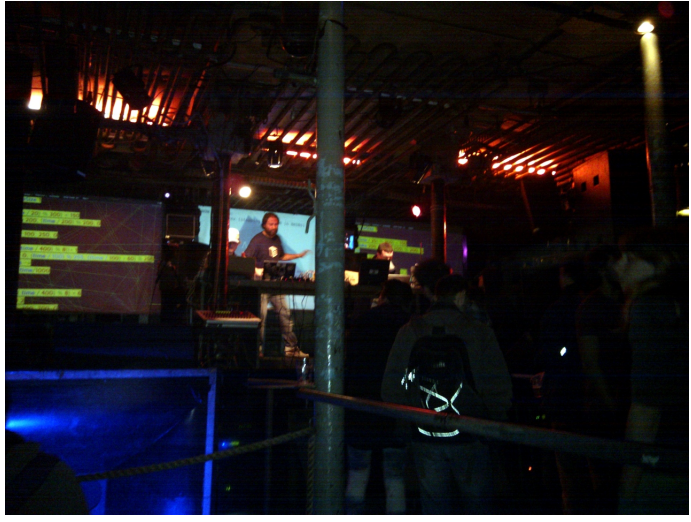


Figura : Performance do duo Mico Rex (Londres, 2013) **Fonte**: Griffths (2013).

Coreografia

Kate Sichio

- *Hacking the Body*/*Hacking Choreography*;
- *Hacking Coreography v.01* e *v.02*);
- *Hacking The Body 2.0*;

Coreografia

Para Downie (2005, cap. 1, p. 3), do ponto de vista computacional, a partitura coreográfica é mais próxima do código escrito em uma linguagem de computador do que a partitura musical tradicional.

Coreografia

• Assimilação do pensamento algorítmico na Dança → *Sensibilidades Computacionais* (DOWNIE, 2005 apud SICCHIO, 2014, cap. 1, p. 3;p. 31)
• *mecanismos de generalização e abstração, representação da coreografia e dança como computação* (DOWNIE, 2005, cap. 1, p. 2–4) → estratégias elaboradas por coreógrafos como Merce Cunningham, Trisha Brown, Bill T. Jones, e William Forsythe.

Coreografia

Esta sensibilidade computacional é presente em dois níveis em um trabalho destes coreógrafos. Primeiramente, em seus processos coreográficos – os sistemas, métodos, e notação através dos quais os coreógrafos criam a dança. Segundo, no trabalho ele mesmo, finalizado, que aparece no palco e é interpretado pelo observador. As primeiras invenções e proclamoções de Cunningham – a democracia do espaço do palco, e a redescoberta do que está atrás do dançarino como ponto de origem do movimento – pode ser interpretado como generalizações do tipo; qualquer ponto do palco é a "frente", e conectado por um conjunto de articulações pode ser pensado como um membro. O que eram constantes, uma vez especificados em uma descrição rígida, se tornam variáveis em uma estrutura generativa.(DOWNIE, 2005, cap. 1, p. 2–4)

Hacking Coreography

Duas experiências com uma Partitura de Eventos do artista Alison Knowles (mais especificamente a peça de performance #8, de 1965)

Hacking Coreography

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

* "alguma coisa" com "tudo"

* "alguma coisa" com "nada"

* "alguma coisa" com "alguma coisa"

* "tudo" com "tudo"

* "tudo" com "nada"

* "nada" com "nada"

Depois que a partitura foi completada, contudo, ela foi hackeada. Isso significa que o executante tenta de alguma forma contornar as instruções originais. Isto foi feito sem preparações prévias e a audiência assistiu isso se desdobrar enquanto era realizada. Nesta primeira performance, o papel e os rótulos foram rasgados para criar novas palavras e categorias (. . .) Então ao invés de "nada"[*Nothing*]*, foram formados dois grupos, "não"*[*No*] *e "coisa"*[*Thing*]. (SICCHIO, 2014, p. 31)

Orientações são escritas como um híbrido de texto discursivo, legível por um executante, e de código de computador em linguagem Java. Isto é, ele não é executável por um computador para resultar em sons, mas por um humano para resultar em movimentos.

Orientações são escritas como um híbrido de texto discursivo, legível por um executante, e de código de computador em linguagem Java. Isto é, ele não é executável por um computador para resultar em sons, mas por um humano para resultar em movimentos.

