

CAPÍTULO 3 – A IMPLEMENTAÇÃO E PROGRAMAÇÃO DO SSInPIC EM PURE DATA

De acordo com Iazzetta (1996, tradução nossa) os agentes devem ter algum conhecimento prévio sobre as possibilidades de comportamento global do sistema interativo de forma a adquirirem uma base-comum, um espaço sógnico compartilhado na performance. Contudo, o humano e o computador operam com diferentes representações sógnicas do mesmo objeto, necessitando de um agente mediador na conexão entre estes signos. “A conexão semiótica entre símbolos computacionais e os objetos que estes símbolos representam é feita através de *software* e interfaces¹” que atuam como mediadores deste processo.

Neste capítulo são apresentados na seção 3.1 as conexões entre dispositivos (equipamentos) e elementos do sistema. Após a primeira seção são apresentadas as programações elaboradas no *software* “*Pure Data*”, a saber: seção 3.2 – programação da interface gráfica de controle a ser utilizada pelo performer computacional; seção 3.3 – programação do controle e monitoramento dos sinais de entrada e saída captados; seção 3.4 – programação do disparo das amostras sonoras, da captação dos parâmetros da performance envolvidos, e dos algoritmos de *cadeias de Markov* de primeira ordem que controlam os disparos; 3.5 - programação das linhas de *delays*, e dos dois processos de síntese sonora (que também envolvem aplicativos feitos por outros programadores).

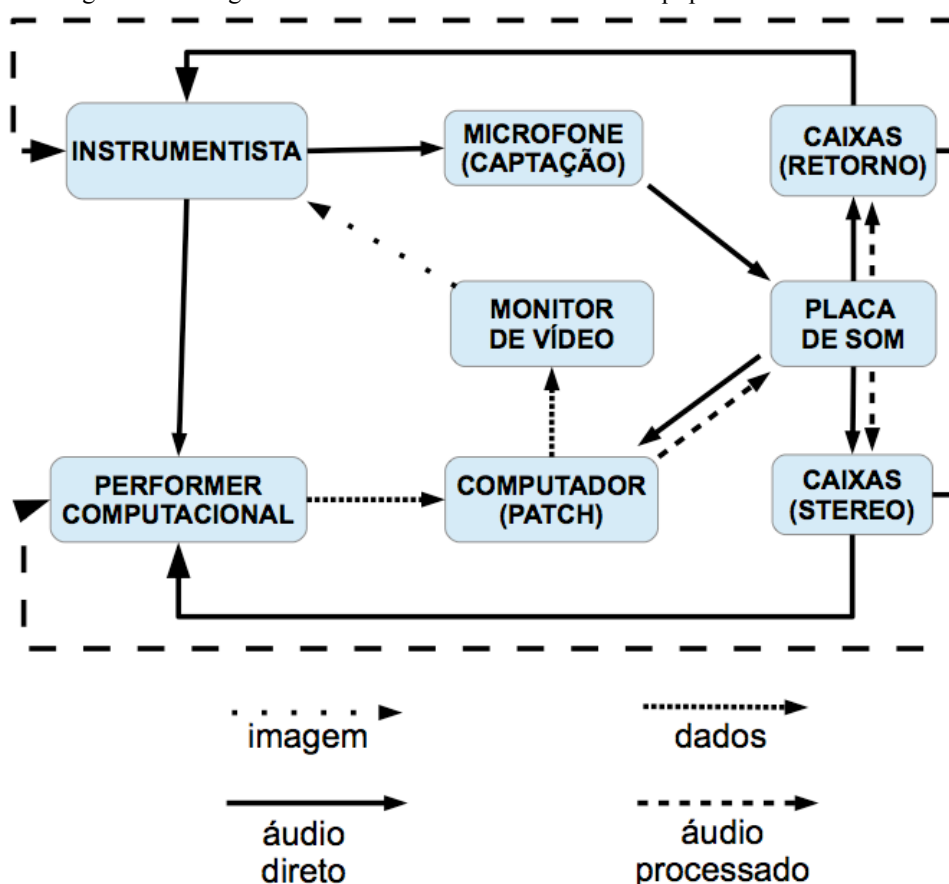
O capítulo encerra-se com um adendo a respeito da utilização do objeto “*random*” e “*shuffle*” na programação de eventos aleatórios, onde propõe-se uma programação que realmente consiga obter variabilidade à cada performance com o SSInPIC.

3.1 Conexões entre Elementos e Equipamentos no SSInPIC

Para uma melhor compreensão da programação do *patch* do SSInPIC convém entender as conexões entre elementos e equipamentos que ocorrem no sistema. Na Figura 15, apresenta-se um diagrama que ilustra estas conexões. O som dos instrumentistas, captados pelos microfones, tem sua difusão amplificada pelas caixas com ou sem processamento pelo *patch*.

¹ No original: “The semiotic connection between computational symbols and the objects these symbols stand for, is made through software and interfaces”.

Figura 15 – Diagrama das conexões entre elementos e equipamentos no SSInPIC



O sinal que é captado pelos microfones é enviado para o *patch* por meio de uma placa de som (“*M-Audio ProFire 2626*”) com oito entradas e oito saídas analógicas (Figura 16) que recebe de volta o som processado do *patch* e envia-o para as caixas. Dentre outros elementos, as análises dos sons instrumentais feitas pelo *patch* constituem um dos principais fatores no comportamento do sistema. No SSInPIC, o objeto “*fiddle~*” pode analisar ataques, frequências ou amplitudes dos sons instrumentais, produzindo dados que servem de parâmetros dos sons eletroacústicos que são gerados pela performance computacional. De acordo com Monteiro e Manzolli (2011, p. 501)

A análise musical com auxílio computacional pode prover informação adicional para estabelecer uma visão mais ampla de uma obra ao extrair informação direta do sinal de áudio. Por extensão, esta metodologia pode tornar-se adequada à análise de obras eletroacústicas e improvisação musical com eletrônicos ao vivo.

Entre os três instrumentistas existem duas caixas de retorno, e logo a frente deles duas caixas que fazem a projeção do mesmo sinal de áudio que é ouvido pelo performer de computador e o público. O performer computacional envia por um monitor de vídeo a

informação de qual instrumentista deve controlar os sons eletroacústicos. Na Figura 17 é mostrada uma imagem de uma das performances com o SSInPIC.

Figura 16 – Imagem da placa de som *M-Audio ProFire 2626*



Figura 17 –Imagem de uma performance com o SSInPIC



3.2 A Interface de Controle e sua programação

Durante a pesquisa e as oficinas realizadas com os instrumentistas foram testadas e programadas diferentes interfaces gráficas no software *Pure Data* para se obter um melhor desempenho do sistema eletrônico nas performances. Primeiramente testou-se uma programação do *patch* na qual seu funcionamento era quase que totalmente autônomo. Durante as oficinas realizadas com o SSInPIC, optou-se por dividir as seções da performance de acordo com o processamento aplicado, os quais são visualizados graficamente através de *canvas* (telas) com diferentes cores. Nestes primeiros experimentos com o funcionamento do aplicativo nas oficinas, a interface gráfica mostrava a duração de cada seção, a duração de

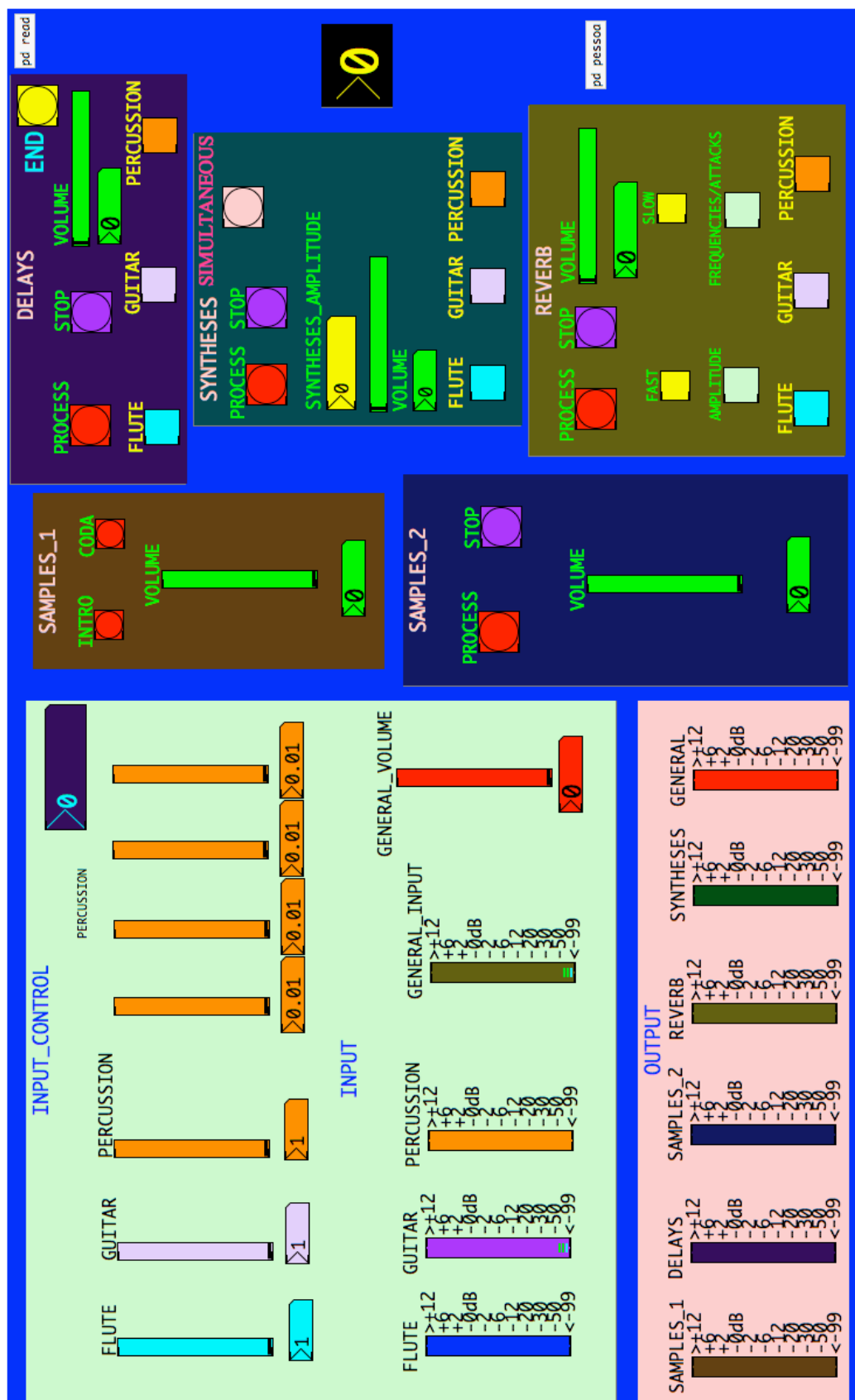
cada processamento, e às vezes o tipo de processamento que ocorria em algumas seções. Dependendo da seção, poderia haver mais de um processamento.

À medida que problemas foram surgindo durante as oficinas, tais como a falta de uma interação satisfatória dos sons eletroacústicos pelos instrumentistas - em seções em que havia esta necessidade -, alguns ajustes eram feitos tanto na programação dos processamentos, quanto na interface gráfica. Este problema ocorria pelo fato dos instrumentistas não terem o conhecimento do momento em que cada um deles estaria interagindo com o aplicativo, já que na maioria das vezes, apenas um deles estaria controlando o processamento dos sons eletroacústicos por meio do sinal de áudio vindo da sua performance². Na idealização do sistema, primeiramente foi pensado que os instrumentistas perceberiam pela escuta qual deles poderia estar interagindo, mesmo que o tempo de interação de cada um fosse curto (cerca de 10 segundos). Além de aumentar o tempo designado a cada instrumentista, optou-se por testar outras soluções, como por exemplo, a desautomatização de algumas funções do *patch* e o uso de um monitor de vídeo em que os instrumentistas recebiam a informação de qual deles estaria estimulando e interagindo com a parte reativa do aplicativo. Na Figura 18, apresenta-se a interface gráfica com estas soluções.

Na Figura 18 podem ser observados vários *GUIs* (*Graphical User Interfaces*) que são elementos gráficos da interface, com os quais o usuário do aplicativo pode interagir (PORRES, 2010). Optou-se na programação do aplicativo por delegar um maior controle ao performer computacional, o que resolveria alguns problemas referentes às automatizações de processos no *patch* identificados nas primeiras oficinas com os instrumentistas. Assim, a interface acima evoluiu de um processo em que apenas eram vistas algumas informações do processamento das interações para uma interface em que o performer computacional interage com o “acionamento de botões” para inicializar e finalizar processos, por exemplo.

Desautomatizando várias funções do *patch* (exceto os disparos de amostras no *canvas SAMPLES_2*, a ser abordado mais adiante) e realizando performances com ele, foi possível idealizar uma melhor proposta de programação. Além disso, desautomatizando algumas funções do *patch* foi possível obter maior variabilidade sonora devido à possibilidade de decisões circunstanciais em virtude do contexto que se configurava a cada momento, como foi comentado no Capítulo 2.

² Pelo fato dos instrumentistas não terem o conhecimento de qual deles estaria interagindo com o aplicativo, todos passavam a tocar simultaneamente, o que dificultava mais ainda a percepção de qual deles controlava os sons eletroacústicos.



Os *sliders* no *canvas* “*INPUT_CONTROL*” servem para o controle da entrada (*input*) do nível do sinal de áudio - captado por microfones - de cada um dos instrumentos musicais para ser analisado e processado pelo aplicativo. Existem quatro *slides* de cor laranja do lado direito deste *canvas* que são os quatro microfones utilizados para o *set* de percussão, os quais tem seus sinais somados no *slider* laranja mais à esquerda, que envia a resultante desta soma aos módulos de processamento. Todos os outros *sliders* do aplicativo são controladores de volume. Ainda no primeiro *canvas*, em sua parte inferior, temos *VUs* que permitem ao usuário visualizar o nível do sinal de cada instrumento que está sendo captado. Desta forma, caso o nível do sinal captado de algum instrumento não esteja equilibrado, ele pode ser regulado pelos *sliders* citados acima. Esta regulação é feita de acordo com as respostas (*output*) do processamento. Neste mesmo *canvas*, há uma caixa de número no canto superior direito que mostra o tempo de duração da performance em minutos e segundos. Logo abaixo há o *canvas* “*OUTPUT*”, onde pode ser visualizado o nível de amplitude de cada processamento. À direita destes dois *canvas* encontram-se mais cinco *canvas* em que se localizam o controle do disparo de amostras (*SAMPLES_1* e *SAMPLES_2*) e de determinados processamentos ou sínteses (*DELAYS*, *SYNTHESES* E *REVERB*). Os *bangs* nestes cinco *canvas* iniciam ou param os disparos das amostras e os processamentos/sínteses. As caixas de número em verde-claro mostram os níveis dos diferentes volumes definidos pelo performer computacional. Existem três *toogles* em cada um dos *canvas* dos processamentos (*flute*: azul-claro; *guitar*: lilás; *percussion*: laranja) que selecionam o instrumento que terá a análise do seu sinal de áudio enviada ao processamento.

No *canvas* *SYNTHESES* há uma caixa de número amarela que mostra em valores de 0 a 1 a amplitude do processamento que é controlada inversamente pelo sinal do instrumento que está interagindo no momento (“*ducking*”). Isto quer dizer que quanto maior a amplitude do instrumento que controla o processamento *SYNTHESES*, menor será a amplitude do processamento e vice-versa.

Existem mais quatro *toogles* (amarelos e verdes) no *canvas* do processamento *REVERB* que subdivide este processamento em mais partes. Nos *toogles* amarelos (*fast* ou *slow*), por exemplo, o performer computacional define a velocidade com que o aplicativo responde ao estímulo da performance instrumental. E nos *toogles* verdes é escolhido o parâmetro do sinal da performance instrumental (*amplitude* ou *frequencies attacks*) que estará sendo analisado como *input* para o processamento.

Existem ainda mais dois *bangs* nesta interface: *SIMULTANEOUS*, localizado no *canvas* *SYNTHESES*, e *END* localizado no *canvas* *DELAYS*. Em *SIMULTANEOUS*, ocorre

um decaimento do nível da amplitude dos módulos *SAMPLES_2*, *SYNTHESES* e *REVERB* para que a sobreposição (soma) dos seus sinais de saída não cause uma saturação de amplitude. Em *END*, envia-se um aviso aos instrumentistas (explicado logo abaixo) de que após 10 segundos, encerrar-se-á a interação do *patch*.

Na extrema direita da interface, de baixo para cima, encontra-se o *subpatch* “*pd pessoa*” que recebe a informação de qual instrumentista está enviando sinal de áudio para o *patch*. A informação recebida é: 1: flautista; 2: violonista; 3: percussionista ou; 0: dez segundos para desligar o aplicativo. Estes números são enviados para um monitor de vídeo onde os instrumentistas podem visualizá-los e saberem quando estarão interagindo com o aplicativo. Na interface gráfica, logo acima deste *subpatch* “*pd pessoa*” há uma representação do que aparece no monitor de vídeo.

3.3 Input e Output

Nesta seção será descrita a programação que trata da entrada do sinal dos instrumentos acústicos e saída do sinal processado pelo aplicativo, como também as características da interface que controlam os dois sinais.

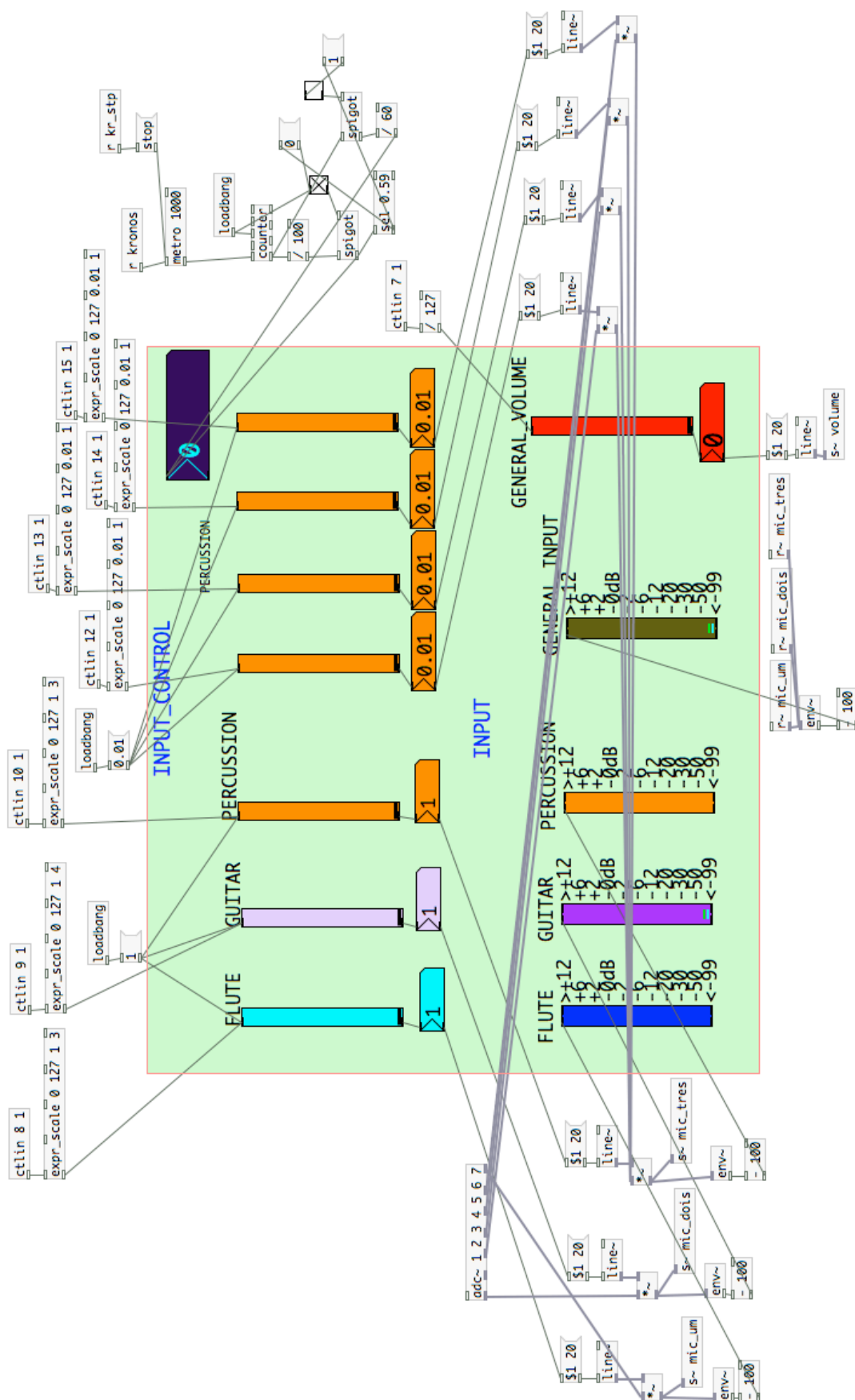
3.3.1 Controle do Sinal de Entrada (INPUT)

Para o controle do nível do sinal de entrada (input) dos instrumentos (*canvas* “INPUT_CONTROL”) foi usado um controlador MIDI dotado de botões, *knobs* e *faders* (B CONTROL FADER BCF 2000 BEHRINGER) - visto na Figura 19 - para facilitar a ação do performer de computador. Para isto, foi utilizado o objeto “*ctlin*” do *Pd* que recebe os valores MIDI resultantes da manipulação dos *faders* do controlador. Os valores recebidos por “*ctlin*” são enviados aos *sliders* do *subpatch* INPUT visto na Figura 20 (para serem visualizados pelo performer computacional) e multiplicam o sinal de áudio captado dos instrumentos musicais, enviando o resultado para os módulos de processamento ou síntese. Para realizar o mapeamento destes valores foi utilizado o objeto “*expr_scale*”.