Hacking Coreography v.02 I

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

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

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

Hacking Coreography v.02 II

```
run(){
move1
move4
move4
move1
move2
move3
move1
move2
move3
move4
}

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

Hacking The Body I

Esta peça é uma exploração de eletrônica codificada ao vivo e movimentos improvisados. Uma dançarina veste uma peça de atuadores hápticos. Estes atuadores são programados em tempo-real via OSC[2] para 'zunir' sobre os lados direito e esquerdo da dançarina para indicar qual lado do corpo a dançarina deve mover. A partitura é codificada ao vivo pela coreógrafa enquanto a dançarina responde por uma retroalimentação háptica. Esta peça explora o live coding de corpos, e movimento como saída, ao invés de saídas sonoras ou visuais como encontrado em muitas execuções de live coding

Hacking The Body II

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

Música computacional

- FooBarBaz (Aspecto histórico, em um evento nacional)
- Magno Calliman (Aspecto performático, em um evento nacional)
- Supercopair (Aspecto telemático, com pesquisadores brasileiros)

Foo Bar Baz I

FooBarBaz é um grupo de improvisação de códigos formado por Gilson Beck, Renato Fabbri, Ricardo Fabbri e Vilson Vieira. Sua primeira apresentação foi durante o Festival Contato 2011. Os membros são ativos em um laboratório virtual conhecido como *labMacambira*(<http://labmacambira.sourceforge.net/>).

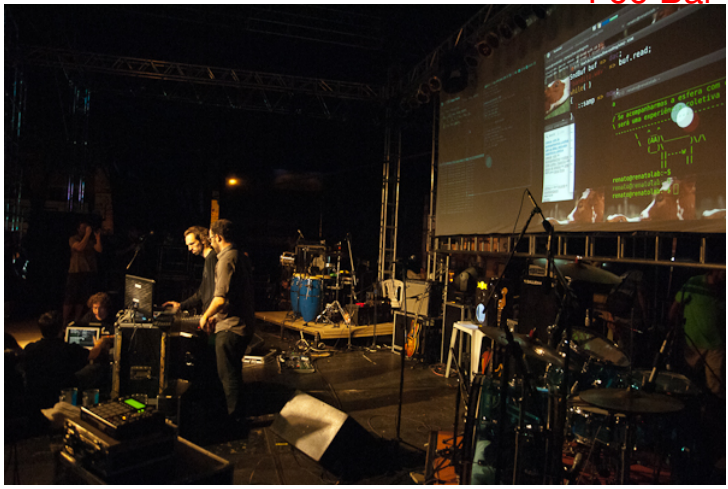


Figura : Festival Contato, 2011. **Fonte**: Fabbri e Vieira (2011).

Foo Bar Baz

```
["samples/fx/s20.wav"] => Foo.name;
[0.] => Foo.prop;
[.25, .15] => Foo.rate;
[2., 1., 1., 4.] => Foo.du;
[.8] => Foo.gain;
```

Funcionamento → *grade temporal* (TimeGrid) → Graham Coleman (<http://www.dtic.upf.edu/~gcoleman/>).

```
//basic timing operations abbreviated

public class TimeGrid {

    dur beat;
    dur meas;
    dur sect;

    int nbeat;
    int nmeas;

    //phase and magnitude of offset
    float measPhase;
    dur measOffset;
    ...

    //sync to beat
    fun void sync() {
        beat - (now % beat) => now;
    }

    fun void sync(dur T) {
```

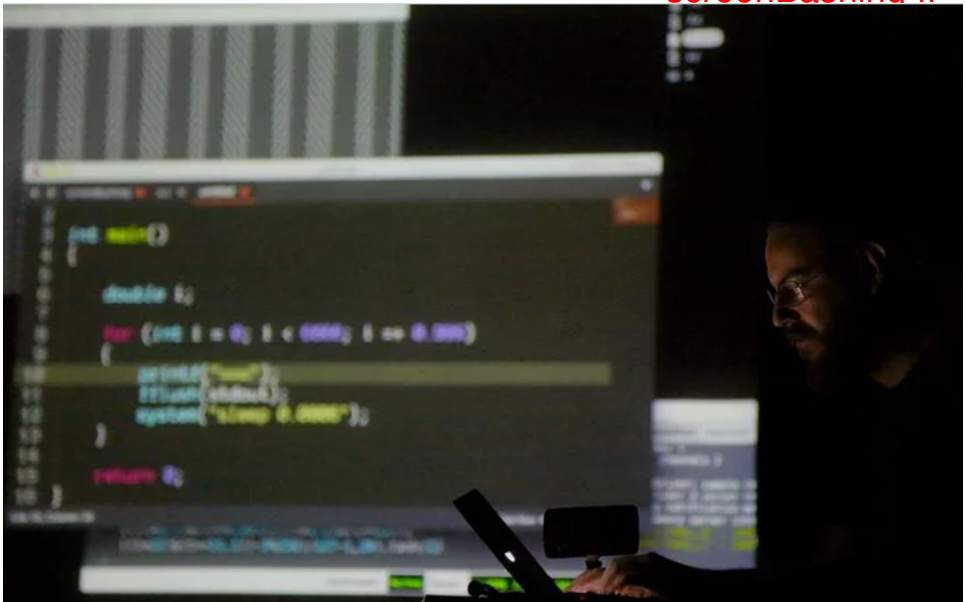
Foo Bar Baz II

```
        T - (now % T) => now;
    }

    //how long to sync to this duration
    fun dur syncDur(dur T) {
        return (T - (now % T));
    }

    //minimum time
    fun dur tmin(dur a, dur b) {
        return (a < b) ? a : b;
    }
    ...
}
```

screenBashing I

Performance de *screenBashing* de Magno Caliman (ver 9), realizada durante o XIII ENCUN[3].

screenBashing II



screenBashing

Não acho totalmente adequado considerar como puramente um improviso. (. . .) acredito que se imaginarmos um continuum, uma linha onde de um lado vc tem uma improvisação totalmente livre (impossível de se alcançar, claro) e do outro uma composição 100% determinada (tão impossível quanto), acredito que screenBashing está posicionada mais à direita. . .

screenBashing I

Código-fonte:

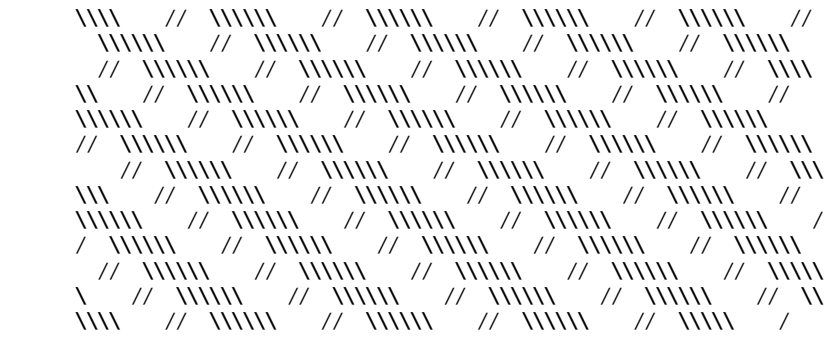
```
# include <stdio.h>

int main()
{
  double i;

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

Execução:

screenBashing II

```
# gcc - compilador
# a.c - arquivo em linguagem c
# -o  - escreve o resultado em um outro arquivo
# a   - arquivo binario alvo
\$> gcc a.c -o a
magno.c: In function 'main':
magno.c:7:12: error: conflicting types for 'i'
```

Resultado:



screenBashing

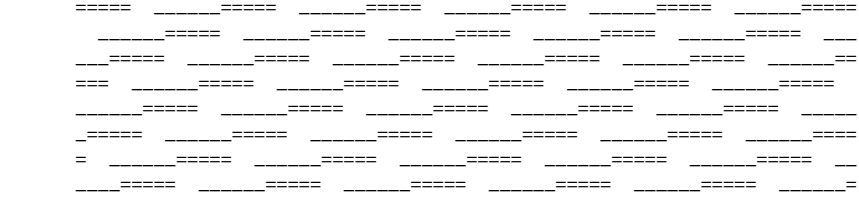
- O código sonoro foi pouco exposto
- Valorização ideológica. → repetição dos algarismos seis e nove → *black metal* abstrato (?)
- uma segunda notação (Capítulo 3), mais enxuta e sem erros, permite apontar a questão ideológica

screenBashing

Código-fonte:

```
# include <stdio.h>
int main(){
  int i=0;
  for (;;i++){
      printf("=====  ______");
      fflush(stdout);
      // 30 frames por segundo
      // ou um limite aproximado da
      // pecepcao humana (1/30)
      system("sleep 0.033");
  }
}
```

Resultado:



Telepresença e espaços virtuais

• Uma rede local, com computadores diferentes, mas com os improvisadores fisicamente próximos;
• Uma rede remota, privada, que comunica um conjunto de pessoas fisicamente distantes Junior, Lee e Essl (2015, p. 152–153)
• O navegador de *internet* se torna o ambiente virtual de criação musical (ROBERTS; WAKEFIELD; WRIGHT, 2013).

Telepresença e espaços virtuais

live coding sessions → sessões de improvisações que ocorrem em encontros, simpósios e *workshops*, ricamente documentadas (<https://supercollider.github.io/archive>).

Telepresença e espaços virtuais

Performance telepresencial em 2014, por Ben Swift, Henry Gardner e Andrew Sorensen, realizada entre dois intérpretes-programadores localizados na Alemanha e Estados Unidos usando um servidor SSH localizado na Australia (Junior; Lee; Essl, 2015, p. 152–153)

Telepresença e espaços virtuais

Supercopair (<https://github.com/deusanyjunior/atom-supercopair>) → ambiente cooperativo de improvisação de códigos.

Telepresença e espaços virtuais

Execuções remotas em navegadores de *internet*, com auxílio da biblioteca *WebAudio API* (<https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html>).