Figura 19 – Imagem do Controlador MIDI com *faders* utilizados para controlar os *sliders* de volume



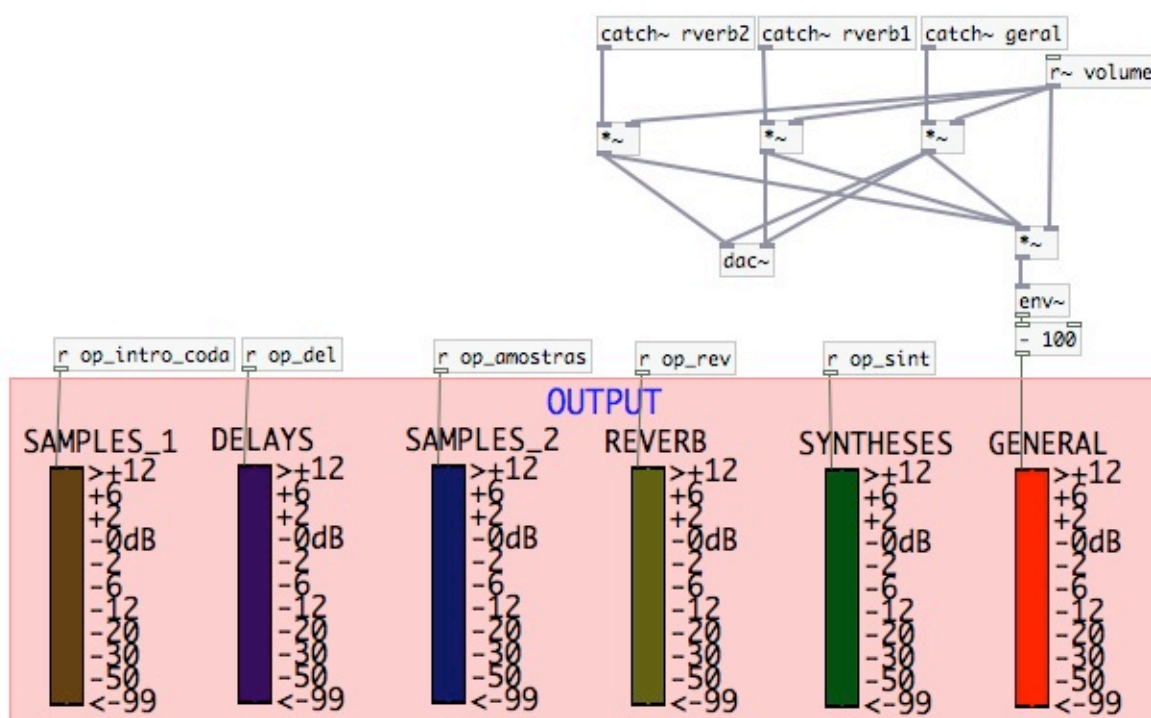
Figura 20 – Subpatch “input”



3.3.2 Monitoramento do Sinal de Saída (OUTPUT)

No *canvas* “*OUTPUT*” (Figura 21), no *subpatch* *pd* “*output_visor*” são recebidos todos os sinais dos processamentos e/ou amostras disparadas. Neste *subpatch*, portanto, é que se encontra o objeto “*dac~*” que converte o sinal digital dos processamentos/sínteses e dos disparos em sinal de áudio analógico. Os sinais de cada processamento, sínteses ou disparo de amostras são enviados para os “*VUs*” através de objetos “*receives*” com seus argumentos respectivos. No objeto “*catch~* geral” os sinais são somados, excetuando o sinal do *subpatch* do *canvas* “*REVERB*” que separa o mesmo sinal em dois canais - vistos nos objetos “*catch~ rverb1*” e “*catch~ rverb2*” - com um pequeno atraso de um em relação ao outro, conforme será comentado na subseção 3.5.3.1.

Figura 21 – Subpatch “*output_visor*”



3.4 Programação das interações dinâmicas (disparo de amostras)

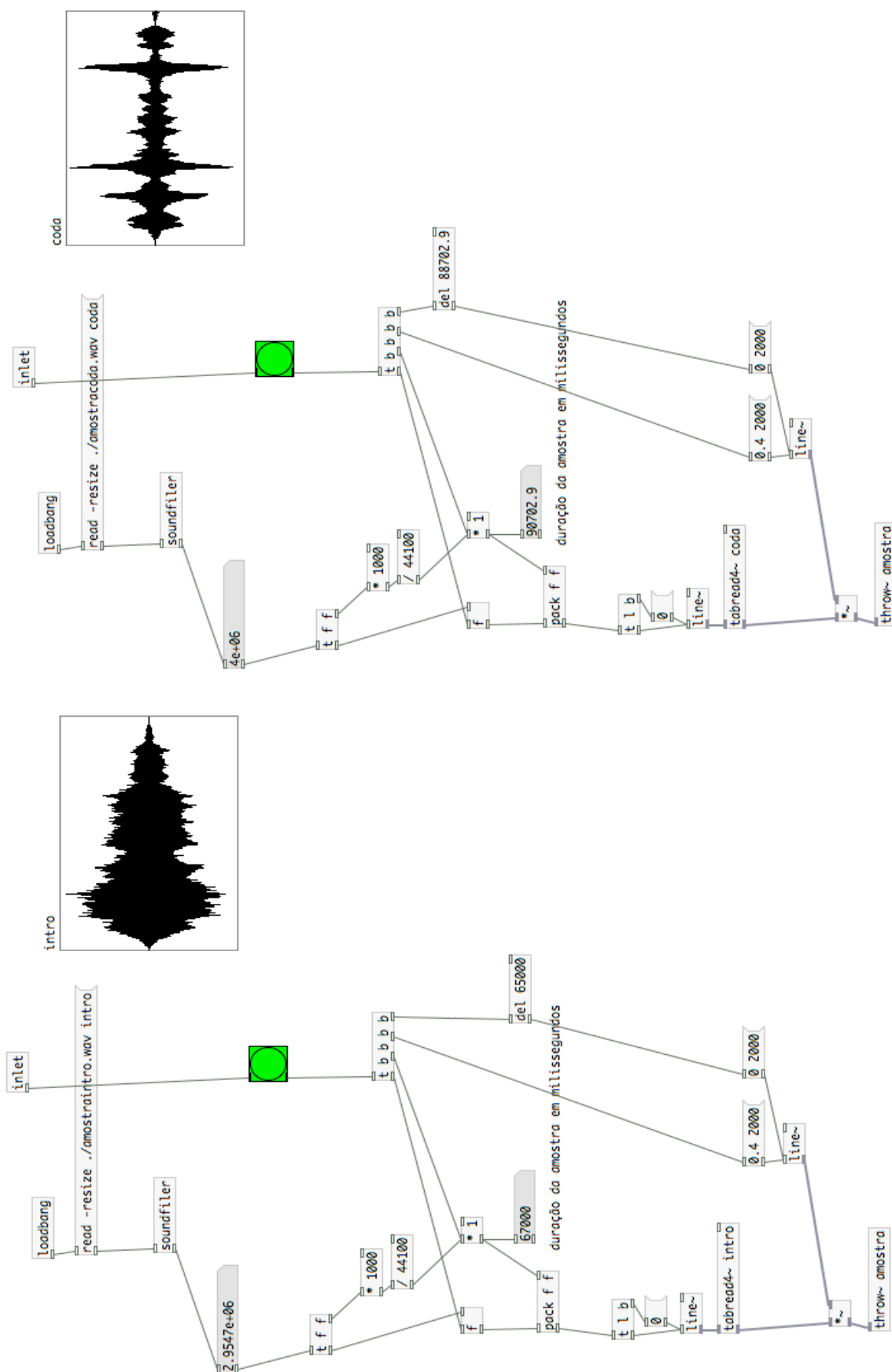
Os disparos de amostras sonoras programados no *patch* geral podem ser divididos em dois grupos: 1) disparos de amostras sonoras pré-gravadas controlados pelo performer de computador; e 2) disparos de amostras sonoras controlados por *cadeias de Markov*. O segundo grupo ainda pode ser subdividido em 2 semigrupos: 1) disparos das amostras sonoras pré-gravadas controlados por *cadeias de Markov*; e 2) disparos das amostras sonoras gravadas

durante a performance controlados por *cadeias de Markov*. A seguir são descritas as programações referentes a todos esses tipos de disparos.

3.4.1 *Canvas “SAMPLES_1”*: Disparos das Amostras Sonoras Pré-Gravadas Controlados pelo Performer Computacional

As duas amostras pré-gravadas que iniciam e finalizam uma performance com o SSInPIC – “*amostraintro.wav*” e “*amostracoda.wav*”, respectivamente - comentadas no Capítulo 2, encontram-se dentro do *subpatch* “*pd amostras_intro_coda*”. Abaixo, na Figura 22 podem ser vistas as formas de onda das duas amostras dentro do *subpatch* e a programação de seus disparos.

Figura 22 – Subpatch “pd amostras_intro_coda”



3.4.2 Canvas “SAMPLES_2”: Disparos de Amostras Sonoras Controlados por Cadeias de Markov

Além das amostras “*amostraintro.wav*” e “*amostracoda.wav*”, outras amostras de áudio são disparadas durante a performance em momentos indeterminados. Nove destas amostras são pré-gravadas e foram manipuladas em estúdio, conforme descrito em 2.2.2.2; outras são gravadas durante a performance. É digno de nota que todas as amostras pré-gravadas não sofreram cortes ou colagens durante o trabalho em estúdio, mas apenas processamentos e mixagem.

Tanto as amostras pré-gravadas, quanto as que foram gravadas ao vivo tem seus disparos controlados pelo *subpatch* “*disparo*”, encontrado no *canvas* “SAMPLES_2”. Tais amostras são disparadas de acordo com o controle de algoritmos probabilísticos de *cadeias de Markov* de primeira ordem, descritos no Capítulo 2. Sempre que ocorrer um disparo de qualquer amostra, a amplitude dos outros processamentos/sínteses (que por ventura pode estar ocorrendo) é diminuída. Um objeto “*random*” com argumento 2, escolhe aleatoriamente entre o disparo de uma amostra pré-gravada (*subpatch* “*pd samples*”) ou o de uma amostra gravada da performance, conforme pode ser visto no *subpatch* “*pd amostras*” da Figura 23, indicado por uma seta vermelha. A cada vez que o objeto “*random 2*” dispara uma amostra sonora, o objeto “*s fade_o*” (seta verde na Figura 23) envia uma mensagem “*bang*” aos objetos “*r fade_o*” nos *subpatches* das Figuras 30, 31 e 44, os quais fazem diminuir as amplitudes das respostas sonoras dos respectivos *subpatches* dos *canvas* “DELAYS”, “SYNTHESES” e “REVERB”. Da mesma maneira, ao final da reprodução de uma amostra sonora contida dentro do *subpatch* da Figura 23, um objeto “*s fade_i*” - localizado nos *subpatches* onde cada amostra sonora é armazenada -, envia uma mensagem “*bang*” aos objetos “*r fade_i*” para subir de volta o nível da amplitude em cada um dos processos comentados – “DELAYS”, “SYNTHESES” e “REVERB”, respectivamente nas Figuras 30, 31 e 44.

3.4.2.1 Disparos das Amostras Sonoras Pré-Gravadas Controlados por Cadeias de Markov.

As amostras sonoras pré-gravadas são disparadas de acordo com o valor de amplitude que é analisado de um dos três instrumentistas durante todo o tempo que o *subpatch* do *canvas* “SAMPLES_2” está funcionando. Durante a performance com o SSInPIC não se sabe qual dos instrumentistas tem a amplitude analisada. Contudo, o disparo da amostra só ocorre quando o objeto “*random 2*” do *subpatch* “*pd amostra*” (Figura 23) envia um *output* com valor numérico igual a 1 (um). Nesse instante é captado um valor da amplitude do instrumento que, por ventura, está sendo analisado no momento, e dispara-se a amostra (do banco que está aberto) com este valor. Conforme a distribuição das nove amostras sonoras (pré-gravadas) nos três bancos (de maneira que cada banco possua três amostras cada uma com um pico de amplitude diferente - Figura 12), e o mapeamento de seus disparos (Figura 13), uma captação da performance instrumental com baixo nível de amplitude, por exemplo, vai disparar a amostra (do banco que foi aberto pelo algoritmo) com pico de amplitude baixo, e assim por diante, conforme o *patch* da Figura 24. De acordo com a Figura 24, seja “*x*” a amplitude captada do instrumento, as amostras serão disparadas segundo o seguinte critério na função $f(x)$:

amostra com pico de amplitude baixo, se $1 \leq x \leq 50$;

amostra com pico de amplitude médio, se $50 < x \leq 60$;

amostra com pico de amplitude alto, se $60 < x$.

O disparo de determinada amostra contida em um dos bancos é que vai determinar a distribuição das probabilidades de abertura de um dos três bancos para o disparo da próxima amostra pré-gravada. Desta forma, a cada amostra disparada é acionada uma programação semelhante a da Figura 25, em que objetos “*moses*” distribuem as probabilidades para a abertura do banco que contém uma das três prováveis amostras a ser disparada numa próxima ocasião. Um dos *outlets* de um dos objetos “*moses*” envia uma mensagem “*bang*” que seleciona um dos números – 1, 2 ou 3 – que irá abrir um dos três bancos de amostras pré-gravadas.

Figura 24 – Imagem da programação do mapeamento: análise do nível da amplitude da performance instrumental x disparo da amostra pré-gravada com o nível do pico de amplitude correspondente

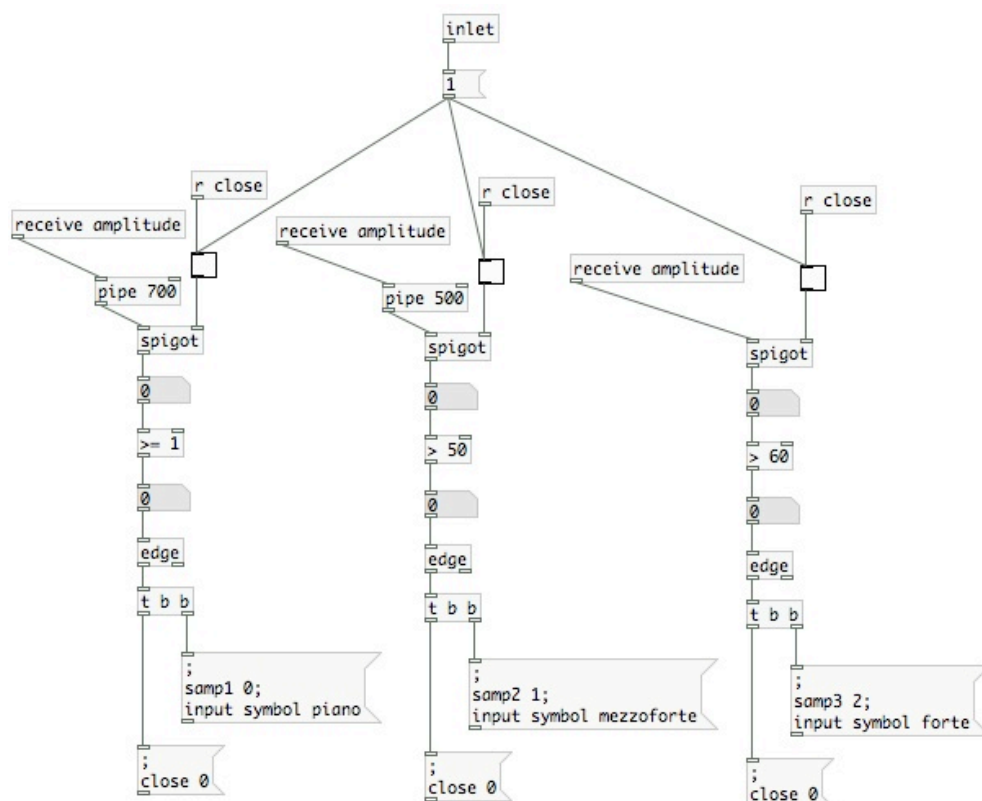
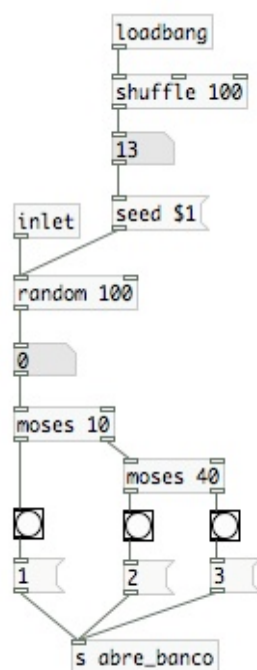


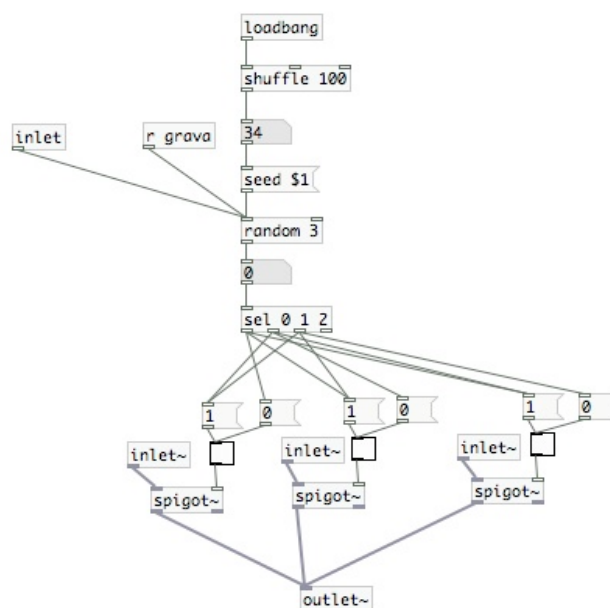
Figura 25 – Imagem com um exemplo da programação da distribuição de probabilidades da abertura do banco (que contém a próxima amostra sonora pré-gravada a ser disparada) que é mapeada ao disparo de determinada amostra



3.4.2.2 Disparos das Amostras Sonoras Gravadas Durante a Performance Controlados por Cadeias de Markov.

Ao terminar a reprodução da amostra “*amostraintro.wav*”, do subpatch “*pd amostras_intro_coda*” (comentadas na seção 3.4.1), no início da performance, é iniciada a gravação de cinco amostras sonoras seguidas com diferentes durações temporais: 3” (segundos); 5”; 8”; 13”; e 21”. E após 47”, que é a soma destas durações, é iniciado o disparo randômico (objeto “*random2*” na Figura 23) das amostras gravadas e pré-gravadas, com intervalos entre os disparos com durações variáveis entre 13” e 21”. As amostras gravadas da performance são gravações do sinal vindo de apenas um dos instrumentistas (flautista, violonista ou percussionista). Não há o conhecimento pelos agentes humanos do momento das gravações das amostras na performance e nem de qual dos instrumentistas é gravado. Na Figura 26, podem ser observadas estas características aleatórias na programação.

Figura 26 – Imagem da programação da escolha aleatória de qual instrumentista é feita a gravação de amostras sonoras durante a performance



Nota-se na programação vista na Figura 26 que a geração do número aleatório pode variar entre 0, 1 e 2, tal que: 0 = flautista; 1 = violonista; e 2 = percussionista. Ao final da reprodução de uma amostra com determinado tempo de duração, é gravada outra amostra de mesma duração no *array* que continha a última amostra disparada. Assim, como comentado no Capítulo 2, o disparo de determinada amostra gravada da performance acontece somente uma vez, ou seja, não haverá sua repetição na performance, diferentemente das amostras pré-

gravadas que podem ser repetidas.

Quando é iniciada uma nova gravação, envia-se uma mensagem “*bang*” pelo objeto “*s grava*” (Figura 27) que é recebida pelo objeto “*r grava*” (Figura 26) que repassa a mensagem para o objeto “*random 3*” (o qual escolhe o instrumentista para gravar a amostra, conforme já explicado). Algumas vezes, durante a performance, pode ocorrer o disparo de uma amostra “vazia”, pois pode acontecer do instrumentista não tocar durante a gravação de uma amostra da sua performance.

Na Figura 27, observa-se um exemplo do funcionamento de um dos cinco *subpatches* – *pd primeira*, *pd segunda*, *pd terceira*, *pd quarta*, *pd quinta* –, que aparecem encapsulados na Figura 23. Na Figura 27, podem ser observadas cinco caixas de mensagens endereçadas ao objeto “*tabread4~*”, as quais são as cinco possibilidades de reprodução da amostra. Um algoritmo de *Cadeias de Markov* de primeira ordem (Figura 28) programado e encapsulado em cinco *subpatches*, como por exemplo, o *subpatch* “*pd markov_d1*” que aparece na Figura 27, é que vai selecionar probabilisticamente uma das cinco possibilidades de reprodução de cada amostra, as quais definem sua velocidade ou se são tocadas de forma inversa ou não. A mesma programação do algoritmo da Figura 28 é encapsulada também no *subpatch* “*pd markov_banco*” que aparece na Figura 23. Este *subpatch* controla a probabilidade da ocorrência de disparo da amostra gravada dentro de cada um dos cinco *subpatches* (*pd primeira*, *pd segunda*, *pd terça*, *pd quarta* e *pd quinta*) que aparecem na Figura 23. Na Figura 29 é apresentado um diagrama de controle estocástico dos disparos das amostras sonoras do SSInPIC, que resume esta seção.