Telepresença e espaços virtuais

- *Gibber* (ROBERTS; KUCHERA-MORIN, 2012) (<http://gibber.mat.ucsb.edu/>);
- *Wavepot*(<http://www.wavepot.com>);
- *Vivace* (VIEIRA et al., 2015) (<http://void.cc/freakcoding>)
- *Termpot* (LUNHANI; SCHIAVONI, 2015) (<http://jahpd.github.io/termpot>)

Definições Históricas da Improvisação de códigos

Exemplos proto-históricos → possuem similaridades com o conjunto de regras práticas publicadas por (WARD et al., 2004)

Definições Históricas da Improvisação de códigos

- Pietro Grossi
- Baía de São Franscisco

Definições Históricas da Improvisação de códigos

Evento Histórico: Live Algorithm Programming and Temporary Organization for its Promotion

Pietro Grossi – Computer Concerto

Grossi começou a se interessar por música computacional durante a primeira metade doas anos 60, quando ele quando ele organizou um programa de rádio centrado em torno da "música inovadora"em geral (GIOMI; LIGABUE, 1999). Contudo a primeira experiência de Grossi com um computador foi em Milão, no Centro de Pesquisa Elétrica da Olivetti-General. Aqui, auxiliado por alguns técnicos internos e engenheiros, ele conseguiu compor e gravar alguns de seus primeiros trabalhos em música computacional. Eles foram, em sua maior parte, transcrições de música clássica ocidental. Contudo, houve algumas exceções, por exemplo, uma faixa chamada Mixed Paganini (MORI, 2015b, p. 126)

Pietro Grossi – DCMP (*Digital Computer Music Program*)

Um dos mais importantes aspectos do trabalho de Grossi foi que todas intervenções eram instantâneas: o operador não tinha que esperar pelo computador terminar todas operações requisitadas, e depois ouvir os resultados. Cálculos de dados e reprodução sonoras eram simultâneos. ***Esta simultaneidade não era comum no campo da Computer Music daquele tempo, e Grossi deliberadamente escolheu trabalhar desta forma, perdendo muito no lado da qualidade sonora. Seu desejo era poder escutar os sons resultantes imediatamente*** (MORI, 2015b, p. 126)

Pietro Grossi – DCMP (*Digital Computer Music Program*)

“preguiçoso” (*pigro*, *lazy*) $\rightarrow$ *reflexividade*

Pietro Grossi – DCMP (*Digital Computer Music Program*)

habilidade de um programa manipular como dados algo que representa o estado do programa durante sua própria execução, o mecanismo para codificação de estados de execução é chamado reificação.(MALENFANT; JACQUES; DEMERS, 1996, p. 1).

Pietro Grossi – DCMP (*Digital Computer Music Program*)

Naquele tempo, os recursos de cálculo eram escassos e, para obter a reprodução em tempo-real citada, era necessário pedir por pouca quantidade de dados. (. . .) A síntese de timbres necessita de uma quantidade imensa de dados, e então a escolha foi descartá-la temporariamente, e todos os sons eram reproduzidos com o timbre de uma onda quadrada. Esta forma de onda era gerada por extração do estado binário do pin de saída da placa mãe que controla o programa. Essa saída tinha um único bit, e então a onda sonora gerada era o resultado desta mudança do estado binário. Desta forma, o computador não emprega quaisquer recursos para calcular a síntese sonora, economizando-os para o processo de produção musical. Grossi não estava interessado na qualidade da saída sonora em sua primeira fase em Pisa. O que importava particularmente era a capacidade em trabalhar em tempo real, ou, em outras palavras, para ter a escolha de escutar imediatamente ao que ele escreveu no teclado do terminal de vídeo (GIOMI; LIGABUE, 1999 apud MORI, 2015b).

Pietro Grossi – TAU2-TAUMUS (*Terminale Audio 2^{a} versione)*

Grossi foi capaz de implementar melhorias de timbre, digitalmente controladas por doze vozes, com síntese analógica.

Estas doze vozes eram divididas em três grupos, compostos de quatro canais cada. O operador poderia atribuir um timbre diferente para cada grupo, que era modulado usando síntese aditiva com sete sobretons. Cada sobretom era controlado individualmente pelo programa.

Pietro Grossi – TELETAU

O TAU2-TAUMUS sofreu uma considerável modificação, sendo que era possível controlar o sistema digital-analógico remotamente. O novo programa foi batizado de TELETAU Mori (2015b, p. 128–129)

Pietro Grossi – TELETAU

Grossi fez sua primeira experiência do tipo durante uma conferência de tecnologia em Rimini em 1970, onde o músico reproduzia algumas de suas composições, bem como sons randômicos, empregando um terminal de vídeo conectado pelo telefone para o computador da CNR em Pisa. A RAI, empresa de radiodifusão italiana, emprestou suas pontes de rádio [*Comunicação entre duas antenas*] *para enviar sinais sonoros entre Pisa e Rimini. É como se fosse o primeiro experimento de telemática musical no mundo* (MORI, 2015b, p. 129).

Baía de São Franscisco

- The League of Automatic Composers
- The Hub
- Ron Kuivilla

Baía de São Franscisco

Mills College em Oakland.