Figura 27 – Imagem de um exemplo de um dos cinco *subpatches* de gravação e disparo das amostras gravadas durante a performance

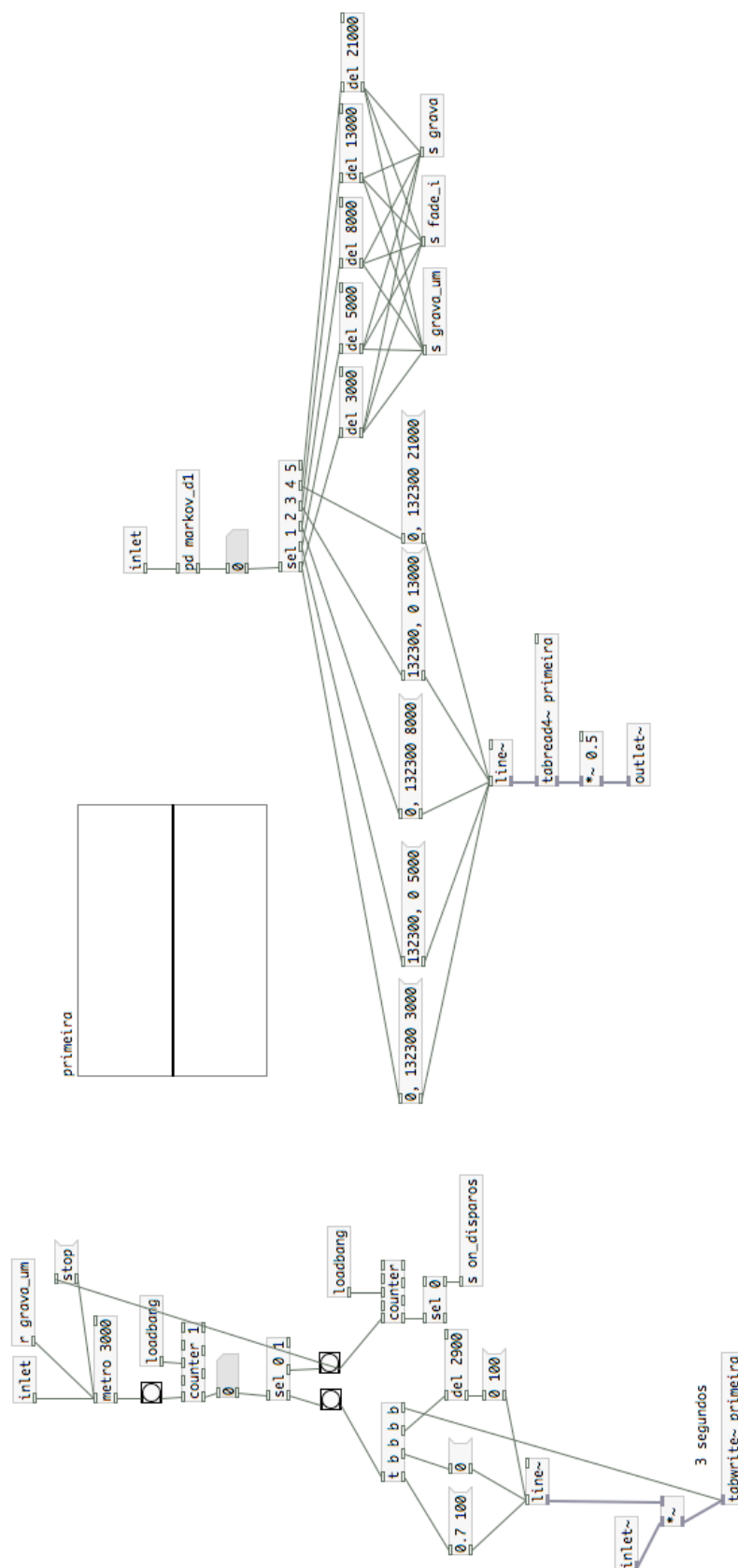
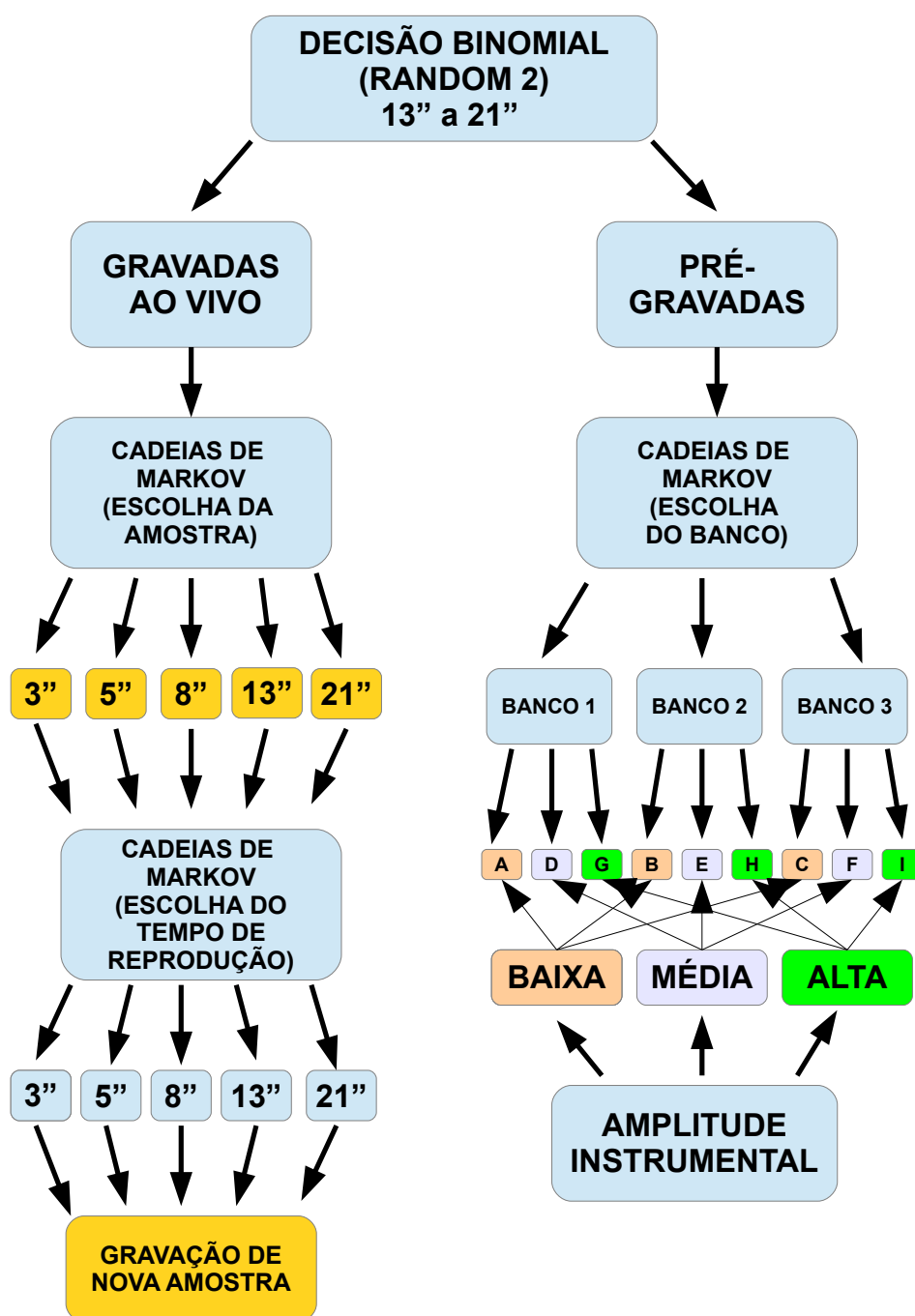


Figura 29 – Diagrama de Controle Estocástico dos disparos de amostras sonoras do SSInPIC



3.5 Programação das interações estáticas (*delays* e geração sonora)

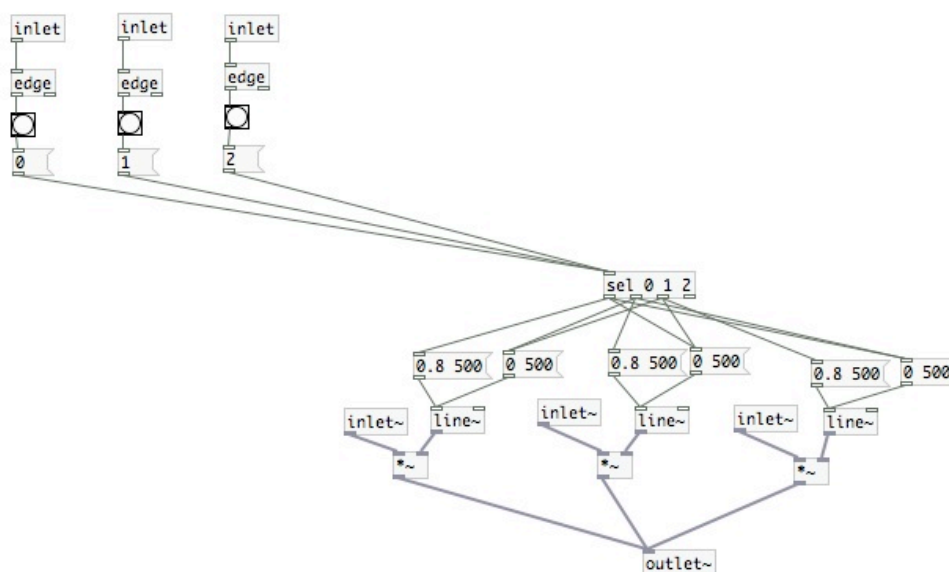
Nesta seção será descrita a programação que envolve as interações estáticas que ocorrem entre *performers* e sons eletroacústicos, ou seja, as interações em que as respostas do aplicativo se comportam de maneira mais reativa e previsível. As descrições englobam os

canvas “DELAYS”, “SYNTHESES” e “REVERB” da interface gráfica.

3.5.1 Canvas delays

A cada vez que um instrumento é selecionado pelo acionamento do seu *toggle* no canvas DELAYS, quatro linhas de *delays* são aplicadas aos sons captados desse instrumentista na performance. Quando um *toggle* é acionado, uma mensagem numérica (0: flauta; 1: violão; ou 2: percussão) é enviada para o objeto “*select*” que inicia um *fade in* no instrumento selecionado e um *fade out* até o valor 0 (zero) de amplitude para os demais instrumentos, o que pode ser visualizado na programação do *subpatch* “*pd micro_d*” da Figura 30. Assim, o processamento com *delays* é aplicado aos sons produzidos por apenas um dos instrumentistas a cada vez, durante a performance.

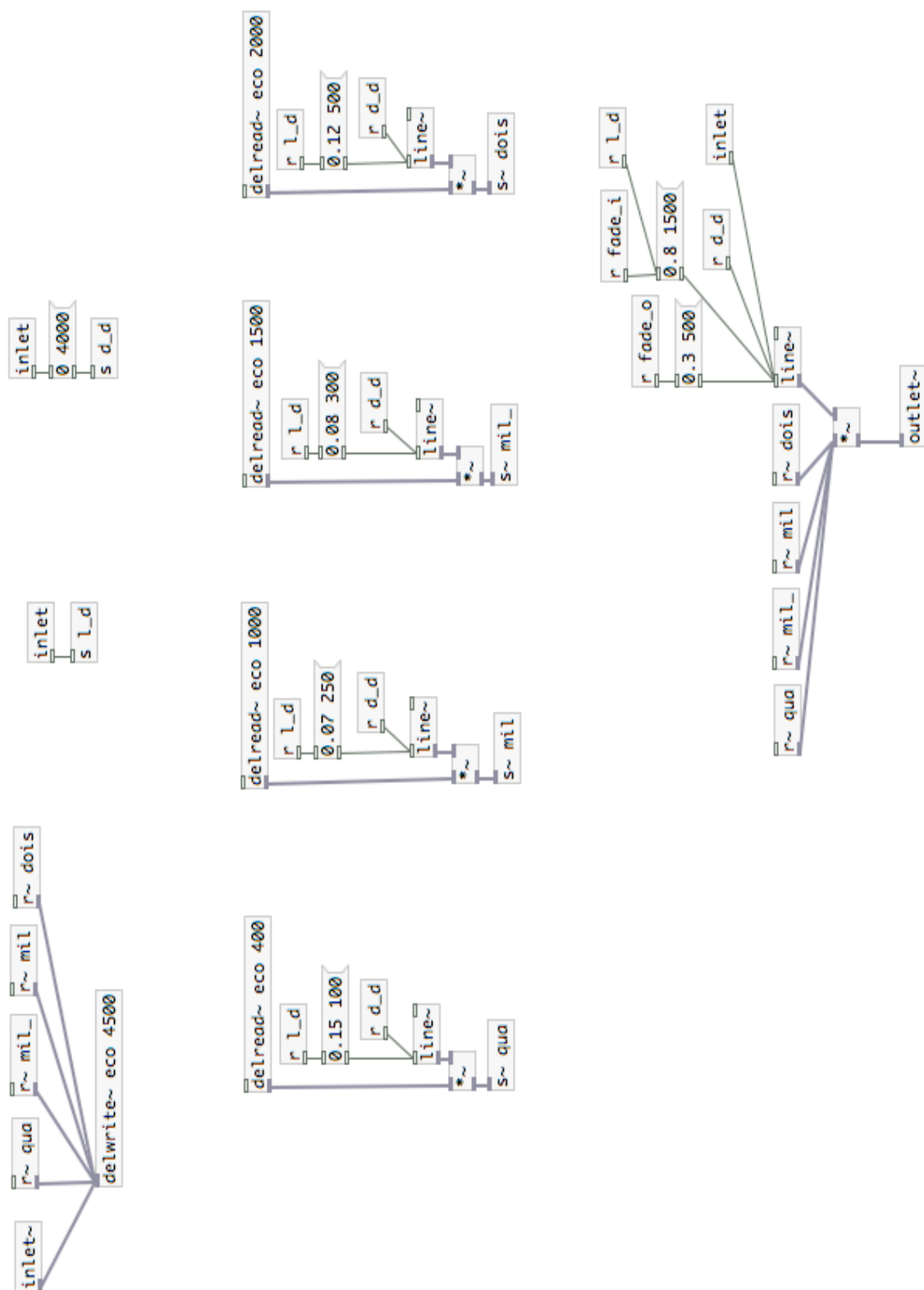
Figura 30 – Subpatch “*pd micro_d*”: *fade in* e *fade out*



As quatro diferentes linhas de *delays* aplicadas aos sons captados dos instrumentos musicais tem diferentes níveis de amplitude. Os tempos de atraso do sinal (*delays*) e das porcentagens da amplitude do sinal original aplicados em cada linha de *delay* foram: 1) 400 milissegundos e 15% da amplitude; 2) 1000 milissegundos e 7% da amplitude; 3) 1500 milissegundos e 8% da amplitude; e 4) 2000 milissegundos e 12% da amplitude. Esses valores dos atrasos do sinal (*delay*) e da porcentagem de amplitude do sinal original foram definidos de forma diferente de valores comumente usados para se obter um efeito de “eco”, o qual utilizaria valores baseados em uma progressão geométrica. No *subpatch* da Figura 31

pode ser vista a programação com esses valores, os quais foram baseados nas experimentações realizadas e nos efeitos pretendidos. Há um efeito de *feedback* resultante da soma do sinal de saída das linhas de *delays* (objeto “*delread~*”) - que é enviado ao objeto “*delwrite~*” - com o sinal do instrumento ao vivo. Esta programação do *subpatch* da Figura 31 procurou obter um interessante efeito de reverb, prolongando eventos efêmeros dos sons acústicos tocados ao vivo pelos instrumentistas.

Figura 31 – Subpatch “pd delays”: processamento dos *delays*



3.5.2 Canvas *SYNTHESES*

No Canvas “*SYNTHESES*” existem quatro módulos de geração sonora: síntese aditiva no *subpatch* “*pd inarmon*”; síntese AM (modulação de amplitude) no *subpatch* “*pd am*”; a síntese aditiva no *subpatch* “*pd textura*”; e síntese granular no *subpatch* “*pd sintese_granular*”. Depois que são acionadas, essas sínteses ocorrem de forma aleatória, simultaneamente ou não, e tem sua amplitude controlada inversamente pela amplitude dos instrumentos na performance com o SSInPIC. A seguir será descrita a programação em *Pure Data* na implementação destes processos de geração sonora com o parâmetro de amplitude controlado pelos *performers*.

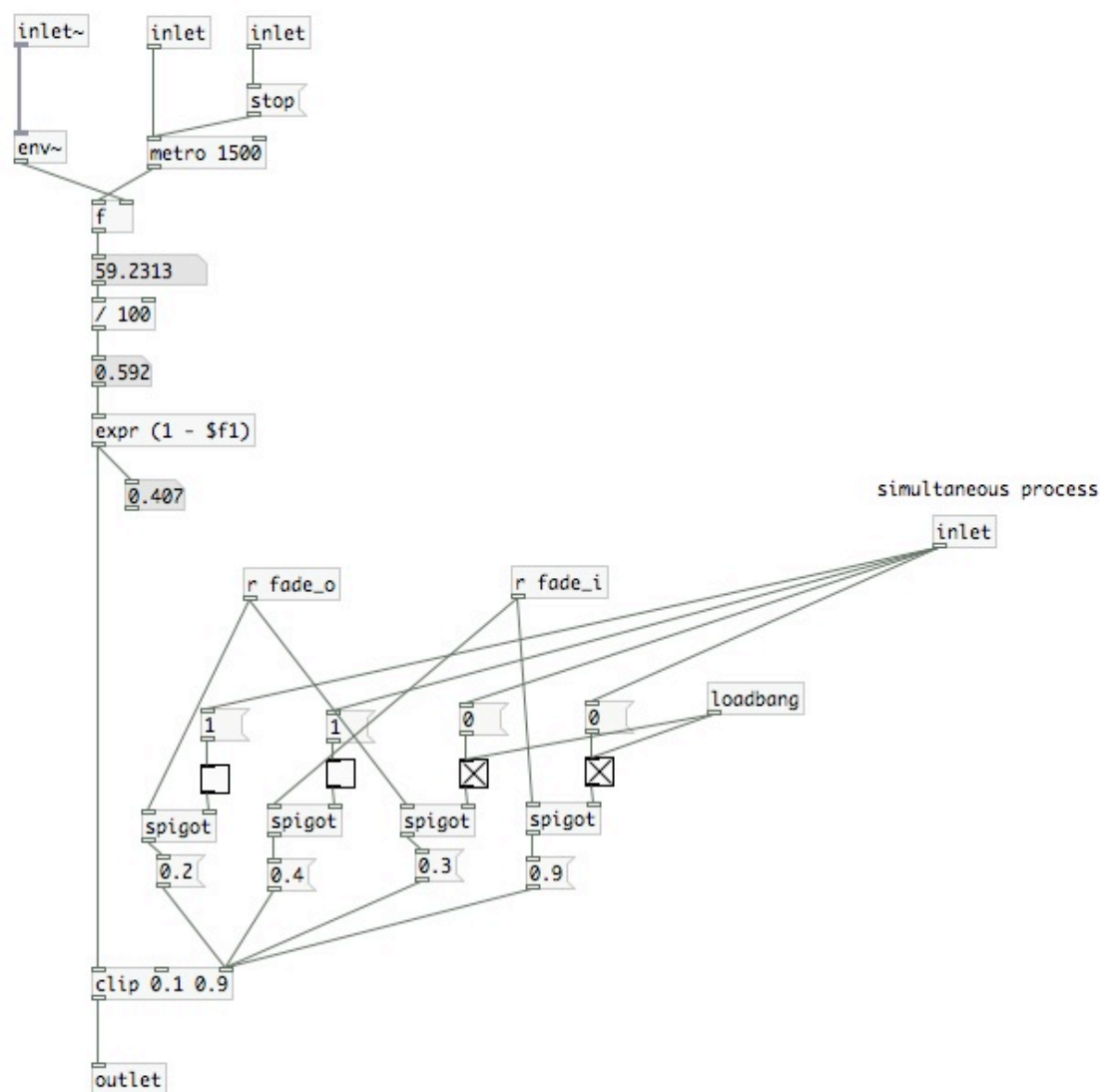
3.5.2.1 Controle da Amplitude

No “*subpatch pd synth*”, localizado no *canvas SYNTHESES*, existem quatro módulos de geração sonora que podem ou não ocorrer simultaneamente. Os instrumentistas interagem controlando a somatória da amplitude geral dos quatro módulos de forma inversamente proporcional, isto é, quanto maior a amplitude da performance do instrumentista, menor é a da somatória dos quatro módulos e vice-versa. Este efeito é conhecido por “*ducking*”. O instrumento que é escolhido para controlar a somatória das amplitudes desses módulos é feito pelo performer computacional através de um *subpatch* chamado “*pd micro_sint*” idêntico ao *subpatch* da Figura 30. O *subpatch* que faz o efeito “*ducking*” é mostrado na Figura 32.

Na Figura 32, pode ser visto o uso do objeto “*env~*”, que transforma o sinal de áudio recebido pelo “*inlet~*” em dados numéricos. O uso do objeto “*float*” foi necessário para amostrar os dados vindos do objeto “*env~*” com a periodicidade desejada (a cada 1500 ms, por meio do objeto “*metro*”). Com base no argumento do “*metro*”, pode-se ver que a análise da amplitude ocorrerá a cada 1500 milissegundos. Os valores de saída são enviados, respectivamente, para dois objetos de operação matemática “*/100*” e “*expr (1 - \$f1)*” cujos resultados serão limitados pelo objeto “*clip*” para um melhor controle da amplitude. Os argumentos do objeto “*clip*” variam durante a performance (ver Figura 32), de acordo com a quantidade de outros processamentos que podem ocorrer simultaneamente aos processos de geração sonora explicados adiante. Os valores de amplitude obtidos são enviados, depois do objeto “*outlet*”, à caixa de mensagem “*\$1 1500*” (Figura 33) que controla o objeto “*line~*”, interpolando os valores recebidos a cada 1500 ms. É importante observar que este valor de 1500 milissegundos é o mesmo tempo do objeto “*metro*” da Figura 32, gerando uma

sincronia entre o envio dos dados do objeto “env~” e o funcionamento do objeto “line~”.

Figura 32 – Subpatch “pd síntes” (programação do controle de amplitude dos módulos de geração sonora por efeito “ducking”)

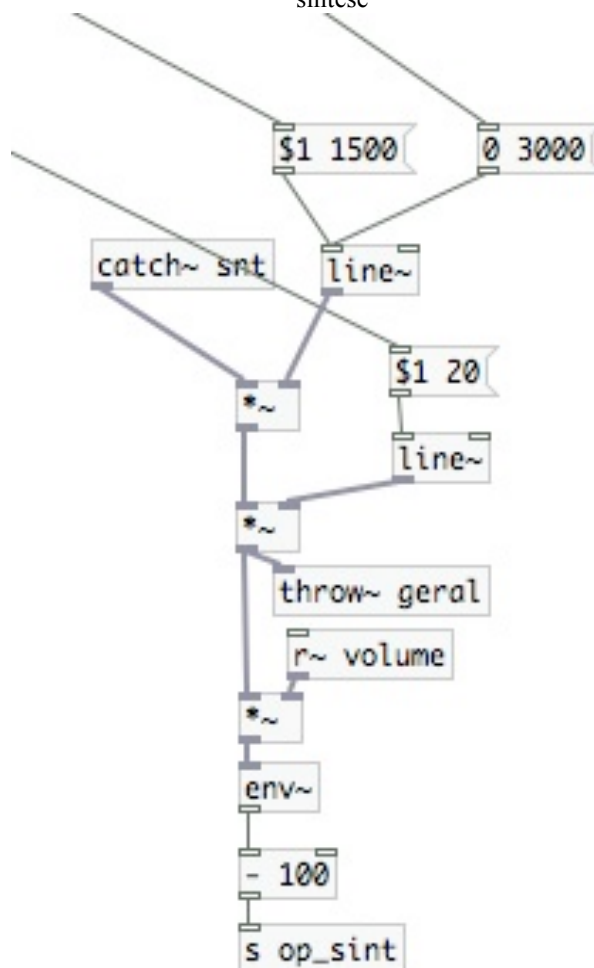


Na Figura 33, o objeto “catch~ *snt*” recebe todo o sinal das sínteses vindos do subpatch “pd *syntheses*”. A caixa de mensagem “\$1 20” recebe valores do *slider* verde “volume” do canvas “SYNTHESES” (ver Figura 18), regulando o nível geral de amplitude dos sons gerados neste canvas, que são então enviadas para o objeto “throw~ *geral*” e depois para o “dac~”.

Conforme dito anteriormente, os quatro módulos de geração sonora do canvas “SYNTHESES” podem funcionar de forma simultânea ou não, o que é ilustrado na Figura 34 através do objeto “random 2” que inicia/continua a geração sonora de determinado módulo ou

a interrompe. Neste caso, o objeto “*random*” realiza esse controle em intervalos de tempo entre 3 e 4 segundos.