Baía de São Franscisco

Com o florescimento da indústria de computadores pessoais na Baía de São Franscisco, o acesso às novas tecnologias e pessoas que desenvolveram elas era talvez o melhor no mundo. (. . .) Esta também é a cultura que deu ao mundo a música "New Age", uma versão aguada e comercializada das músicas com base em modos e drones que Terry Riley, Pauline Oliveros, e LaMonte Young inventaram durante os anos cinquenta e sessenta. Mas a música feita na Costa Oeste também incluiam improvisações barulhentas e livre de restrições, que sobraram das revoluções contra-culturais dos anos 60(BROWN; BISCHOF, 2002).

The League of Automatic Composers

• Segunda metade da década de setenta: Jim Horton começa a adquirir micro-controladores KIM-1 (*Keyboard Input Monitor*, <http://www.6502.org/trainers/buildkim/kim.htm>), com interesses musicais.
• David Behrman e John Bischoff, Rich Gold, Cathy Morton, Paul Robinson, e Paul Kalbach.

KIM1

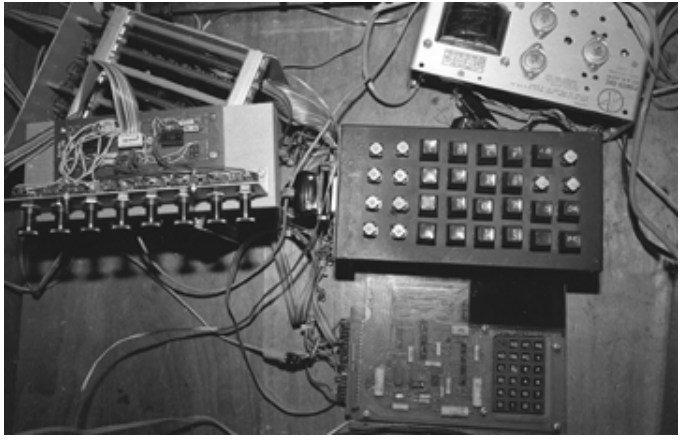


Figura : Sistema de música computacional de John Bischof *circa* 1980. Foto: Eva Shoshanny[5]. **Fonte**: Brown e Bischof (2002).

The League of Automatic Composers

Na primavera de 1979, montamos uma série quinzenal regular de apresentações informais sob os auspícios da *Bay Center for the Performing Arts*. Todos outros domingos à tarde passávamos algumas horas configurando nossa rede de KIMs na sala *Finnish Hall*, na Berkeley, e deixávamos a rede tocando, com retoques aqui e ali, por uma ou duas horas. Os membros da audiência poderiam ir e vir como quisessem, fazer perguntas, ou simplesmente sentar e ouvir. Este foi um evento comunitário de tipos como outros compositores aparecendo, tocando ou compartilhando circuitos eletrônicos que tinham projetado e construído. Um interesse na construção de instrumentos eletrônicos de todos os tipos parecia estar "no ar". Os eventos da sala *Finn Hall* foram feitos para uma cena com paisagens sonoras geradas por computador misturado com os sons de grupos de dança folclórica ensaiando no andar de cima e as reuniões ocasionais do Partido Comunista na sala de trás do edifício velho venerável. A série durou cerca de 5 meses que eu me lembre.(BROWN; BISCHOF, 2002)

The League of Automatic Composers

- Tim Perkis
- Sam Ashley, Kenneth Atchley, Ben Azarm, Barbara Golden, Jay Cloidt e Brian Reinbolt.

The League of Automatic Composers

“rede de composições”

The League of Automatic Composers

Os membros da liga geralmente adaptavam composições solo para usar dentro da banda. Estes solos eram desenvolvidos independentemente por cada compositor, e eram tipicamente baseados em esquemas de algoritmos de um tipo ou outro. Existiam características de improvisação diferentes para muitas delas, como bem as músicas eram diferentes em detalhes. Teorias matemáticas, sistemas de afinação experimentais, algoritmos de inteligência artificial, projetos de instrumentos de improvisação, e performance interativa eram algumas das áreas exploradas nestes trabalhos (. . .) Os solos tocavam simultaneamente no cenário de grupo, se tornando "sub"-composições que interagem, cada uma enviando e recebendo dados pertinentes para o funcionamento musical. (BROWN; BISCHOF, 2002, 12^{o} parágrafo).

The Hub

1986:

- Bischoff e Perkis, após a saída de Horton
- Chris Brown e Mark Trayle
- *THE NETWORK MUSE - Automatic Music Band Festival*.
- Scott Greham-Lancaster e Richard Zvonar
- Phil Burk, Larry Polansky e Phil Stone.

The Hub

• Nome simbólico de um sistema musical centralizado • *(. . .) um pequeno microcontrolador como caixa de correio, para postar dados usados no controle de seus sitemas individuais, que eram então acessados por outro intérprete, para usar de qualquer maneira e em qualquer tempo que escolher.*

The Hub

The Hub originalmente surgiu como uma maneira de limpar uma bagunça. (. . .) Toda vez que nós ensaiamos, um conjunto complicado de conexões ad-hoc entre computadores tinham de ser feitas. Isso criou um sistema com um comportamento rico e variado, mas sujeito a falhas, e trazer outros jogadores ficava difícil. Mais tarde, procuramos uma maneira de abrir o processo, para torná-lo mais fácil para os outros músicos tocarem no contexto de rede. O objetivo era criar uma nova maneira para pessoas fazerem música juntos. A solução bateu no ponto da facilidade de uso, e fornecimento de uma interface de usuário padrão, de modo que os jogadores poderiam conectar praticamente qualquer tipo de computador. The Hub é um pequeno computador dedicado a passar mensagens entre os jogadores. Ele serve como uma memória comum, mantendo informações sobre a atividade de cada jogador que seja acessível para os computadores de outros jogadores (BROWN; BISCHOF, 2002, seção 2.1).

The Hub

1987

- Nick Collins e Phil Niblock: curadoria para uma performance telemática entre a *Experimental Media* e *The Clocktower* em Nova York.
- Divididos em dois trios, se comunicam entre os dois espaços:
- John Bischoff, Tim Perkis, Mark Trayle;
- Chris Brown, Scot Gresham-Lancaster, e Phil Stone;
- Dois *Hub* intercomunicáveis.
- Cada *Hub* era um sistema centralizado para cada trio.
- “Simple Degradation”;
- “Borrowing and Stealing”;
- “Vague Notions”;
- performance de um *sexteto acusticamente divorciado mas informacionalmente ligado* (BROWN; BISCHOF, 2002, seção 2.2).

Ron Kuivila

• 1985: performance de *Water Surfaces*, na STEIM (*STudio for Electro-Instrumental Music*), em Amsterdã McLean e Wiggins (2009);
• 2007: reconstrução da peça "*TOPLAP001 - A prehistory of live coding*" (<http://toplap.org/wiki/TOPLAP_CDs>.);

Water Surfaces

Esta obra usou programação FORTH ao vivo; Curtis Roads (1986) testemunhou e relatou a performance de Ron Kuivila feita na STEIM em Amsterdã, em 1985; a performance original termina com a quebra do sistema...

Water Surfaces

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

Water Surfaces

Eu vi o software FORTH de Ron Kuivila quebrar e queimar no palco em Amsterdã em 1985, mas antes disso, não fez uma música muito interessante. A performance consistiu de digitação(WANG, 2005)

Live Algorithm Programming and Temporary Organization for its Promotion

- Manifesto elaborado por Ward et al.; McLean e Wiggins;
- DJ, Música algorítmica, Música de Processos;
- Regras práticas do *livecoding* publicadas por artistas-programadores ingleses;

LAPTOP

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

• Música como um Processo Gradual[6]
• A ligação conceitual do *live coding* com a Música de Processos e a Música Generativa é relativa ao uso de algoritmos mas não ao resultado sonoro como processo de escuta.
• Música Minimalista de Processos é apresentada por Mailman (2013, p. 128) → Música de Algoritmos Simples, um processo determinístico que age sobre focos de quadros temporais.
• Música Generativa é sensitiva às circuntâncias, isso quer dizer que irá reagir diferentemente dependendo das suas condições iniciais, onde ocorre e assim por diante.

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

TOPLAP I

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

TOPLAP II

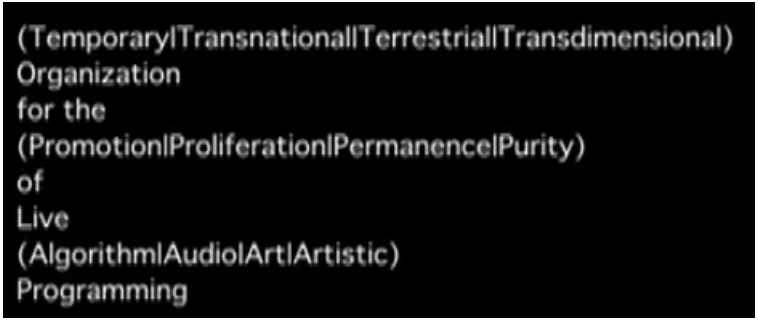


Figura : Definição do siginificado de TOPLAP. **Fonte**: Ramsay (2010).

TOPLAP III

A organização TOPLAP (www.toplap.org), cuja sigla possui diversas interpretações, uma sendo Organização Temporária para a Proliferação da Programação de Algoritmos Ao Vivo, foi criada para promover e explorar o live coding. TOPLAP nasceu em um bar enfumaçada em Hamburgo à uma da manhã em 15 de Fevereiro de 2004 (WARD et al., 2004, p. 246).

Manifesto Lubeck04

Escrito em um ônibus, é maisconhecido como “*Show us your screens*”, e prescreve algumas regras práticas do *live coding* (WARD et al., 2004, p. 247).