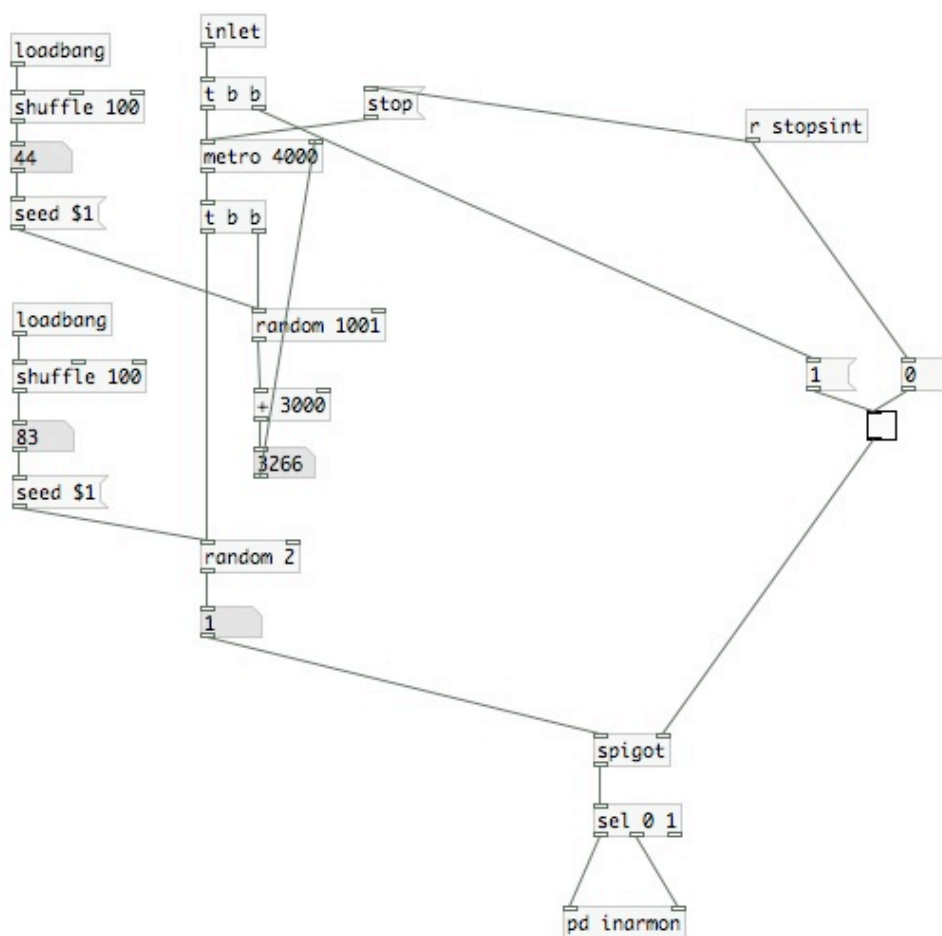
Figura 33 – Imagem da programação do controle da amplitude geral dos dados recebidos dos processamentos de síntese



Cada um dos quatro módulos de geração sonora está presente num *subpatch* semelhante ao apresentado na Figura 34. A diferença entre eles diz respeito ao início ou à interrupção de seus funcionamentos, conforme o controle exercido pelo objeto “*random*”. A periodicidade com que ocorrem as mudanças de estado (funcionamento – interrupção) nos quatro *subpatches* pode variar entre 3 a 4 segundos.

O objeto “*sel*” envia uma mensagem “*bang*” para o *inlet* esquerdo do *subpatch* “*pd inarmon*” quando recebe a mensagem “0” (zero) do objeto “*random 2*” e para o *inlet* direito quando recebe a mensagem “1”, o que liga e desliga os módulos de geração sonora, respectivamente. Em cada um dos quatro *subpatches* semelhantes à Figura 34, o objeto “*sel*” envia mensagens “*bangs*” para os *subpatches* de geração sonora - da mesma maneira que para o *subpatch* “*pd inarmon*”, a saber: “*pd am*”; “*pd textura*”; e “*pd sintese_granular*”.

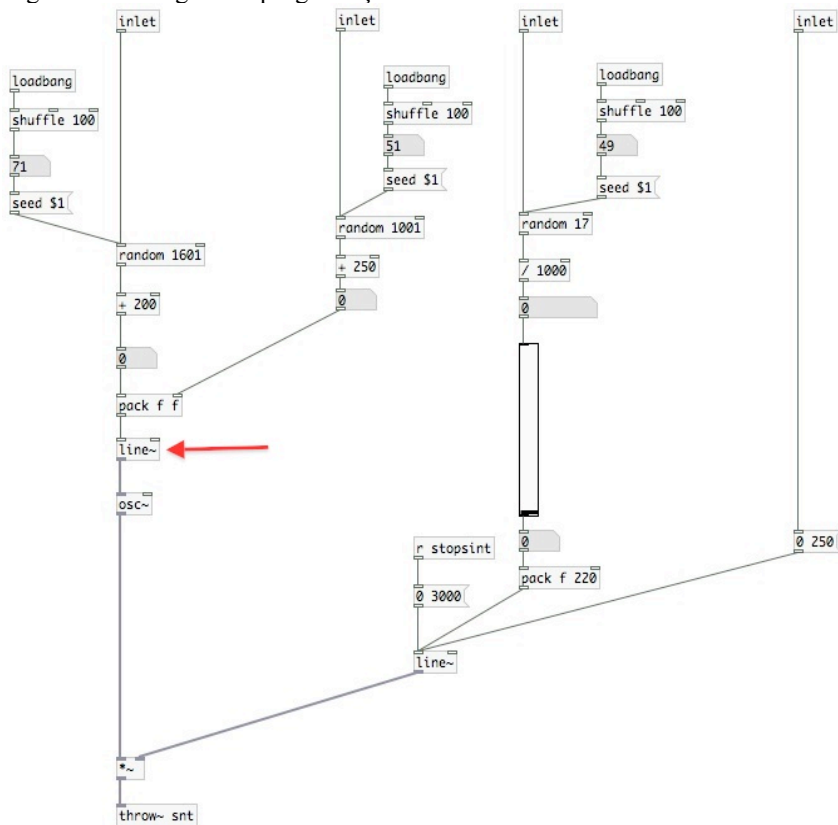
Figura 34 – Imagem de um dos 5 *subpatches* que iniciam, continuam ou interrompem cada um dos 4 módulos de geração sonora no *canvas SYNTHESSES*



3.5.2.2 Subpatch pd inarmon (síntese aditiva)

No *subpatch pd inarmon* existem oito módulos iguais ao da Figura 35, em que o objeto “*line~*” (indicado pela seta) faz uma rampa do último valor recebido pelo *inlet* esquerdo do objeto “*pack*” até o valor atual recebido, os quais são interpretados pelo objeto “*osc~*” como frequências, o que produz *glissandi* como *output* desses *subpatches* (módulos). As frequências mínima e máxima de cada um dos oito módulos são de 200 hz e 1800 hz, respectivamente. Os *glissandi* duram um tempo determinado em milissegundos pelo valor recebido no *inlet* direito do objeto “*pack*”, o qual muda de acordo com uma mensagem “*bang*” enviada a cada 1250 milissegundos pelo *outlet* do segundo objeto “*inlet*”. Quando um valor de frequência é enviado antes de terminar um *glissando*, haverá uma interrupção deste para que a partir deste ponto que foi interrompido comece um novo *glissando* com o valor da frequência atual, contudo, isto não quer dizer que o áudio seja interrompido. Por outro lado, se for enviado um novo valor ao *inlet* direito do objeto “*pack*” antes do término de um *glissando*, este valor só servirá como uma nova duração de *glissando* quando o próximo valor de frequência for enviado ao *inlet* esquerdo do objeto “*pack*”.

Figura 35 – Imagem da programação de um dos oito módulos da síntese aditiva



A cada 1800 milissegundos é enviado um valor pseudorrandômico de amplitude ao terceiro “*inlet*” (visto na Figura 35, da esquerda para a direita) *de cada um dos oito módulos de síntese*. Tal valor dividido por 1000 varia entre 0.001 e 0.016 *RMS* (raiz da média dos quadrados) - utilizado como valor da amplitude de cada um dos 8 módulos – e é alcançado em 220 milissegundos a partir do valor anterior enviado ao objeto “*line~*”. Existe também a possibilidade de um ou mais módulos dentre os oito estarem com amplitude no valor 0 (zero) caso o objeto “*random 17*” envie uma mensagem de valor 0 (zero) - mesmo que o módulo não receba uma mensagem para ser desligado em seu quarto *inlet* (Figura 35) .

É importante observar que o controle dos parâmetros descritos (frequência, duração do *glissando* e amplitude) não ocorrem simultaneamente. Contudo, cada parâmetro é controlado de forma sincronizada entre os oito módulos. Apesar desta simultaneidade do controle do mesmo parâmetro dentro dos oito módulos, a duração do *glissando* dentro de cada um dos oito módulos pode sempre ser diferente, pois depende do valor aleatório gerado pelo objeto “*random 1001*”. Desta forma, esta programação de síntese aditiva dentro do *subpatch pd inarmon* gera uma certa riqueza tímbrica pelo seu conteúdo espectral inarmônico, devido a inúmeras possibilidades de frequências e *glissandos* com probabilidades de diferentes amplitudes, realçando diferentes frequências em momentos imprevisíveis. Segundo Miranda (2002, p. 50-51, tradução nossa), “a técnica [de síntese aditiva] assume que uma forma de onda periódica pode ser modelada como a soma de senóides com envelopes de amplitude variáveis e frequências variáveis no tempo [...] é aceito como talvez o mais poderoso método de modelagem espectral”³.

3.5.2.3 Subpatch pd am (síntese amplitude modulada)

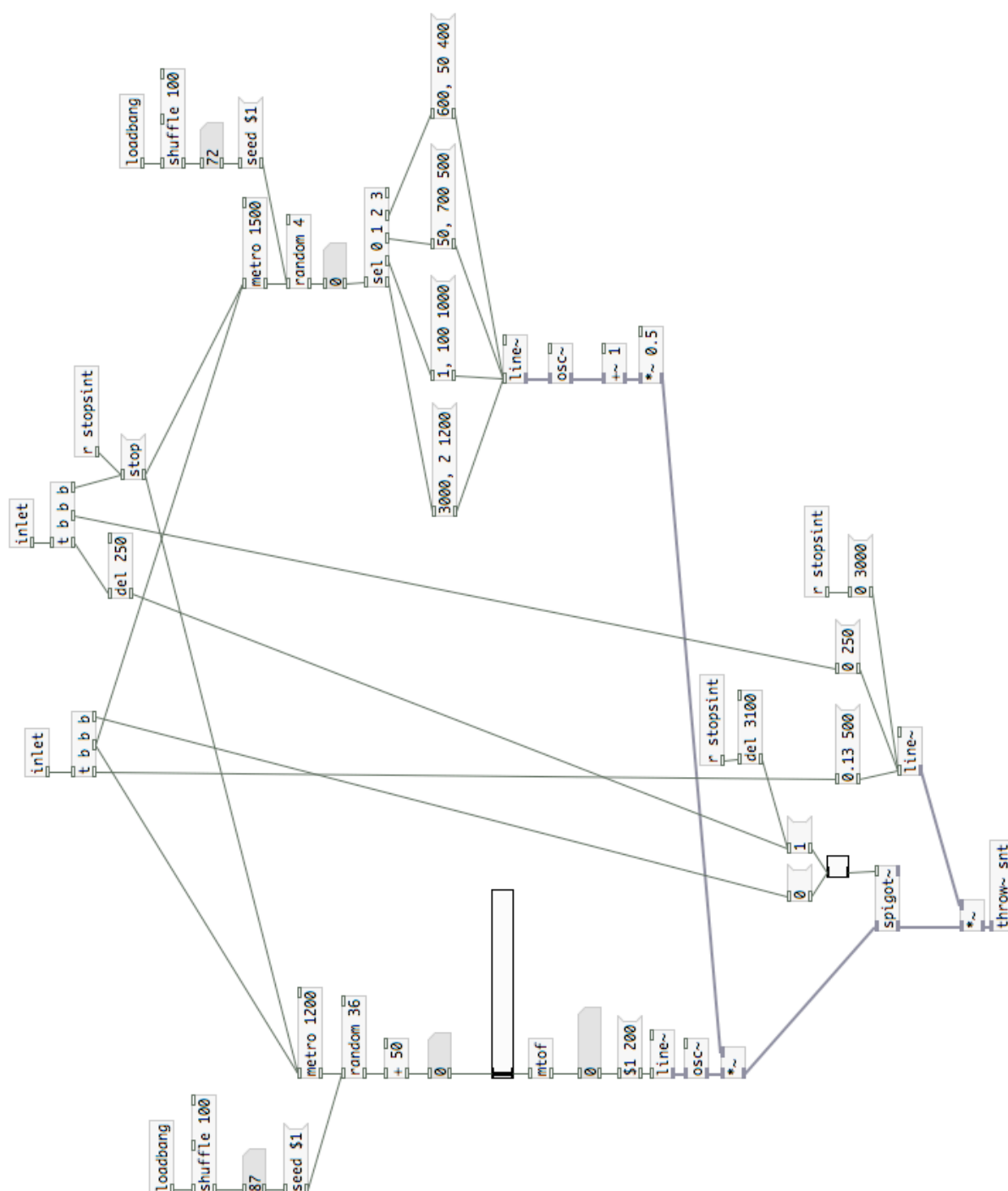
No *subpatch pd am* (Figura 36) encontra-se uma programação para a realização de síntese AM (amplitude modulada) em que o som gerado é sempre composto de três frequências simultâneas. De acordo com Miranda (2002, p. 20), “a modulação ocorre quando algum aspecto do sinal de áudio (chamado de portador) varia de acordo com o comportamento de outro sinal de áudio (chamado de modulador). A modulação de amplitude, portanto, ocorre quando o modulador controla a amplitude do portador”⁴. As três frequências

³ No original: “The technique assumes that any periodic waveform can be modelled as a sum of sinusoids at various amplitude envelopes and time-varying frequencies. [...] is accepted as perhaps the most powerful and flexible spectral modelling method”.

⁴ No original: “Modulation occurs when some aspect of an audio signal (called a *carrier*) varies according to the behaviour of another audio signal (called a *modulator*). Amplitude modulation therefore occurs when a

geradas nesta síntese da programação da Figura 36 possuem os seguintes valores: da frequência do portador; a soma entre a frequência portadora e a frequência moduladora; e a subtração entre a frequência portadora e a frequência moduladora.

Figura 36 – Subpatch pd am: síntese AM (Amplitude Modulada)

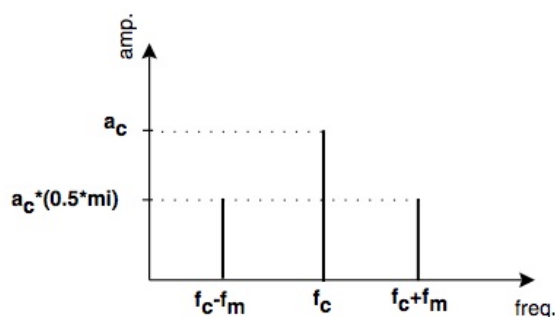


modulator drives the amplitude of a carrier”.

De acordo com o gráfico da Figura 37, temos “ ac ” como amplitude da frequência portadora, “ fc ” como frequência portadora, “ fm ” como frequência moduladora e “ mi ” como índice de modulação. Observa-se no gráfico que a amplitude da frequência portadora é o dobro da amplitude das outras duas frequências. O cálculo para se encontrar a amplitude destas frequências (A_k) é dado pela seguinte fórmula (MIRANDA, 2002):

$$A_k = ac \times 0,5 \times mi \quad (1)$$

Figura 37 – Gráfico “frequências x amplitude” na síntese AM



Fonte: MIRANDA (2002)

Diferentemente da “síntese por modulação em anel” (*Ring Modulation*) - que é um tipo específico de síntese por modulação de amplitude - na síntese por modulação de amplitude “simples” (ou ainda modulação de amplitude “clássica”) há a presença da frequência portadora, o que não ocorre na Síntese “*RM*” (*Ring Modulation*). Isto ocorre pelo fato da frequência moduladora na síntese por modulação de amplitude “simples” modular a frequência portadora apenas com valores positivos (FREE SOFTWARE FOUNDATION, 1991)⁵. Além disso, diferentemente da síntese “*RM*”, na síntese “*AM*”, “[...] a frequência moduladora produz valores de amplitude diferentes de zero” (CYCLING’74, 2012, tradução nossa)⁶. Na Figura 35, nota-se o uso de duas caixas de operação com sinal (“+~1” e “*~0.5”) após a caixa do oscilador no lado direito do *subpatch*. São estas operações que fazem com que o sinal do modulador passe a conter somente valores positivos (neste caso, valores entre 0 e 1).

Ainda na Figura 36, observa-se que do lado esquerdo do *subpatch* estão os valores das frequências que terão sua amplitude modulada, ou seja, os valores da “frequência portadora”. No lado direito do *subpatch* encontram-se os valores da frequência moduladora. No lado

⁵ Informação disponível em: <http://en.flossmanuals.net/pure-data/ch021_amplitude-modulation/>.

⁶ Informação disponível em: <<http://www.cycling74.com/docs/max5/tutorials/msp-tut/mspchapter09.html>>.

esquerdo do *subpatch* (frequência portadora) há um objeto “*line~*” que recebe aleatoriamente a cada 1200 milissegundos valores de frequências entre 147 e 1108 *hertz*, alcançando cada valor recebido por interpolação linear durante 200 milissegundos, produzindo *glissando*. A cada 1500 milissegundos o objeto “*random 4*” seleciona aleatoriamente uma das quatro mensagens que são enviadas ao objeto “*line~*” do lado direito do *subpatch* (frequência moduladora). O objeto “*line~*” faz uma interpolação linear entre os dois primeiros valores da mensagem durante o tempo em milissegundos que é dado pelo terceiro valor. Nota-se que a mensagem “3000, 2 1200” - disparada quando o objeto “*random 4*” envia o número 0 (zero) - produz um efeito de “trêmolo”. Isto se deve ao fato de que quando duas ondas com frequências muito próximas (no máximo 40 *hertz* de diferença, aproximadamente) são somadas ocorre um efeito de “batimento”. Segundo Menezes (2003), quando as cristas de duas ondas A e B se coincidem, ocorre um aumento da amplitude no som resultante por “interferência construtiva”, do contrário, quando o pico mínimo de uma onda coincide com o pico máximo da outra onda,

[...] existe oposição de fase, em interferência destrutiva e anulação do som. Tais momentos de aumento de amplitude seguido de anulação do som resultam numa pulsação em amplitude do som resultante que equivale à diferença entre as frequências de A e B, constituindo os batimentos percebidos pelo ouvido como uma *modulação de amplitude* (MENEZES, 2003, p. 59).

Contudo, como comentado acima, diferentemente da síntese “*RM*”, na síntese “*AM*” não ocorre a anulação do som na oposição de fase das duas ondas, somente um decaimento na amplitude que não chega a zero (CYCLING’74, 2012). Os batimentos podem ser calculados pela diferença entre estas ondas, na qual o resultado será o número de batimentos por segundo. No caso da mensagem “3000, 2 1200” na frequência moduladora, quando esta oscila com uma frequência suficientemente baixa, a diferença e a soma entre a frequência moduladora e a portadora gera frequências tão próximas que o fenômeno dos batimentos vem a tona (Figura 36), sem, no entanto ocorrer a anulação da amplitude.

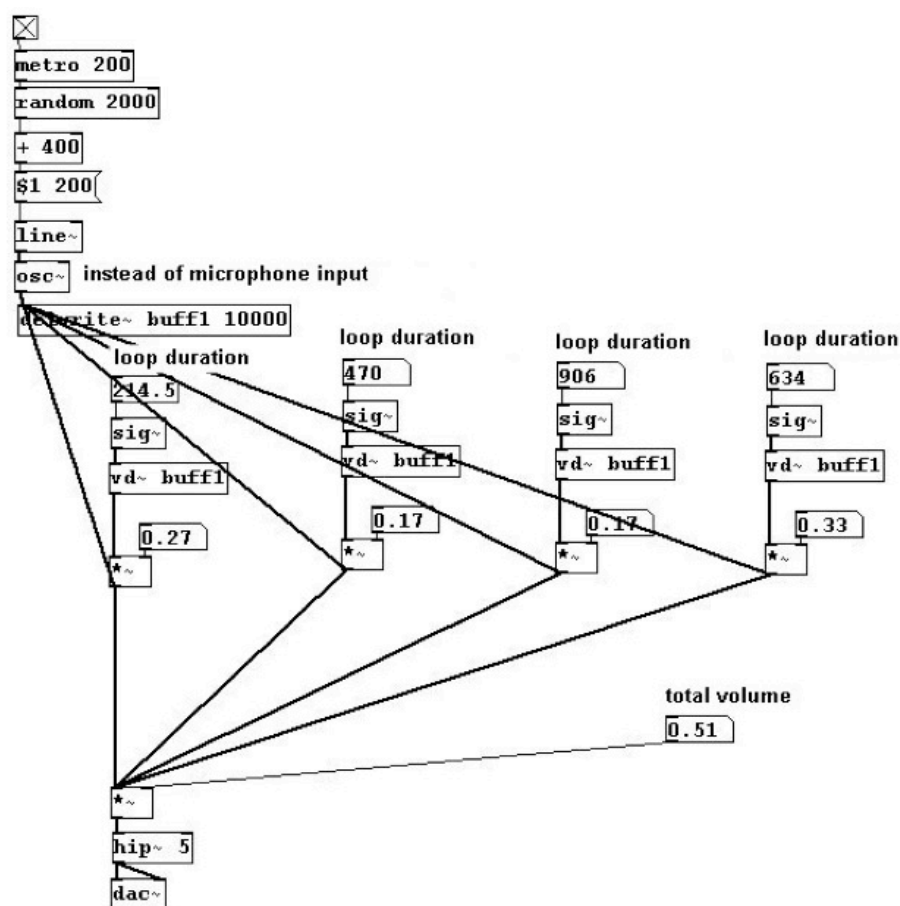
3.5.2.4 *Subpatch pd textura*

O *subpatch pd textura* se configura como uma síntese aditiva, pois através da sobreposição de *loops*, implementados por linhas de *delays* ocorre a soma de ondas senoidais. Em cada linha de *delay* há uma onda senoidal cuja frequência caminha contínua e rapidamente (*glissando*) com base na interpolação linear entre valores de frequências gerados

a cada 200 milissegundos. Este *subpatch* é uma adaptação de um *patch* encontrado na seção “3.4.2.7 – Textura” de Kreidler (2009) (representado aqui na Figura 38). Os *loops* são produzidos pelos objetos “*vd~*” que recebem em seus *inlets* a informação dos tempos dos *delays*, os quais acabam funcionando como durações dos *loops*, por conta do *feedback* implementado. A cada vez que o *subpatch* “*pd textura*” recebe uma mensagem “*bang*” em seu *inlet* esquerdo, (como na Figura 34 na qual ele estaria no lugar do *subpatch* *pd inarmon*), o objeto “*metro*” aciona a geração de valores a cada 200ms pelo objeto “*random*” e, com isso, a duração do *loop* produzido por cada objeto “*vd~*” é mudada, conforme a programação vista na Figura 39.