Manifesto Lubeck04 I

Exigimos:

- Acesso à mente do intérprete, para todo o instrumento humano.
- Obscurantismo é perigoso. Mostre-nos suas telas.
- Programas são instrumentos que podem modificar eles mesmos.
- O programa será transcendido - Língua Artificial é o caminho.
- O código deve ser visto assim como ouvido, códigos subjacentes visualizados bem como seu resultado visual.
- Codificação ao vivo não é sobre ferramentas. Algoritmos são pensamentos. Motosserras são ferramentas. É por isso que às vezes algoritmos são mais difíceis de perceber do que motosserras.

Reconhecemos contínuos de interação e profundidade, mas preferimos:

- Introspecção dos algoritmos.
- A externalização hábil de algoritmo como exibição expressiva/impressiva de destreza mental.
- Sem *backup* (minidisc, DVD, safety net computer).

Manifesto Lubeck04 II

Nós reconhecemos que:

• Não é necessário para uma audiência leiga compreender o código para apreciar, tal como não é necessário saber como tocar guitarra para apreciar uma performance de guitarra.

• Codificação ao vivo pode ser acompanhada por uma impressionante exibição de destreza manual e a glorificação da interface de digitação.

• Performance envolve contínuos de interação, cobrindo talvez o âmbito dos controles, no que diz respeito ao parâmetro espaço da obra de arte, ou conteúdo gestual, particularmente direcionado para o detalhe expressivo. Enquanto desvios na tradicional taxa de reflexos táteis da expressividade, na música instrumental, não são aproximadas no código, por que repetir o passado? Sem dúvida, a escrita de código e expressão do pensamento irá desenvolver suas próprias nuances e costumes.

Manifesto Lubeck04

- “Obscurantismo é perigoso. Mostre-nos suas telas”
- “Algoritmos são pensamentos, motosserras são ferramentas”

Mostre-nos suas telas

• "*Using Contemporary Technology in Live Performance; the Dilemma of the Performer*" (SCHLOSS, 2003)
• Crítica ao sétimo dos questionamentos sugeridos para uma performance de improvisação ao vivo com computadores Ward et al..

Mostre-nos suas telas I

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

Mostre-nos suas telas II

7. Pessoas que representam devem representar. Um concerto de música de computador não é uma desculpa/oportunidade para um programador(a) se sentar no palco. Sua presença melhora ou impede o desempenho da representação?

Algorithms are Thoughts, Chainsaws are Tools

Vídeo de Stephen Ramsay (2010), publicado no Vimeo, em 27 de fevereiro de 2010, como um *Coffee-Table Movie*. É uma análise pessoal da performance de *Strange Places* de Andrew Sorensen.

Algorithms are Thoughts, Chainsaws are Tools

Algoritmo como pensamento é um espaço conceitual abstrato

Algorithms are Thoughts, Chainsaws are Tools

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

Algorithms are Thoughts, Chainsaws are Tools

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

Referências I

BROWN, C.; BISCHOF, J. *INDIGENOUS TO THE NET: Early Network Music Bands in the San Francisco Bay Area*. 2002. Disponível em: <http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html>. Citado 6 vezes nas páginas 73, 75, 76, 79, 82 e 83.

CANUTE live in Jubez Karlsruhe Algorave. 2015–27–01. Disponível em: <https://www.youtube.com/watch?v=uAq4BAbvRS4>.
Citado na página 23.

CHESIRE, T. *Hacking meets clubbing with the 'algorave'*. Wired Magazine, 2013. Disponível em: <http://www.wired.co.uk/magazine/archive/2013/09/play/algorave>.
Citado na página 20.

Referências II

COLLINS, N.; McLean, A. Algorave: Live performance of algorithmic electronic dance music. In: *Proceedings of the International Conference on New Interfaces for Musical Expression*. [s.n.], 2014. p. 355–358. Disponível em: <http://nime2014.org/proceedings/papers/426_paper.pdf>.
Citado 2 vezes nas páginas 21 e 22.

COLLINS, N.; OLOFSSON, F. A protocol for audiovisual cutting. p. 4, 2003. Disponível em: <http://quod.lib.umich.edu/cache//b/b/p/bbp2372.2003.011/bbp2372.2003.011.pdf#page=2;zoom=75>.
Citado na página 22.

DOWNIE, M. *Choreographing the Extended Agent: performance graphics for dance theater*. phdthesis — MIT, 2005. Disponível em: <http://openendedgroup.com/writings/downieThesis.html>.
Citado 3 vezes nas páginas 27, 28 e 29.

Referências III

ENO, B. Base de dados, *Music for Airports liner notes*. 1978. Disponível em: <http://music.hyperreal.org/artists/brian_eno/MFA-txt.html>.
Citado na página 22.

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 91.

FABBRI, R.; VIEIRA, V. *Livecoding*. Wiki - Pontão Nós Digitais, 2011. Disponível em: <http://wiki.nosdigitais.teia.org.br/Livecoding>.
Citado na página 41.

GIOMI, F.; LIGABUE, M. *Conversazioni e riflessioni con Pietro Grossi*. [S.l.]: Sismel Edizioni del Galluzzo L'istante zero, 1999.
Citado 2 vezes nas páginas 62 e 66.