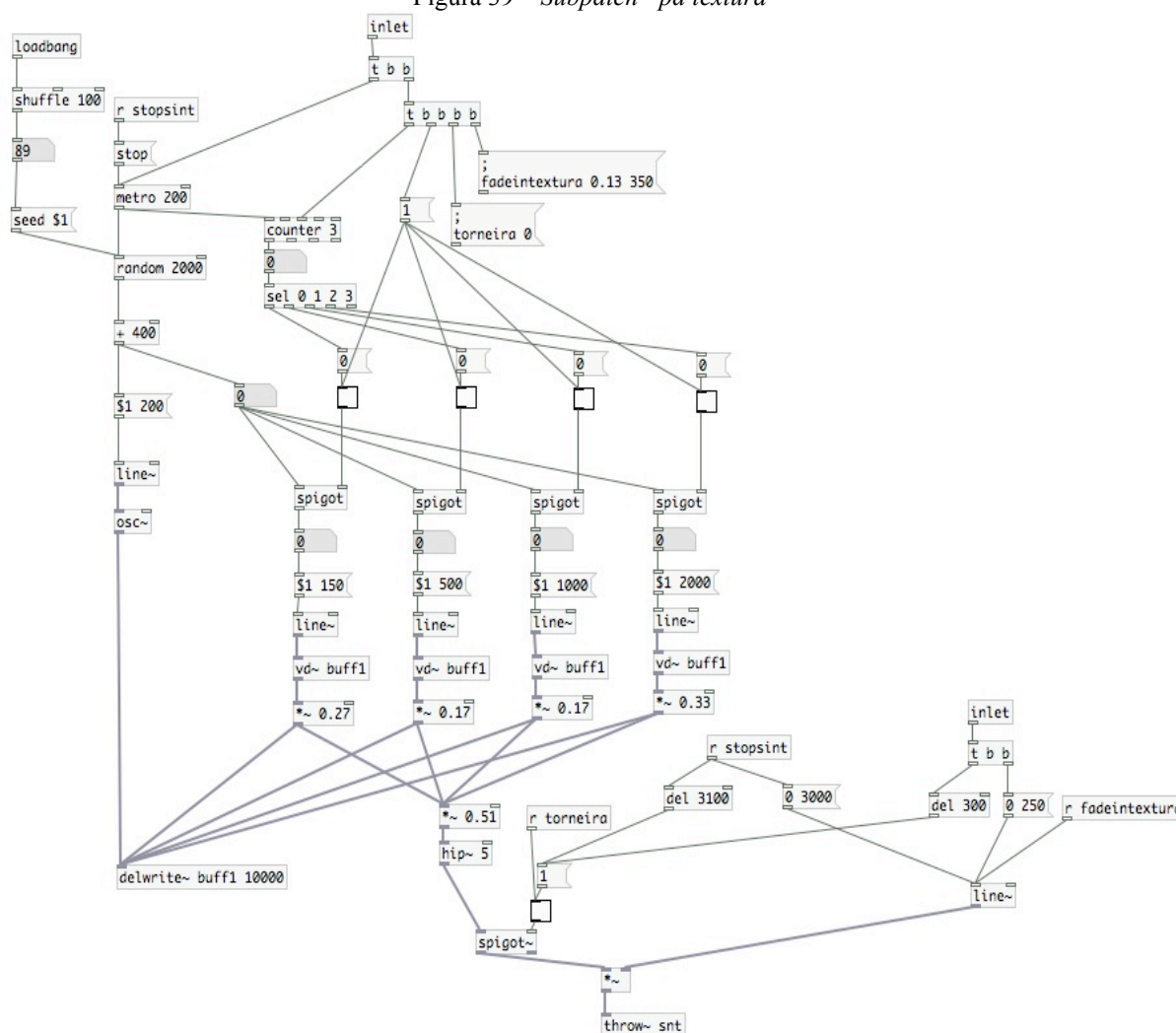
Devido a essa implementação que altera constantemente as durações dos *loops*, foi necessário utilizar o objeto “*line~*” em vez de “*sig~*” (utilizado por Kreidler – ver Figura 38) para que não houvessem “*clicks*” a cada vez que o objeto “*vd~*” recebesse um novo valor de duração do *loop*. Além disso, diferentemente do *patch* da Figura 38, optou-se por mudar a duração de um *loop* de cada vez.

Figura 38 – Patch da “Seção 3.4.2.7 – Textura”, de Kreidler (2009)



Fonte: Kreidler (2009)

Figura 39 – Subpatch “pd textura”



3.5.2.5 Subpatch *pd sintese_granular*

O subpatch “*pd sintese_granular*” possui um módulo de síntese granular cuja programação foi baseada no patch da seção “3.7.3.1 – *Granular Technique as a Synthesizer*” encontrado em Kreidler (2009) – o qual pode ser visto na Figura 40. Segundo Miranda (2009, p. 101, tradução nossa),

A síntese granular gera uma rápida sucessão de excertos sonoros muito curtos chamados grãos (por exemplo, 35 milésimos de segundo) que juntos formam eventos sonoros de maior duração. Os resultados geralmente apresentam um grande senso de movimento e fluxo sonoro. Esta técnica de síntese pode ser metaforicamente comparada com o funcionamento das imagens cinematográficas, em que a impressão de movimento contínuo é produzida pela exibição de uma sequência de imagens ligeiramente diferentes com uma velocidade superior à capacidade de rastreamento dos olhos.⁷

⁷ No original: “Granular synthesis works by generating a rapid succession of very short sound bursts called

Segundo Kreidler (2009), o *patch* da Figura 40 constitui a programação de um sintetizador para criar nuvens de frequências (síntese granular), em que é mais conveniente o uso de um gerador aleatório por meio do objeto “*noise~*”, o qual gera parâmetros de controle dos valores frequenciais aleatórios como argumentos dos osciladores na definição das alturas (*pitch*) dos grãos. Foram feitas algumas modificações neste *patch* para adaptá-lo ao funcionamento do SSInPIC, conforme o *subpatch* “*pd sintese_granular*” visto na Figura 41, que será comentado adiante.

Da mesma forma que os *subpatches* “*pd am*” e “*pd textura*”, o *subpatch* “*pd sintese_granular*” também se encontra encapsulado num *patch* similar ao da Figura 34, no lugar em que aparece o *subpatch* “*pd inarmon*”. Na Figura 41, observam-se dois “*inlets*” que recebem as mensagens “*bang*” do objeto “*sel*” da Figura 34, as quais ligam e desligam o *subpatch*. Nota-se o uso de dois objetos “*samphold~*” que enviam o último valor do sinal recebido por seus *inlets* esquerdos a cada vez que diminuem os valores recebidos nos *inlets* direitos. Desta forma, é conveniente o uso de um objeto “*phasor~*” para enviar sinal ao *inlet* direito do objeto “*samphold~*”, pois “*phasor~*” produz um sinal com valores crescentes de 0 até 1, de forma que quando chega a 1 o próximo valor será 0, ou seja, neste momento, diminui de valor, o que fará com que “*samphold~*” envie o último valor recebido pelo seu *inlet* esquerdo. O argumento do objeto “*phasor~*” é que vai determinar a velocidade com que o “*samphold~*” pode enviar um novo valor.

Apesar do mesmo objeto “*phasor~*” enviar o sinal simultaneamente para os dois objetos “*samphold~*”, o *output* do objeto “*samphold~*” do lado direito do *subpatch* será enviado em um momento diferente do *output* enviado pelo objeto “*samphold~*” do lado esquerdo. Isto ocorre devido a uma mudança de fase no sinal do “*phasor~*” que vai para o lado direito, realizada na programação por meio dos objetos “*+~0.5*” e “*wrap~*”. A utilização de dois objetos “*samphold~*” vão constituir, portanto, dois “estratos granulares”⁸ nesta síntese sonora. Na Figura 42, são mostrados dois perfis de um ciclo de onda dente-de-serra⁹ enviados, simultaneamente, cada um para seu respectivo objeto “*samphold~*” em um recorte temporal de 200 milissegundos. De acordo com Barreiro e Fornari (2010, p. 205),

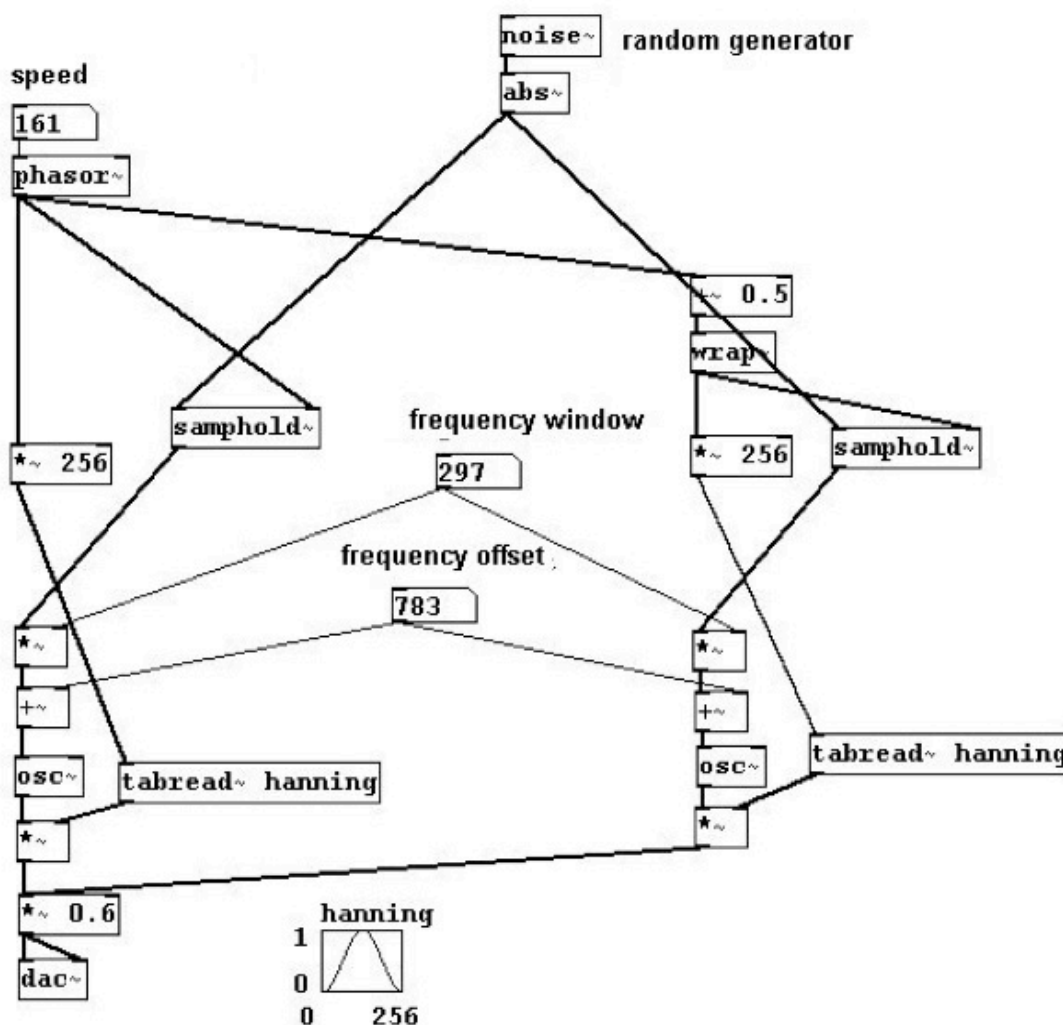
grains (e.g. 35 milliseconds long) that together form larger sound events. The results generally exhibit a great sense of movement and sound flow. This synthesis technique can be metaphorically compared with the functioning of motion pictures in which an impression of continuous movement is produced by displaying a sequence of slightly different images at a rate above the scanning capability of the eye”.

⁸ “Estrato granular consiste numa sequência temporal de grãos. [...]” (BARREIRO; FORNARI, 2010, p. 204).

⁹ É a forma de onda produzida pelo objeto “*phasor~*” que envia sinal com valores crescentes de 0 até 1.

[...] quando se deseja ter mais do que um grão simultâneo (grain overlap), vários geradores de grãos devem ser usados. Isso impõe a necessidade de definir a relação de fase entre os diferentes estratos de grãos. A implementação sem sincronia de fase, encontrada nos sistemas de síntese granular assíncrona, produz estratos que são completamente independentes. Quando o tempo entre os grãos nos diferentes estratos precisa ser controlado, uma abordagem fase-síncrona é necessária.

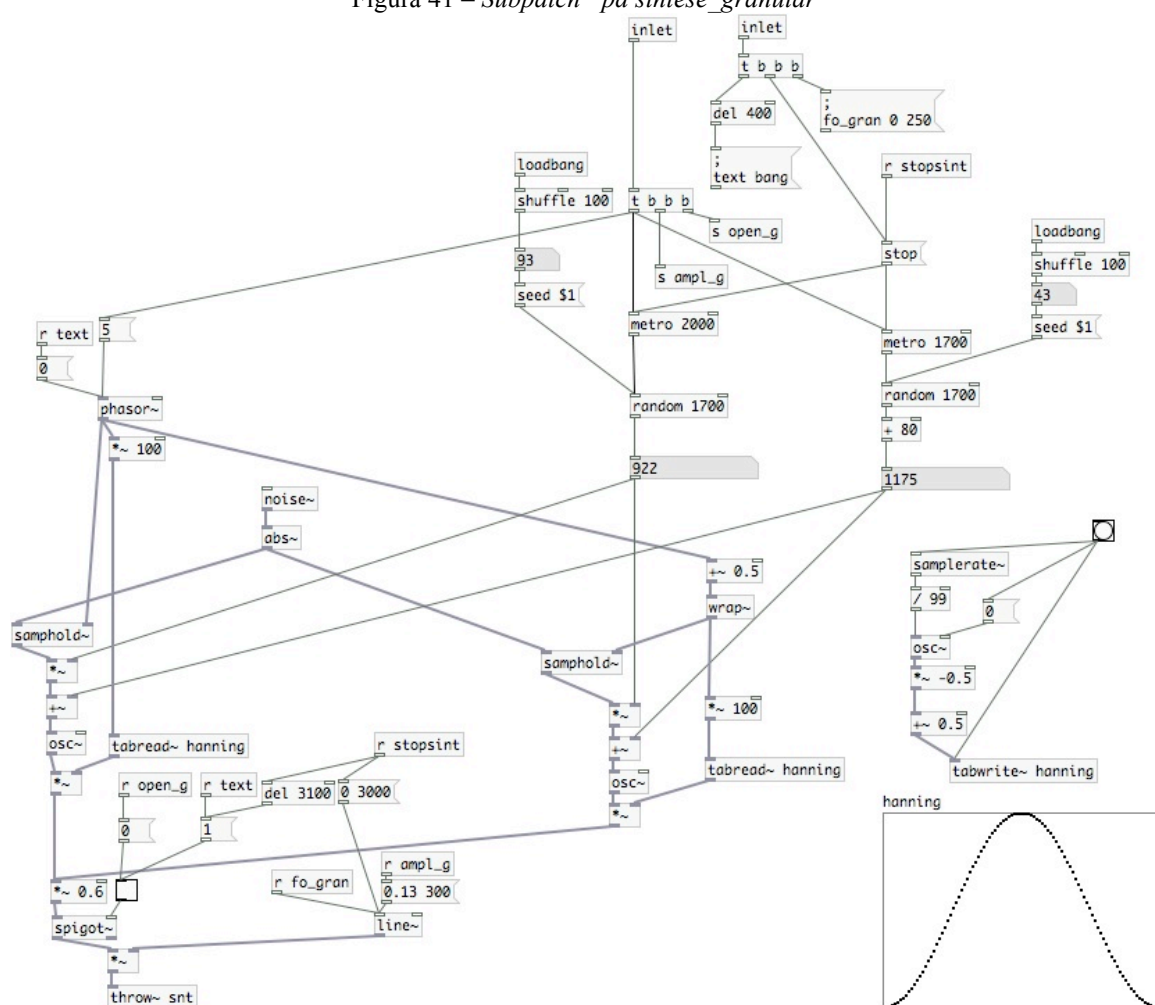
Figura 40 – Patch da “Seção 3.7.3.1 – Granular Technique as a Synthesizer”, de Kreidler (2009)



Fonte: Kreidler (2009)

Como pode ser visto na Figura 41, o valor escolhido como *input* do objeto “*phasor~*” foi 5. Isto significa que serão enviados 5 vezes por segundo valores crescentes de 0 até 1 para os *inlets* direitos dos dois objetos “*samphold~*”, porém, com a diferença de fase, como explicado acima. Desta forma, nos dois estratos granulares com 5 grãos por segundo cada e com defasagem de meio grão entre eles (100 milissegundos), é estabelecida uma “abordagem fase-síncrona” na relação entre suas fases. No entanto, o que vai definir o tamanho do grão é o “janelamento” feito com a curva de envelope “*hanning*” por meio do objeto “*tabread~*”.

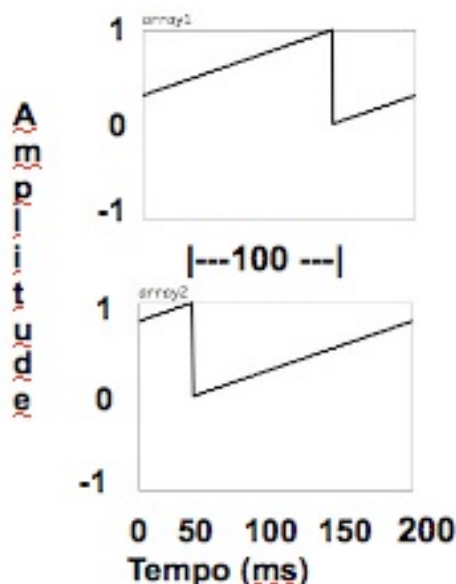
Figura 41 – Subpatch “pd sintese_granular”



Conforme comentado acima, no *subpatch* da Figura 40, o objeto “*noise~*” gera parâmetros de controle nos valores dos argumentos dos objetos “*osc~*” que definem a altura (*pitch*) dos grãos sonoros. O objeto “*noise~*” envia valores entre -1 (menos um) e 1 (um) de forma aleatória, os quais são enviados para o objeto “*abs~*” que envia somente *outputs* com valores positivos, ou seja, quando recebe valores negativos, converte-os em positivos. Estes valores são multiplicados pelo valor da “*frequency window*” e somados com o valor da “*frequency offset*” (Figura 40), cujos resultados determinam as frequências (*pitch*) dos grãos sonoros. Além das alterações realizadas neste *subpatch* da Figura 40 comentadas acima, as quais podem ser vistas no *subpatch* da Figura 41, optou-se por gerar valores aleatórios na “*frequency window*” entre 0 (zero) e 1699 Hz e na “*frequency offset*” entre 80 e 1779 Hz – através de objetos “*random*” -, a cada 2000 milissegundos e 1700 milissegundos, respectivamente. Optou-se por variar os valores das “*frequency window*” e “*frequency offset*” com o intuito de se obter maior variedade nas alturas dos grãos sonoros, o que contribuiu no aumento da complexidade do sistema. Os valores frequenciais dos grãos sonoros estariam,

portanto, entre 80 e 3478 Hz¹⁰.

Figura 42 – Gráficos de comparação das fases das ondas dente-de-serra que chegam simultaneamente ao *inlet* direito de cada objeto “*samphold~*” no *subpatch* *pd* “*sintese_granular*”.



Segundo Miranda (2002, p. 103, tradução nossa) existem três tipos de abordagens técnicas para a realização de síntese granular: a abordagem sequencial (*sequential approach*) em que “a duração dos grãos e os intervalos entre eles são controláveis, mas os grãos não devem se sobrepor”; “a abordagem de dispersão [*scattering approach*] que usa mais que um gerador simultâneo”; e a abordagem de “amostragem granular [*granular sampling approach*] que emprega um mecanismo granulador que extrai pequenas partes de uma amostra sonora, aplicando-as um envelope”. De acordo com a programação descrita do *subpatch* da Figura 41, observa-se que foi empregada a abordagem de dispersão comentada por Miranda (2002), por utilizar dois geradores de grãos sonoros simultaneamente.

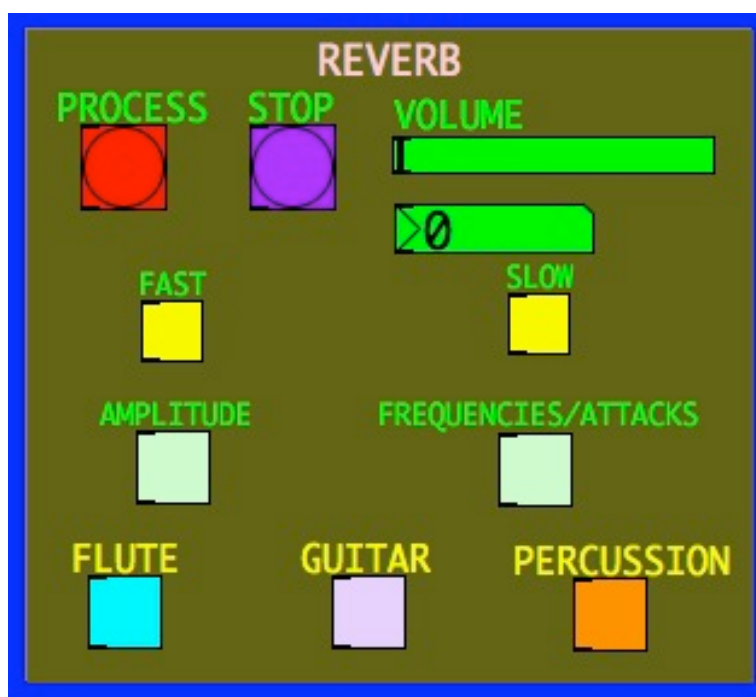
3.5.3 Canvas Reverb

Nem sempre é o som captado pelo microfone que será processado, mas a sua análise poderá gerar dados de controle de uma síntese sonora, por exemplo. No *subpatch* “*rvb*”

¹⁰ Foram feitas alterações nestes valores posteriormente à performance com o SSInPIC - “Atradores Ou-vistos” - registrada em vídeo no DVD anexoado a este trabalho. Esta alteração foi efetuada, pois, na programação utilizada na performance, poderiam ocorrer valores frequenciais dos grãos sonoros fora do âmbito frequencial passível de ser gerado com *sample rate* de 44100. Sobre este assunto ver “Teorema de Nyquist” em CYCLING’74 (2012).