Referências IV

GRIFFTHS, D. *Tag Archives: algorave*. Dave's blog of art and programming, 2013. Disponível em: <http://www.pawfal.org/dave/blog/tag/algorave/>.
Citado 2 vezes nas páginas 24 e 25.

GRIFFTHS, D. *A cryptoweaving experiment*. 2015. Disponível em: <http://kairotic.org/a-cryptoweaving-experiment>>.
Citado na página 15.

GRIFFTHS, D. *Weavecoding performance experiments in Cornwall*. 2015. Disponível em: <http://www.pawfal.org/dave/blog/tag/weavecoding/>.
Citado 3 vezes nas páginas 12, 13 e 17.

Junior, A. D. d. C.; Lee, S. W.; Essl, G. Supercopair: Collaborative live coding on supercollider through the cloud. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 152–158. ISBN 978 0 85316 340 4.
Citado 2 vezes nas páginas 53 e 55.

Referências V

LUNHANI, G.; SCHIAVONI, F. Termpot: criação, edição de funções no navegador em tempo de execução. p. 154–159, 2015. Disponível em: <http://compmus.ime.usp.br/sbcm2015/files/proceedings-print.pdf>.
Citado na página 58.

MAILMAN, J. B. Agency, determinism, focal time frames, and processive minimalist music. In: *Music and Narrative Since 1900*. [s.n.], 2013. p. 125–144. Disponível em: <https://www.academia.edu/749803/Agency_Determinism_Focal_Time_Frames_and_Narrative_in_Processive_Minimalist_Music>.
Citado na página 90.

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

Referências VI

MCLEAN, A. *Artist-Programmers and Programming Languages for the Arts*. Tese (Doutorado) — Department of Computing, Goldsmiths, University of London, October 2011. Disponível em: <http://slab.org/writing/thesis.pdf>.
Citado na página 18.

MCLEAN, A. et al. (Ed.). *Proceedings of the First International Conference on Live Coding*. [S.l.]: ICSRiM, University of Leeds, 2015. 300 p. ISBN 9780853163404.

Citado na página 4.

McLean, A.; WIGGINS, G. *Patterns of movment in live languages*. 2009. Disponível em: <https://www.academia.edu/7249277/Patterns_of_movement_in_live_languages>.
Citado 2 vezes nas páginas 84 e 88.

MORI, G. Analysing live coding with etnographical approach. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 117–124. ISBN 978 0 85316 340 4.
Citado na página 10.

Referências VII

MORI, G. Pietro grossi's live coding. an early case of computer music performance. In: *ICLC2015 Proceedings*. [S.l.: s.n.], 2015. p. 125–132. ISBN 978 0 85316 340 4.
Citado 5 vezes nas páginas 62, 63, 66, 69 e 70.

RAMSAY, S. *Algorithms are Thoughts, Chainsaws are Tools*. Vimeo, 2010. Disponível em: <https://vimeo.com/9790850>.
Citado 3 vezes nas páginas 92, 93 e 102.

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>.
Nenhuma citação no texto.

RIETVELD, H. C. Bloomsbury Publishing Inc., 2013. 1–14 p. Disponível em: <http://file.ebook777.com/005/DjCulInTheMixPowTecAndSocChaInEleDanMus.pdf>.
Nenhuma citação no texto.

Referências VIII

ROADS, C. The second steim symposium on interactive composition in live electronic music. p. 45–50, 1986. Disponível em: <ttp://www.jstor.org/stable/3679484>.
Citado 2 vezes nas páginas 85 e 86.

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 na página 58.

ROBERTS, C.; WAKEFIELD, G.; WRIGHT, M. The web browser as synthesizer and interface. p. 6, 2013.
Citado na página 53.

SCHLOSS, A. Using contemporary technology in live performance; the dilemma of the performer. v. 32, p. 239–242, 2003. Disponível em: <https://people.finearts.uvic.ca/~aschloss/publications/JNMR02_Dilemma_of_the_Performer.pdf>.
Citado na página 99.

Referências IX

SICCHIO, K. Hacking choreography: Dance and live coding. p. 31–39, 2014. Disponível em: <http://muse.jhu.edu/journals/computer_music_journal/v038/38.1.sicchio.pdf>.
Citado 2 vezes nas páginas 28 e 32.

VIEIRA, V. et al. Vivace: A collaborative live coding language. arXiv, n. 1502, p. 16, 2015. Disponível em: <http://arxiv.org/abs/1502.01312>.
Citado na página 58.

WANG, G. *Read me paper - Revision as of 01:11, 1 August 2005 - A Historical Perspective*. 2005. Disponível em: <http://toplap.org/wiki/index.php?title=Read_me_paper&oldid=60#A_Historical_Perspective>.
Citado na página 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 7 vezes nas páginas 59, 88, 89, 92, 94, 95 e 99.