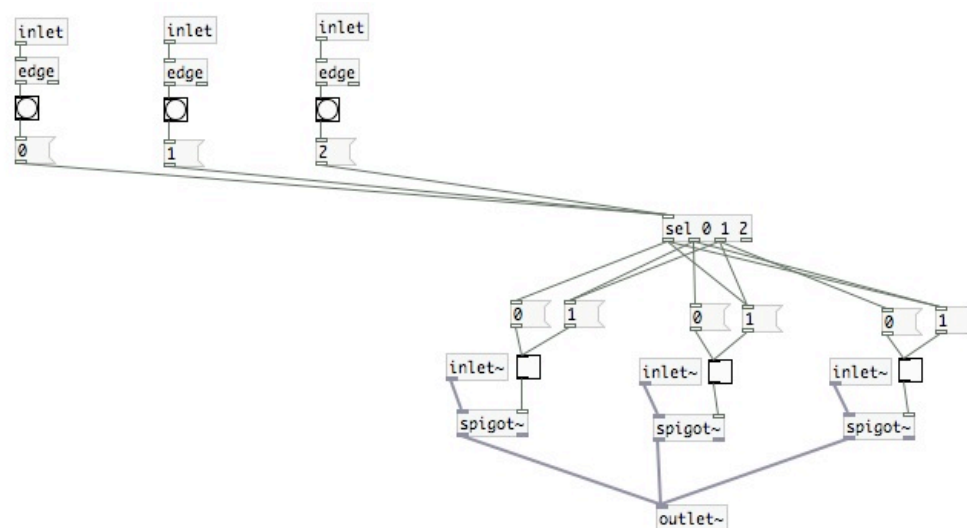
localizado no *canvas* “*REVERB*” acontece esse tipo de procedimento, pois, valores de parâmetros analisados da performance instrumental ao vivo são interpretados como valores frequenciais para uma síntese sonora que simula artificialmente um efeito de *reverb* nestes valores de frequência. Os parâmetros analisados dos instrumentos musicais são a amplitude ou a identificação de ataques e suas frequências, os quais podem ser escolhidos pelo performer de computador através dos *toogles* de cor verde-clara (*amplitude* e *frequencies/attacks*) no *canvas* “*REVERB*” que pode ser visto na Figura 43. A seguir, apresenta-se uma descrição deste módulo de síntese “*REVERB*”.

Figura 43 – Canvas REVERB



Como foi visto na interface gráfica do *patch* do SSInPIC (Figura 18), existem três *toogles* nos *canvas* “*SYNTHESES*”, “*REVERB*” e “*DELAYS*”, que permitem a seleção do sinal do instrumento – *flute*, *guitar* e *percussion* - que será analisado para gerar algum parâmetro da síntese sonora e/ou processado durante a performance. O *subpatch* que envia o sinal dos instrumentos para o processamento ou síntese nos módulos “*DELAYS*” e “*SYNTHESES*” encontra-se na Figura 30. No *subpatch* “*pd rvb*”, o envio dos sinais captados da performance é ligeiramente diferente, conforme pode ser visto no *subpatch* “*pd micro_r*” (Figura 44). As diferenças na programação do módulo “*REVERB*” ocorrem por não haver necessidade de realizar “*fade in*” e “*fade out*” dos sinais de áudio dos instrumentos, pois na programação original de Puckette (1996) isso já tinha sido feito.

Figura 44 – Subpatch *pd “micro_r”*: programação da captação do sinal do instrumento escolhido para interação com o módulo de síntese no subpatch “*pd revb*”



Os *toogles* dos instrumentos - da esquerda para a direita (*flute*, *guitar*, *percussion*) - do *canvas* “*REVERB*” (Figura 43) estão ligados respectivamente aos “*inlets*” – da esquerda para a direita – do subpatch visto na Figura 44. O sinal de áudio do instrumento selecionado é recebido pela única caixa “*inlet~*” do subpatch “*revb*” da Figura 45.

Na Figura 45, são observados oito “*inlets*” além do objeto “*inlet~*”, mencionado logo acima. Da esquerda para a direita, os dois primeiros *inlets* do subpatch “*pd revb*” referem-se aos dois *toogles* amarelos (*fast/slow*) localizados no *canvas* “*REVERB*” (Figura 43). Os *toogles* “*fast*” e “*slow*” enviam, respectivamente, as mensagens 200 e 2000 para o *inlet* direito de dois objetos “*pipe*”, definindo, assim, o atraso em milissegundos com que o fluxo de valores recebidos no *inlet* esquerdo será repassado pelo *outlet*.

O sinal de áudio recebido pelo “*inlet~*” do subpatch “*pd revb*” é encaminhado para o objeto “*fiddle~*”, o qual, de acordo com Puckette, Apel e Zicarelli (1998), calcula a altura e a amplitude do sinal de entrada por meio de uma análise de Fourier, podendo variar a quantidade de *outputs* dependendo dos argumentos que são criados. No subpatch “*pd revb*” os argumentos do objeto “*fiddle*”, da esquerda para a direita, são: 1024 (tamanho da janela de análise em *samples* – sempre uma potência de 2); 1 (quantidade de alturas a serem identificadas simultaneamente – o que corresponderia à quantidade de “vozes”); 20 (número de picos a serem analisados no espectro); 5 (número de parciais que, caso encontrados, deverão ser informados por meio de dados de frequência e amplitude).

Em comparação com o arquivo de ajuda (*help*) do objeto “*fiddle*” no ambiente *Pure*

Data, mudou-se apenas o último argumento citado (número de parciais), conforme o *subpatch* da Figura 45. O segundo *outlet* do objeto “*fiddle~*” detecta ataques e envia uma mensagem “*bang*” a cada ataque detectado. Convém observar que estes ataques são detectados independentemente da identificação ou não de uma altura (PUCKETTE; APEL; ZICARELLI, 1998). O quarto *outlet* apresenta uma análise contínua da amplitude do sinal em decibéis. O quinto *outlet* corresponde ao último argumento do objeto, que analisa, portanto, as frequências e as amplitudes dos cinco primeiros parciais encontrados.

O *subpatch* “*pd analyse*”, encapsulado no *subpatch* da Figura 45, é visualizado na Figura 46 e recebe como *input* em seu *inlet* esquerdo a análise dos cinco primeiros parciais detectados por “*fiddle~*” (quinto *outlet* de “*fiddle~*”), e em seu *inlet* direito recebe as mensagens “*bang*” enviadas pelo segundo *outlet* do objeto “*fiddle*” (que detecta ataques). Na Figura 46, observa-se um objeto “*random 5*” que seleciona randomicamente, a frequência de um dos cinco parciais captados da análise da performance instrumental a cada vez que o objeto “*fiddle~*” detecta um ataque. Contudo, os dados produzidos pelo objeto “*fiddle~*” tem uma velocidade bastante caótica e muitas vezes não seriam adequados para controlar determinado tipo de síntese sonora. Para solucionar este problema, buscou-se uma estratégia que possibilitasse o controle da velocidade dos *outputs* de “*fiddle~*”, de maneira que estes dados se tornassem parâmetros adequados para a síntese sonora. Para isto, utilizou-se um filtro “passa-baixa” com o objeto “*lop~ 1*” que também pode receber dados e não apenas sinais (KREIDLER, 2009). Além disso, utilizou-se o objeto “*snapshot~*” para controlar ainda mais a velocidade dos dados produzidos por “*fiddle~*”, além de transformar o sinal enviado por “*lop~ 1*” em dados, novamente. Em vez de usar um objeto “*metro*” para controlar a velocidade com que “*snapshot~*” envia os dados recebidos, utilizou-se as próprias mensagens “*bang*” enviadas pelo segundo *outlet* do objeto “*fiddle~*” quando este detecta ataques vindos da performance instrumental. Desta forma, como será comentado posteriormente, os músicos passam a ter um maior controle da síntese, já que, somente quando produzem ataques, é que se envia um valor aproximado de uma frequência identificada para servir de parâmetro para a síntese sonora.

Figura 45 - *Subpatch* “*pd revb*” (controle do módulo de síntese *REVERB*)

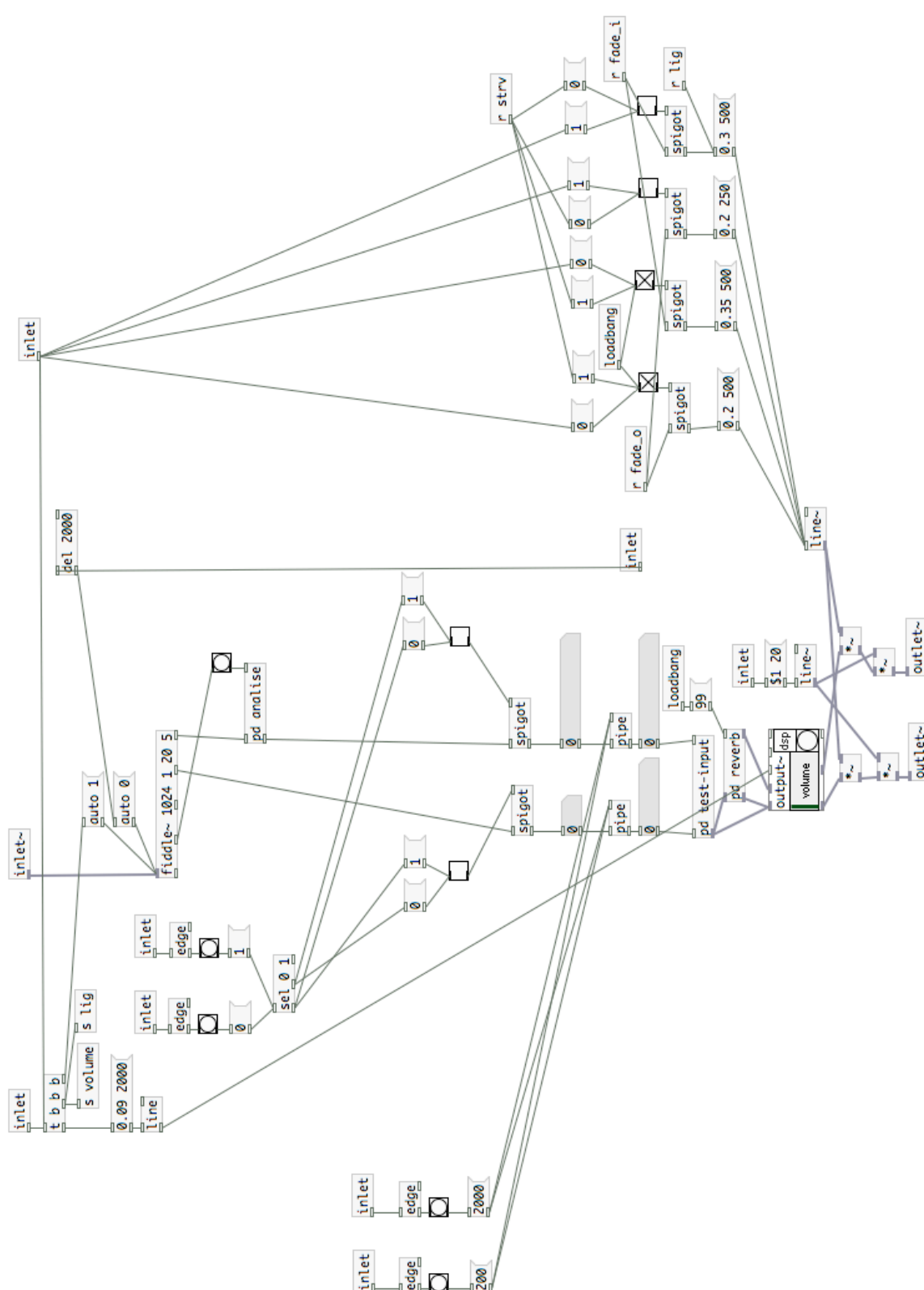
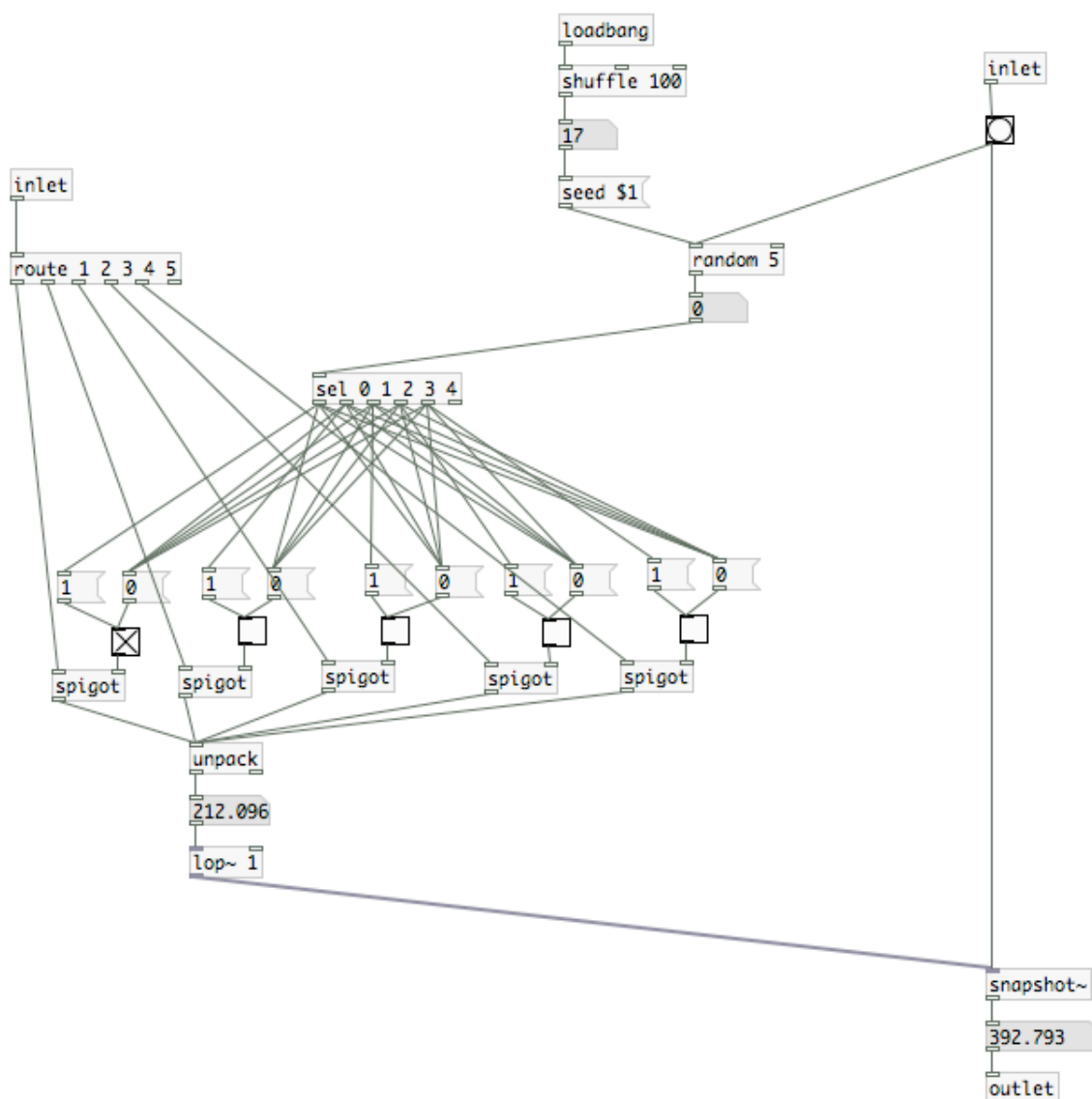


Figura 46 – Subpatch “pd analyse”



Como dito acima, o objeto “*fiddle~*” também analisa a amplitude de um sinal de áudio, informada pelo quarto *outlet*. Estes dados de amplitude também serão interpretados como parâmetros frequenciais da síntese sonora. Contudo, a velocidade caótica desses dados de amplitude pelo objeto “*fiddle~*” não será controlada como ocorre com a programação feita na análise das frequências. Desta forma, a análise e o controle da velocidade destes dois parâmetros (frequência e amplitude) sobre o mesmo tipo de síntese terão resultados sonoros diferentes. De acordo com os experimentos realizados, a geração sonora realizada com valores da análise das frequências terão características mais agudas e rítmicas, e a realizada com os valores da análise de amplitude, mais graves e contínuas.

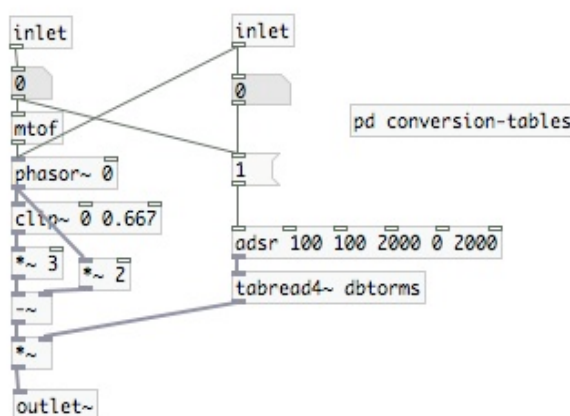
Os valores destas análises são enviados ao *patch Reverberator* (G08.pd.reverb) encontrado na biblioteca do *Pure Data* na pasta 3 “*audio examples*” – ver Puckette (1996).

Este *patch* é constituído do *subpatch* *pd “test-input”* (que contém a abstração¹¹ “*adsr.pd*”), do *subpatch* “*pd reverb*” (que contém a abstração “*reverb-echo.pd*”) e da abstração “*output~.pd*” – todos eles incorporados à programação do “REVERB” discutido neste tópico.

3.5.3.1 Patch *G08.pd.reverb* e abstrações

Os valores de amplitude da performance instrumental analisados pelo objeto “*fiddle~*” no *subpatch* da Figura 45 são enviados ao *subpatch* *pd “test-input”* (Figura 47), onde são recebidos por um objeto “*mtof*” que transforma números MIDI em frequência¹². Desta forma, se os valores aproximados da análise da amplitude estão entre 30 e 100 (em decibéis), por exemplo, quando interpretados como números MIDI serão transformados pelo objeto “*mtof*” em valores de frequência entre 45 e 2640Hz, aproximadamente. Uma das modificações realizadas no *subpatch* *pd “test-input”* em relação ao seu original foi a adição de outro *inlet* (no lado direito do *subpatch*) que recebe os valores de frequência de um dos cinco primeiros parciais do sinal da performance instrumental analisado (a cada ataque) pelo objeto “*fiddle~*”. A adição deste *inlet* teve o objetivo de fazer com que os valores de frequências recebidos não passassem pelo objeto “*mtof*” - como pode ser observado na Figura 47 - para que não fossem interpretados como valores MIDI, o que resultaria em frequências muito agudas na geração sonora, indesejáveis como *outputs*.

Figura 47 – Subpatch “*pd test-input*”



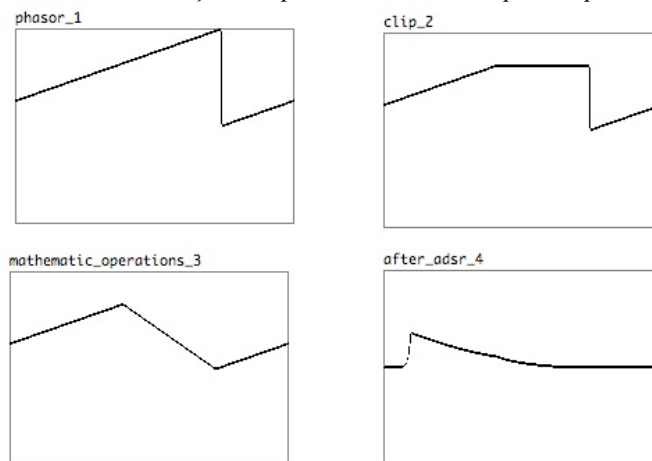
Fonte: Puckette (1996)

¹¹ “Abstrações são *patches* que assumem funções de objetos externos e podem ser usados por vários *patches* diferentes ao mesmo tempo.” (FIGUEIRÓ; KROGER, 2007, p.8).

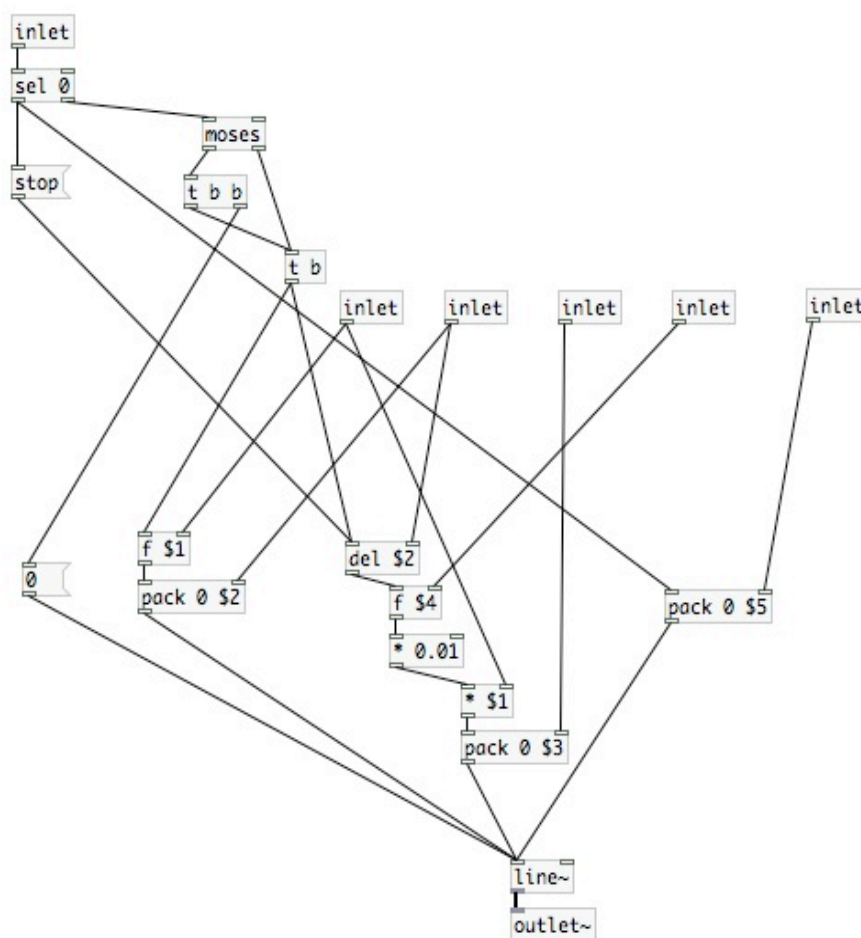
¹² Por exemplo, quando o objeto “*mtof*” recebe o número MIDI 69 correspondente à nota Lá 3, seu output terá um valor de 440, que é a frequência em Hertz da nota Lá 3.

Os valores analisados pelo objeto “*fiddle~*” são enviados ao objeto “*phasor~*” (Figura 47), onde definem a frequência de uma onda dente-de-serra com valores entre 0 (zero) e 1 (um). Estes valores enviados para o objeto “*clip~ 0 0,667*” são limitados até o valor máximo de 0,667 (crista da onda) e, passando em seguida por objetos de operações matemáticas, muda o perfil da “onda dente-de-serra” para um perfil de “onda triangular”, com seu sinal oscilando apenas na parte positiva. Na figura Figura 48, observam-se as transformações (de _1 a _4) da onda dente-de-serra até o último perfil de onda multiplicado pela abstração “*adsr.pd*”. A Figura 49 mostra a programação da abstração “*adsr.pd*”, a qual define o perfil do envelope dinâmico do sinal enviado como *output* do *subpatch* “*pd test-input*”. A cada *input* recebido no *subpatch* “*pd test-input*” (Figura 47), ou seja, a cada valor de frequência, é enviado o valor 1 (um) pela caixa de mensagem à abstração “*adsr.pd*”, produzindo uma rampa ascendente de 0 (zero) a 100 (cem) pontos em 100 milissegundos e, logo em seguida, fazendo uma rampa descendente de 100 (cem) a 0 (zero) em 2000 milissegundos. Estes 100 pontos da rampa serão percorridos no array “*dbtorms*” da Figura 50, o qual se encontra dentro do *subpatch* “*pd conversion-tables*” que aparece na Figura 47.

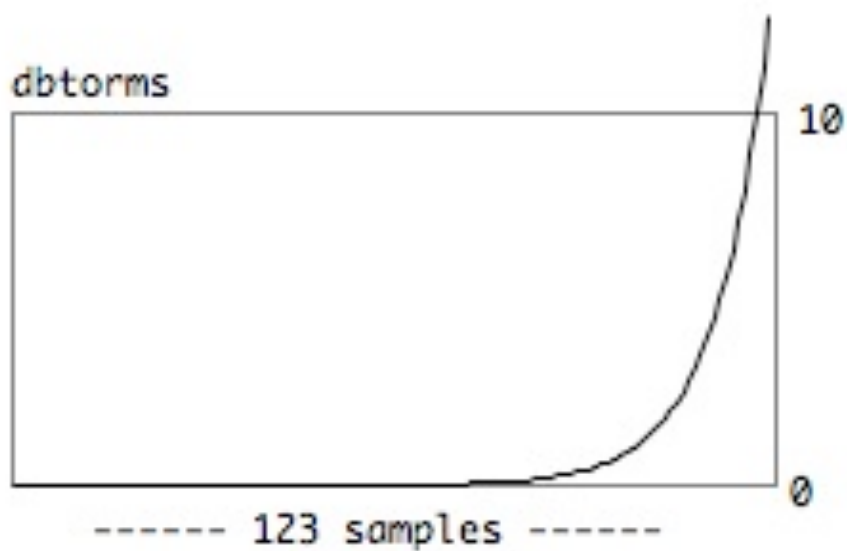
Figura 48 – Transformações do perfil de onda no *subpatch* “*pd test-input*”



No gráfico da Figura 50, pode-se observar que os últimos 100 pontos (*samples*) do eixo horizontal são aqueles em que os valores do eixo vertical começam a se elevar mais rapidamente. Estes últimos valores são enviados de forma crescente e estão compreendidos entre 0 e 12 aproximadamente, como pode ser observado no gráfico da Figura 50. Para o cálculo da curvatura do gráfico da Figura 50, o programador do *patch G08.Reverb.pd* utilizou o objeto “*dbtorms*” que interpreta valores em decibéis e os transforma em *RMS* (raiz da média dos quadrados), conforme a Figura 51.

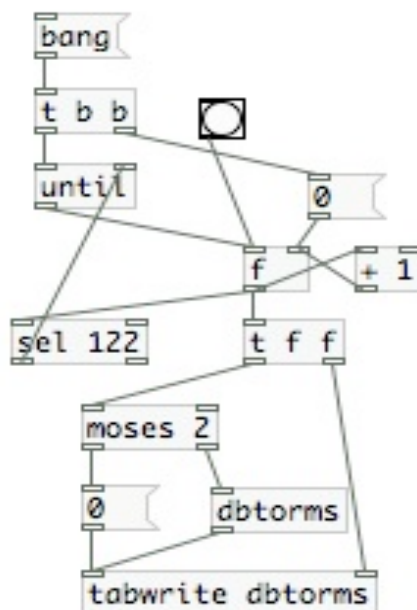
Figura 49 – Abstração “*adsr.pd*”

Fonte: Puckette (1996)

Figura 50 – Array “*dbtorms*”

Fonte: Puckette (1996)

Figura 51 – Imagem da programação para a construção do gráfico do array “dbtorms



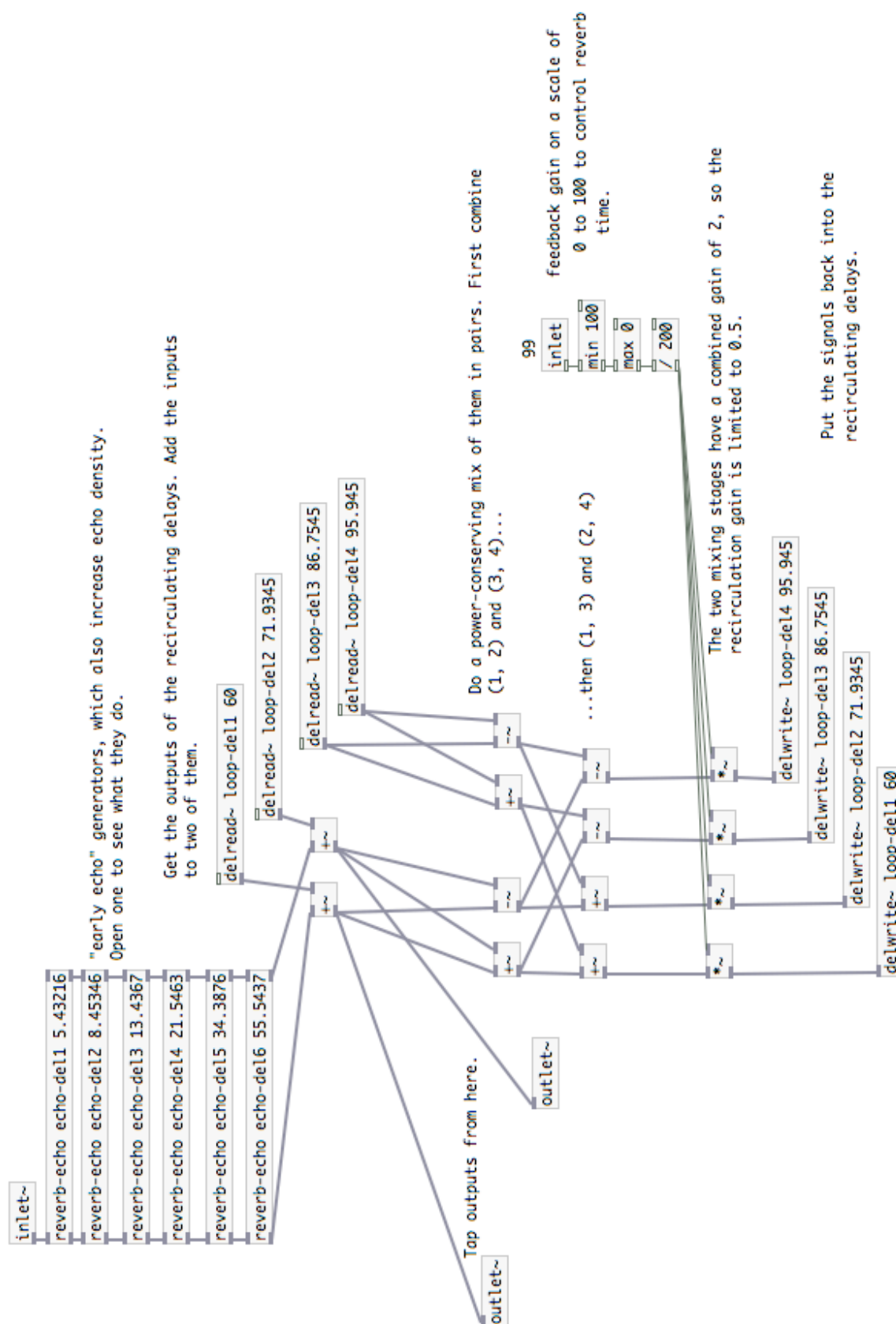
Fonte: Puckette (1996)

Portanto, os valores de *output* do objeto “~” da Figura 47 – entre 0 (zero) e 0,667 – são multiplicados por valores de 0 (zero) a 12 (doze) aproximadamente no final da rampa ascendente que é executada pela abstração “*adsr.pd*”. Este processo é realizado por intermédio do objeto “*tabread4~ dbtorms*” (Figura 47), que envia estes valores ao objeto de multiplicação de sinal. Esta multiplicação acontece a cada vez que o *subpatch* “*pd test-input*” recebe um novo *input*, e o sinal gerado como *output* (com ondas de perfil semelhante a “*after_adsr_4*” da Figura 48) – é então enviado ao *subpatch* “*pd reverb*” – cuja programação pode ser vista na Figura 52.

Observa-se na programação do *subpatch* “*pd reverb*” (Figura 52) - que faz parte do *patch* *G08.reverb.pd* - seis cópias da abstração “*reverb-echo.pd*”, a qual realiza duas cópias do sinal de entrada com diferença de fase, tornando a potência do sinal de saída exatamente o dobro do sinal de entrada. Isto faz aumentar a densidade do eco através das diferentes durações das linhas de *delays*, sendo que um dos canais é atrasado em relação ao outro (PUCKETTE, 1996)¹³.

¹³ Na performance com o SSInPIC registrada em DVD nos Anexos deste trabalho, o sinal gerado como *output* do *subpatch* do *canvas* “*REVERB*” foi apenas duplicado nas duas caixas de som, pois, inicialmente, o efeito que mais interessava no *patch* *G08.pd.reverb* de Puckette (1996) não era tanto o *reverb* - potencializado pelo atraso de um sinal em relação ao outro nos dois canais - mas sim, a obtenção de um efeito sonoro longo, de duração estirada. Contudo, na versão final do aplicativo do SSInPIC, optou-se por manter a versão de Puckette (1996) em relação à programação do atraso de um sinal em relação ao outro em dois canais, proporcionando um maior efeito de reverberação nas respostas sonoras do *subpatch* do *canvas* “*REVERB*”.

Figura 52 - Subpatch "pd reverb"

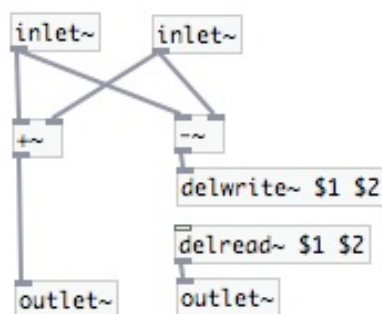


Fonte: Puckette (1996)

A Figura 53 mostra a abstração “*reverb-echo.pd*” (que aparece encapsulada na Figura 52). Observa-se que a parte inteira dos números que servem de argumentos nas cópias da abstração (os quais definem os tempos dos *delays* em milissegundos), são baseados na série de Fibonacci, a saber: “ $5 + 8 = 13$; $8 + 13 = 21$; $13 + 21 = 34$; $21 + 34 = 55$.”

Ainda na Figura 52, observa-se um objeto “inlet” do lado direito que recebe o argumento que controla o ganho do *reverb* que se dá por “quatro linhas de *delays* alimentadas por seu sinal de entrada e pelo seu próprio sinal de saída recirculado” (PUCKETTE, 2007, p. 218, tradução nossa). Este argumento foi definido no valor máximo de “99” pelo programador do *patch* do SSInPIC.

Figura 53 – Abstração “*reverb-echo.pd*”



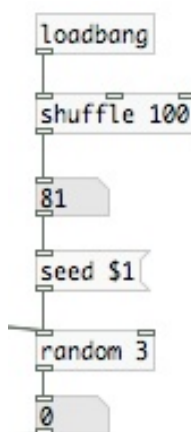
Fonte: Puckette (1996)

* * *

É digna de nota uma observação sobre a programação do objeto “*random*”. Este objeto produz uma série de números pseudorrandômicos através de uma complexa equação (PUCKETTE, 1996). Por meio da definição do valor de uma das variáveis desta equação, é possível prever a sequência de números que objetos “*random*” com mesmo argumento irão produzir. A definição do valor desta variável pode ser enviada ao objeto “*random*” através da mensagem “*seed*”. A cada vez que se envia um mesmo valor de “*seed*” para objetos “*random*” com um mesmo argumento, a sequência de números obtida como *output* será a mesma. Cada objeto “*random*” quando criado já possui um valor de “*seed*” embutido. A cada vez que se fecha e abre novamente o *patch*, o valor da “*seed*” embutido nos objetos “*random*” é modificado. Contudo, quando se fecha e abre novamente o *Pure Data*, o valor da “*seed*” embutido em “*random*” não é modificado e, portanto, produzirá a mesma lista de números de antes. Isto pode se tornar um problema quando se desejar que os processamentos se

comportem de maneira randômica a cada performance. Como na performance de um sistema interativo isso torna-se pertinente quando se deseja um comportamento mais imprevisível do sistema, elaborou-se uma estratégia na programação para que os objetos “*random*” não produzissem sempre a mesma lista de números, a cada vez que se abrisse novamente o *Pure Data* para uma nova performance com o sistema. Na Figura 54, pode-se ter um exemplo da programação para todos os objetos “*random*” que foram utilizados no aplicativo.

Figura 54 – Imagem de uma programação para a geração de uma lista de números não-previsível a cada vez que é inicializado o *Pure Data*



Na Figura 54, o objeto “*shuffle*” produz um número randômico entre 0 e 100 a cada vez que o *patch* (ou o próprio *Pd*) é aberto. Tal número servirá de argumento para a mensagem “*seed*” enviada para o objeto “*random*”. Uma possibilidade poderia ter sido utilizar o objeto “*shuffle*” no lugar do “*random*”. No entanto, julgou-se que essa não seria uma alternativa interessante, uma vez que o objeto “*shuffle*” não repete um número até que todos os outros números possíveis (dentro do limiar definido pelo argumento) tenham ocorrido. Portanto, utilizando-se o *patch* da Figura 54, é possível que a sequência de números gerada pelo objeto “*random*” seja realmente imprevisível a cada abertura do aplicativo.

* * *

A discussão de conceitos envolvidos com sistemas sonoro-interativos realizada no Capítulo 1, as potencialidades e as modelagens de interação apresentadas no Capítulo 2, e a descrição da programação do aplicativo no Capítulo 3 são as ferramentas necessárias para a consolidação desta pesquisa na análise de uma performance com o SSInPIC, a qual é apresentada no Capítulo 4.

CAPÍTULO 4 – OFICINAS E PERFORMANCE COM O SSInPIC

Neste capítulo, será feita uma análise da performance da composição interativa “Atratores Ou-vistos” (PEREIRA, 2011b) apresentada com o SSInPIC no dia 4 de novembro de 2011 na Universidade Federal de Uberlândia. Esta análise se baseia no registro realizado em vídeo¹ e no registro do áudio com a utilização de *software* multipista e microfones. Participaram da performance três instrumentistas – flautista, violonista e percussionista – alunos² do curso de música da Universidade Federal de Uberlândia e o próprio autor desta pesquisa que, além de ter realizado a programação do aplicativo, atuou como *performer* de computador durante os ensaios e a apresentação realizada.

Todos os participantes tiveram experiências musicais anteriores com INI (Improvisação Não-Idiomática), o que conforme comentado no Capítulo 1, seria a abordagem musical mais adequada para a sonoridade pretendida com um sistema sonoro-interativo, de acordo com as definições de Dobrian (2004) a respeito de sistemas sonoro interativos. Desta forma, os *performers* envolvidos necessitavam conhecer as possibilidades musicais que a tecnologia envolvida no SSInPIC oferecia, o que se deu através de algumas oficinas que também viabilizaram aperfeiçoamentos do aplicativo. Da mesma forma que no trabalho de Campos e outros (2006, p. 5), verificou-se que os músicos envolvidos na performance de “Atratores Ou-vistos”

[...] desenvolveram a capacidade de ‘prever’ a reação do computador aos seus estímulos sonoros e assim, no momento de performance, antecipar a ação do outro instrumentista. Ou seja, desenvolveram uma paleta de gestos sonoros e estruturas improvisadas para se adaptar a determinados gestos do outro instrumentista ou a determinada resposta sonora do computador [...]. Este senso de equilíbrio musical, por parte dos intérpretes e compositor, é único. Assim a realização da mesma obra por outros instrumentistas irá resultar em uma nova estruturação e, consequentemente, em novas sonoridades, diretamente vinculadas às experiências assimiladas nas oficinas de preparação e relacionadas ao conhecimento e à vivência musical de cada um.

O título da composição interativa “Atratores Ou-vistos” é baseado na característica de sistemas auto-organizados em que, através da interação entre seus elementos, geram-se “atratores”, os quais imprimem um comportamento emergente no sistema. A palavra “Ou-vistos” foi criada com a junção das palavras “ouvidos” e “vistos”, como também “ou” e “vistos”, assim como fez Lewis Carrol em suas obras literárias com a criação de “palavras

¹ Disponível em: <http://www.youtube.com/watch?v=htWkwEOy_hQ>.

² Flautas: Janaína Nóbrega; Violão: Danilo Aguiar; Percussão: Lúcio Scaranaro.

esotéricas” (DELEUZE, 2003). Desta forma, o significado do título da composição é “Atratores Ouvidos e Vistos” ou ainda “Atratores ou Vistos”. A palavra “vistos” refere-se à ação dos instrumentistas de “visualizarem” a informação - através de um monitor de vídeo - de qual instrumentista deve estimular os sons eletroacústicos, pois quando ele recebe essa informação algum parâmetro é captado e analisado da sua execução instrumental. Além disso, refere-se ao *performer* computacional que além de “ouvir” os gestos instrumentais e a reação dos sons eletroacústicos a esses gestos, também os “visualiza”, o que influencia em suas decisões - qual instrumentista controla os sons eletroacústicos e qual processamento aplicar. Dessa forma, tanto a “escuta” quanto a “visão” do *performer* computacional contribuem na formação de atratores. Por outro lado, a palavra “ouvidos” também refere-se à “escuta” que os instrumentistas tem do processamento eletrônico, ou seja, quando “veem” o seu número no monitor (comando para interação), “escutam” que tipo de processamento ocorre para escolher que tipos de gestos instrumentais poderão melhor influenciar o comportamento eletrônico na busca de um atrator. Além disso, “olhares” trocados entre os instrumentistas podem contribuir na formação de atratores que buscam a manutenção do “ambiente sonoro” da improvisação livre.

As análises apresentadas neste capítulo se basearam nos dados dos capítulos anteriores e em uma metodologia que utiliza, tanto o vídeo que registrou a performance, quanto os arquivos de gravação do áudio em pistas separadas feita no *software Reaper v4.02*³. A gravação em pistas separadas auxiliou o pesquisador a identificar momentos importantes na performance, principalmente aqueles em que amostras sonoras eram gravadas e, posteriormente, disparadas de forma imprevisível pelo algoritmo. Essa ferramenta da gravação multi-pista permitiu a escuta de cada pista de gravação dos sons do *patch* e dos sons dos instrumentos em separado, evitando, assim, a incerteza que poderia haver - no resultado final da performance gravada - entre a identificação de um disparo de amostra gravada na performance e da gravação da própria performance dos instrumentistas que ocorria ao vivo. No vídeo da performance podia ser feita uma comparação dos gestos do instrumentista no momento do disparo de alguma amostra que, por ventura, haveria sido gravada de sua própria execução em algum momento anterior. Assim, era possível identificar a sua interação com a amostra que ele próprio havia produzido num momento anterior, vinculando, assim, passado e presente.

Na primeira seção deste capítulo é apresentada a estrutura da performance que,

³ Disponível em: < <http://www.reaper.fm/download.php> >.

corroborando com Debrun (1996), faz parte das “condições iniciais” de um sistema auto-organizado. Na segunda seção “Análise das Interações Dinâmicas entre *Performers* e Sistema Eletrônico na Performance de ‘Atratores Ou-vistos’”, é apresentado - baseado no registro videográfico - um quadro com todas as amostras sonoras disparadas na performance com marcações temporais. Neste quadro, tais amostras são relacionadas a determinados aspectos da emergência de atratores. Na segunda seção também são apontados os cálculos das probabilidades, dos graus de entropia e redundância das amostras sonoras pré-gravadas que foram disparadas na performance de “Atratores Ou-vistos”. Na terceira seção é feita uma análise das interações estáticas que ocorrem em tempo-real entre os *performers* e o *patch*, descrevendo momentos específicos - marcados no tempo de acordo com o registro em vídeo – identificando a ocorrência de atratores, adaptações, ruídos informacionais e redundância, com o objetivo de observar ocorrências de comportamento emergente do sistema.

4.1 Estrutura da Performance de “Atratores Ou-vistos”

Como o SSInPIC possibilita estruturas variáveis de performance, em sua primeira apresentação pública, o *performer* computacional optou por definir *a priori* uma estrutura para a performance como uma medida de “ordem” no sistema, o que resultou na composição interativa “Atratores Ou-vistos”. Contudo, essa composição ainda possui um alto grau de abertura musical, devido à INI dos instrumentistas e às respostas eletroacústicas do *patch*, que terão resultados sonoros indeterminados a cada nova apresentação da peça. Definiu-se uma duração total “aproximada” de 14’30”, como também a sequência e a duração aproximada de cada módulo do aplicativo que possibilitasse uma interação estática (processos reativos) com os *performers* (“*DELAYS*”, “*SYNTHESES*” e “*REVERB*”). Na análise videográfica, observou-se de forma mais exata os pontos em que ocorreram essas divisões (Figura 55), as quais tiveram suas durações aproximadas estabelecidas *a priori*.

Observa-se que a performance de “Atratores Ou-vistos” possui uma “forma em arco” - conforme pode-se observar nas setas da Figura 56 que indicam a seguinte estrutura: amostra (introdução); *delays*; *syntheses*; *reverb*; *syntheses*; *delays*; amostra (coda). A marcação temporal dos pontos de mudança de uma parte para outra dessa estrutura podem ser observados na Figura 55 acima.

A sequência observada nas figuras 55 e 56 foi executada na performance por meio do acionamento de botões na interface gráfica do *patch* pelo *performer* computacional, ou seja, está relacionada à parte não automatizada do *patch*. Em vista disto, as ações do aplicativo que

se referem ao seu comportamento automatizado foram omitidas nessas Figuras. Tais ações se referem ao disparo das amostras sonoras gravadas e pré-gravadas que ocorrem em momentos imprevisíveis.

Figura 55 – Diagrama da análise das durações da performance instrumental e dos processos eletroacústicos

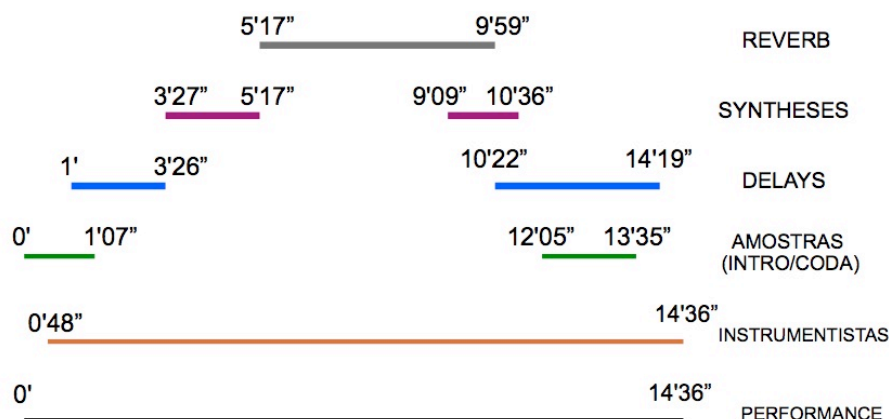
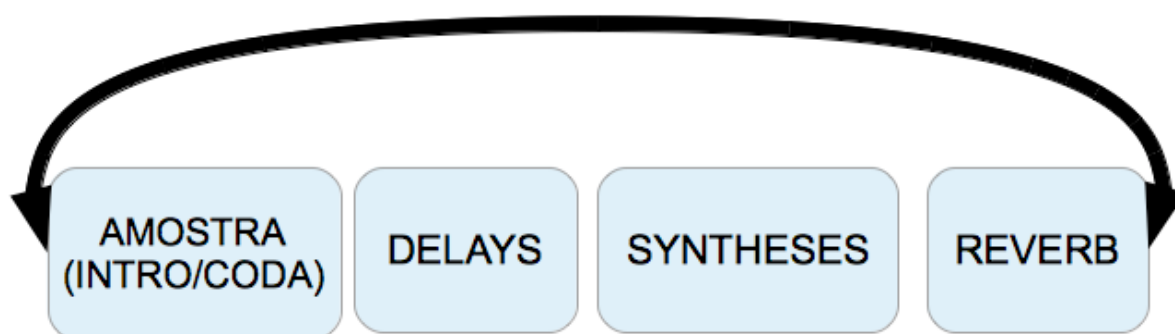


Figura 56 – Estrutura em arco da composição interativa “Atradores Ou-vistos” (parte reativa do *patch*)



4.2 Análise das Interações Dinâmicas entre *Performers* e Sistema Eletrônico na Performance de “Atradores Ou-vistos”

Como comentado no Capítulo 3, a gravação das amostras sonoras da performance inicia-se assim que termina a reprodução da amostra “*amostraintro.wav*”. Quarenta e sete segundos (47”) após o início das gravações, é iniciado o disparo das amostras, tanto gravadas, quanto pré-gravadas, uma a cada vez sem determinação de ordem, pois tais disparos são controlados por algoritmos probabilísticos de cadeias de *Markov*. Na performance, ocorreram alguns disparos de amostras sem conteúdo sonoro (“vazias”), devido ao instrumentista não ter tocado no momento que o som era captado do seu microfone. Isso ocorreu, pois durante a

performance, não se tinha conhecimento de qual dos instrumentistas teria o áudio captado nos momentos de gravação. De acordo com a programação do aplicativo descrita na subseção 3.4.2, quando ocorressem disparos de amostras sonoras, os outros sons eletroacústicos que, por ventura, estivessem ocorrendo, sofreriam uma “adaptação” pela diminuição de sua amplitude, devido a um aumento de camadas de sons eletroacústicos. Se a amostra gravada da performance é “vazia”, isto é, a amostra não é perceptível pela escuta, não há a sobreposição de camadas sonoras, e, portanto, não há a necessidade da diminuição da amplitude nos sons eletroacústicos. Contudo, esta “amostra vazia” não tem seu conteúdo identificado pela programação, ou seja, é simplesmente considerada como uma amostra sonora qualquer, de forma que as amplitudes das respostas sonoras dos *subpatches* dos *canvases* “*DELAYS*”, “*SYNTHESES*” e/ou “*REVERB*” são diminuídas da mesma forma, pois o computador interpreta esse fato com uma sobreposição de camadas. Assim, esse evento (desnecessário), configura-se como um evento aleatório, provocando um “ruído informacional” no sistema que pode ou não estabelecer um novo atrator com os outros elementos como, por exemplo, fazer com que os instrumentistas interajam com essas variações dinâmicas inesperadas.

Em relação aos estímulos enviados pelos *performers* ao aplicativo (*inputs*) no disparo de amostras pré-gravadas, os dados apresentados na Tabela 4 mostram que nenhum nível de amplitude captado da performance foi considerado “alto” pelo aplicativo. O nível de amplitude adotado como “alto” na programação do *subpatch* da Figura 24 no dia da performance da peça “Atratores Ou-vistos” foi de > 65 db (maior que sessenta e cinco decibéis), assim, efetuou-se um ajuste no *patch* para futuras performances, reduzindo este nível para > 60 db (maior que sessenta decibéis).

TABELA 4 – Sequência das amostras pré-gravadas disparadas, as probabilidades de aberturas de seus bancos e os níveis de amplitude captada na performance

SEQUÊNCIA	ABERTURA	BANCO	AMOSTRA	AMPLITUDE
1	33,33%	3	C	BAIXA
2	70%	1	D	MÉDIA
3	70%	2	E	MÉDIA
4	60%	1	D	MÉDIA
5	70%	2	E	MÉDIA
6	60%	1	D	MÉDIA
7	20%	1	D	MÉDIA
8	70%	2	E	MÉDIA
9	60%	1	D	MÉDIA
10	70%	2	E	MÉDIA
11	30%	3	F	MÉDIA
12	50%	3	F	MÉDIA

A escuta dos aspectos intrínsecos (SMALLEY, 1997) das amostras pré-gravadas com o objetivo de transformá-las em um modelo auditivo (NANCE, 2007) para interação (imitação/contraste) - como comentado no Capítulo 2 -, pode ser observada em alguns momentos na performance dos instrumentistas. No Quadro 2 encontra-se uma lista de todas as amostras sonoras disparadas (gravadas na performance e pré-gravadas) e as relações estabelecidas entre os elementos do sistema nas interações dinâmicas (instrumentistas/amostras sonoras). Esse quadro, portanto, possibilita uma análise das medidas sistêmicas (ordem, organização, e complexidade) no SSInPIC.

Quadro 2 – Ordem das amostras sonoras disparadas (gravadas/pré-gravadas) relacionadas com o estabelecimento de uma modelagem auditiva e medidas sistêmicas entre os elementos do SSInPIC

	GRAVADAS	PRÉ- GRAVADA S	MODELO AUDITIVO	ORGANIZAÇÃO	ORDENAÇÃO	COMPLEXIDADE	MINUTAGEM
1	alteração	-	violão (?)	baixo	baixo	médio	2'10" a 2'15"
2	alteração	-	violão (?)	baixo	baixo	médio	2'39" a 2'47"
3	-	C	todos	alto	-	médio	3'05" a 3'19"
4	-	D	-	-	-	alto	3'34" a 3'46"
5	alteração	-	flauta	baixo	baixo	médio	4'01" a 4'06"
6	-	E	-	-	-	alto	4'25" a 4'35"
7	alteração	-	-	-	baixo	alto	4'52" a 5'
8	alteração	-	-	-	baixo	alto	5'22" a 5'27"
9	-	D	violão/ percussão	médio	médio	baixo	5'43" a 5'55"
10	-	E	-	-	médio	médio	6'14" a 6'24"
11	-	D	-	-	alto	baixo	6'43" a 6'55"
12	-	D	percussão	baixo	redundante	baixo	7'14" a 7'26"
13	alteração	-	-	-	mínimo	alto	7'42" a 7'45"
14	-	E	-	-	alto	médio	8'04" a 8'14"
15	vazia	-	-	-	-	alto	8'29" a 8'50"
16	-	D	-	-	redundante	baixo	9'11" a 9'23"
17	alteração	-	percussão	baixo	baixo	baixo	9'39" a 9'52"
18	vazia	-	-	-	-	alto	~10'05" a 10'38" (?)
19	-	E	flauta	baixo	redundante	baixo	10'47" a 10'57"
20	-	F	-	-	-	alto	11'17" a 11'28"
21	-	F	-	-	médio	médio	11'45" a 11'56"

Os dados do Quadro 2 foram obtidos por meio da escuta do registro em pistas de áudio separadas de cada instrumento e do *patch* (stereo), o que foi comentado na introdução deste Capítulo.

Na 1ª Coluna do Quadro 2, encontra-se a ordem em que foram disparadas 21 amostras (linhas) na performance de “Atradores Ou-vistos”, as quais podem ser localizadas temporalmente no vídeo através da minutagem registrada (na mesma linha) na coluna

MINUTAGEM.

Na coluna GRAVADAS, a palavra “alteração” refere-se à uma amostra gravada na performance que foi disparada com alteração em sua mudança na velocidade de reprodução. Nessa mesma coluna, ainda podem ocorrer as palavras “sem alteração” – que se refere à amostra gravada na performance em que não houve alteração na sua reprodução (disparo) -, e a palavra “vazia” - que se refere ao disparo de uma amostra em que na sua gravação o instrumentista não tocou. No caso de um hífen (-) nessa coluna, significa que houve um disparo de uma amostra “pré-gravada” localizada nesta mesma linha do quadro na coluna ao lado.

Na Coluna PRÉ-GRAVADAS encontram-se as letras que se referem as amostras pré-gravadas - conforme a Figura 12 no Capítulo 2 –, que foram disparadas na performance. No caso de um hífen (-) nesta coluna, significa que houve um disparo de uma amostra gravada da performance, localizada nesta mesma linha do Quadro 2 na coluna anterior.

Na coluna MODELO AUDITIVO é indicado o grau de conexão entre os elementos (instrumentistas/amostras sonoras), de forma que, quanto maior a quantidade de instrumentos – violão, flauta, percussão – numa mesma célula dessa coluna, maior é a interação dos elementos do sistema com o modelo auditivo, havendo potencialmente, portanto, maior grau de organização (convergência) – o que pode ser observado no vídeo em determinados pontos temporais de acordo com a coluna MINUTAGEM do Quadro 2 na linha correspondente. Os nomes dos instrumentos referem-se a quais instrumentistas estabeleceram uma interação sonora, baseada numa imitação aproximada da amostra, ou ainda, por contraste, conforme discutido no Capítulo 2. Quando ocorre a palavra “todos” nessa coluna, esta refere-se à interação sonora dos três instrumentistas com a amostra disparada, ou seja, ocorre um maior grau de organização ou convergência na conexão entre os elementos, estabelecendo-se um forte atrator entre eles. O hífen (-) utilizado nas células desta coluna, refere-se ao não estabelecimento de uma interação sonora entre os instrumentistas e as amostras disparadas, que neste caso, representa o aumento da entropia e da expressão da complexidade intrínseca ao sistema, que causava maior indeterminação dos resultados sonoros nestes momentos. O ponto de interrogação entre parênteses (?) refere-se à uma incerteza sobre o estabelecimento do modelo auditivo pelos instrumentistas (registrados nas células desta coluna) com a amostra registrada na linha em questão, o que também pode ser visto como um aumento da entropia. O uso deste ponto de interrogação se deve ao fato da modelagem auditiva como medida de organização entre os elementos do sistema ter sido considerada pelo autor desta pesquisa como uma medida sistêmica classificada numa categoria de “propriedade subjetiva”, pois

“[...] pode depender do estado do sujeito que observa o sistema” (PESSOA JÚNIOR, 2001, p. 29).

Na coluna ORGANIZAÇÃO, foi medido o grau de organização (ou de convergência) do sistema baseado no número de instrumentos musicais relacionados em cada linha da coluna MODELO AUDITIVO, isto é, um instrumento corresponde ao grau baixo, dois ao grau médio e três ao grau alto de organização. No caso de um hífen, isso significa que não houve, neste ponto, organização do sistema em que os instrumentistas tivessem contribuído.

Na coluna ORDENAÇÃO, foi medido o grau de ordenação de acordo com a repetição de informações no sistema no que se refere aos disparos de amostras pré-gravadas e gravadas e reproduzidas com ou sem alteração. Ou seja, como o sistema se ordenou em relação às amostras sonoras disparadas na performance “Atratores Ou-vistos”. Uma amostra gravada da performance em um momento e disparada em outro aumentava em algum grau a ordenação do sistema. Como essas amostras eram disparadas uma única vez não ocorreu um alto grau de ordenação. Além disso, todas as reproduções dessas amostras (gravadas da performance) foram alteradas (apesar de haver a possibilidade de não sofrerem alterações), o que diminuía sua semelhança com o evento sonoro gravado anteriormente na performance. Dessa forma, essas amostras que sofreram alterações em sua reprodução davam um baixo grau de ordenação do sistema. Quando ocorria o disparo de uma amostra “vazia”, aparece um hífen na coluna ORDENAÇÃO, significando que não houve ordenação do sistema neste ponto. Já as amostras pré-gravadas não sofriam alterações em sua reprodução, contudo, seu material era proveniente de fontes sonoras que não estavam presentes durante a performance. Assim, cada primeira reprodução de uma amostra pré-gravada existia um elemento surpresa, o que diminuía a ordem no sistema, sinalizado pelo hífen na coluna ORDENAÇÃO. Uma, duas, três repetições das amostras pré-gravadas davam, respectivamente, graus baixos, médios e altos de ordenação do sistema. Quatro ou mais repetições foram classificadas nessa coluna como “redundante”.

Como acontece durante toda a performance, o atrator estabelecido é logo desfeito. Em capítulos anteriores, foi comentado que essa “*quase-formação* de atratores” (FURLANETE, 2010, grifo do autor) é característica de um sistema com auto-organização. Entende-se, portanto, que a “*quase-formação*” de um atrator estaria relacionada à eliminação de um excesso de redundância na informação decorrente da formação, de fato, de um atrator. Desta forma, o excesso de redundância (medida de ordem) poderia causar uma inadaptação do sistema a possíveis ruídos informacionais. Apesar de uma amostra gravada do passado ser reproduzida no tempo presente, muitas vezes ocorreram alterações em suas reproduções, que

poderiam afastar em algum grau sua analogia com o passado, diminuindo seu grau de redundância e aumentando seu grau de entropia. Contudo, apesar de ser uma performance de improvisação (ou seja, é impossível saber de antemão sobre algum aspecto do conteúdo de uma amostra sonora que poderá ser gravada dos instrumentistas), as alterações de reprodução dessas amostras já estão previstas na programação do aplicativo. Desta forma, percebe-se que não há aumento da complexidade, que viria de elementos externos ao sistema, mas apenas a expressão de sua própria complexidade interna, que pode aumentar ou diminuir durante a performance sem exceder as limitações do sistema que foram definidas em sua idealização, o que caracteriza, portanto, um sistema com auto-organização secundária, conforme Debrun (1996).

Na coluna COMPLEXIDADE, mediu-se o grau da complexidade baseado nos elementos de imprevisibilidade, ou seja, buscou-se medir o grau de entropia que ocorria durante a performance do sistema. Nos disparos de amostras gravadas durante a performance, reproduzidas com alteração, a cada vez que essas amostras eram imitadas pelos instrumentos musicais, o grau de complexidade era diminuído no sistema, sendo classificado como um grau “médio” de complexidade. No disparo de número 1 (primeira coluna), por exemplo, há a possibilidade (indicada pelo ponto de interrogação) do violão ter imitado uma amostra gravada ao vivo logo no início da performance, o que foi considerado como um grau médio de complexidade. Tratando-se de um disparo (número 1) de uma amostra sonora que ocorrerá apenas uma vez (amostras gravadas ao vivo) o grau de ordenação foi considerado baixo, e pelo fato dos outros instrumentistas – além do violão - não conseguirem um ponto de convergência, a organização também foi considerada baixa. No disparo número 12, pode-se notar que quanto maior a ordenação (repetição das amostras) menor será a complexidade. O disparo de número 17, no entanto, constitui-se como uma exceção, já que possuía um alto grau de semelhança tímbrica com sua imitação (sons de percussão x sons de percussão). Dessa forma, o grau de complexidade foi considerado “baixo” nesse caso. Houve imprevisibilidade no seu conteúdo, tendo em vista que nunca se sabe de qual instrumentista e em que momento uma amostra é gravada. Contudo, o seu conteúdo – gestos instrumentais executados no vibrafone e canos - já havia sido executado ao vivo pelo percussionista a partir de 7’46” (ponto em que termina a reprodução do décimo terceiro disparo e inicia a gravação de outra amostra⁴). Aos 9’39” (momento do disparo da amostra), o percussionista utiliza o

⁴ Apesar de não se ter o conhecimento de qual instrumentista é gravada uma amostra sonora durante a performance, há uma certa previsibilidade em relação aos momentos em que estas gravações ocorrem. Estes momentos ocorrem após o disparo da amostra “amostraintro.wav”, em que são gravadas 5 amostras durante 47”,

vibrafone, o que provavelmente demonstra que foi influenciado pelo disparo da amostra gravada de sua execução em 9'17", caracterizando assim, uma interação dinâmica.

No caso de não ocorrer uma imitação dessas amostras, o grau de complexidade no sistema era considerado “alto”, da mesma maneira com as amostras pré-gravadas não imitadas. O grau de complexidade era diminuído quanto mais repetições de uma amostra pré-gravada ocorriam. Além disso, as imitações instrumentais contribuíam ainda mais para diminuir o grau de complexidade do sistema.

É importante salientar que a análise descrita a respeito do comportamento auto-organizado do sistema nos tópicos deste capítulo são baseadas em recortes – interações dinâmicas e interações estáticas – que não levam em conta a simultaneidade temporal desses recortes na performance. Alguns dos resultados apresentados no Quadro 2 são baseados em julgamentos subjetivos da percepção, conforme foi comentado. Contudo, nesta pesquisa, foi possível demonstrar os graus de entropia e redundância daquelas amostras sonoras que podem ser disparadas e reproduzidas da mesma forma mais de uma vez, ou seja, as amostras pré-gravadas.

4.2.1 Entropia e Redundância nos Disparo das Amostras Pré-Gravadas

Com a relação das amostras sonoras que foram disparadas na performance, pode-se agora passar a uma análise do funcionamento do algoritmo que controla tais disparos, baseada em seus graus de entropia e da redundância. Na Tabela 4 apresentada na página 133, observa-se a relação das amostras pré-gravadas na sequência em que foram disparadas na performance “Atratores Ou-vistos”, além da probabilidade de abertura de seu banco e o nível de amplitude captado da execução do instrumentista, o qual influenciava a escolha da amostra pelo *patch*. Estes dados foram possíveis de se obter a partir do mapeamento feito no aplicativo (ocorrências das amostras/níveis de amplitude captados da performance instrumental) e do registro em áudio e vídeo da performance onde pôde ser observada a sequência em que estas amostras foram disparadas. Na Tabela 4, foi considerada apenas a probabilidade de “abertura do banco” da amostra pré-gravada a cada disparo, sem levar em conta a probabilidade do disparo de uma amostra gravada na performance.

Observa-se na Tabela 4 que as *cadeias de Markov* se comportaram de uma forma menos previsível em apenas dois disparos das amostras pré-gravadas - a 7ª e a 11ª na ordem

e após o disparo de uma amostra gravada da performance. Contudo, os instrumentistas não foram informados sobre os momentos da gravação das amostras.

dos disparos -, pois, seus bancos tinham uma probabilidade menor de serem abertos do que os outros. A 6ª amostra (“D”) e a 10ª amostra (“E”) selecionaram, respectivamente, o banco 1 que tinha apenas 20% de probabilidade de ser aberto para o disparo da 7ª amostra, e o banco 3 que tinha apenas 30% de probabilidade para o disparo da 11ª amostra.

Utilizando-se dos dados de probabilidades da Tabela 4 é possível obter os graus de Redundância (R), Entropia Máxima ($S_{\max}$) e Entropia (S) por meio das Equações 1, 2 e 3 - descritas na subseção 1.2.3 do Capítulo 1 – de cada amostra pré-gravada disparada na performance “Atratores Ou-vistos”. Conforme visto nessas Equações, é necessário calcular antes os valores da Entropia (S) e Entropia Máxima ($S_{\max}$) para se obter o valor da Redundância (R). Segundo Pessoa Júnior (2001, p. 29), a redundância pode medir a ordem no sistema, neste caso, como uma “propriedade dinâmica” por “[...] depender do conjunto de estados do sistema ao longo de um intervalo de tempo [...]”. Na Tabela 5, podem ser vistos os graus de Entropia (S), Entropia Máxima ($S_{\max}$) e Redundância (R) de cada uma dessas amostras.

TABELA 5 – Graus de Entropia, Entropia Máxima e Redundância de cada amostra pré-gravada disparada na performance da composição “Atratores Ou-vistos”

AMOSTRA	S	$S_{\max}$	R
C	0,36	0,36	0
D	0,43	2,15	0,80
E	0,36	1,44	0,75
F	0,35	0,71	0,50

De acordo com os resultados obtidos na Tabela 5: $R_c = 0$; $R_d = 0,8$; $R_e = 0,75$; e $R_f = 0,5$, percebe-se que há um maior grau de redundância na amostra “D” que possui um número N (Equação 3) igual a 5 que é maior do que o das outras amostras, significando que foi disparada 5 vezes. Na amostra “C”, em que há um grau de redundância nulo, o número N é igual a 1, ou seja, foi disparada apenas uma única vez. Dessa forma, nota-se que quanto mais ocorrências da mesma amostra sonora na performance, maior será o seu grau de redundância.

4.3 Análise das Interações Estáticas entre *Performers* e Sistema Eletrônico na Performance de “Atratores Ou-vistos”

Conforme comentado no Capítulo 2, foram selecionados gestos instrumentais da flauta-doce, do violão e da percussão que estimulariam respostas do aplicativo em performances

com o SSInPIC, gerando interações estáticas, isto é, aquelas interações do aplicativo por meio de processos reativos. Apesar desta seleção, a performance dos músicos tinha um alto grau de liberdade, garantindo sua característica de improvisação contemporânea, a busca por uma linguagem não-idiomática e uma escuta com foco nas interações tímbricas e nos aspectos intrínsecos dos sons. A seguir, são destacados alguns pontos relacionados às interações estáticas na performance de “Atratores Ou-vistos”, apresentada com o SSInPIC.

4.3.1 *Delays*

- *Violonista*

Logo no início da performance, após o disparo da amostra “*amostraintro.wav*”, o performer computacional enviou o comando pelo monitor para que o violonista interagisse com o processamento baseado em linhas de *delays*. O violonista utilizou gestos instrumentais baseados em técnicas estendidas como raspagens de unha nas cordas, trêmolos com glissandos, slaps, etc, que foram destacados e prolongados pelos *delays*. Desta forma, o som processado realimentou o processo de interação, pois influenciou o instrumentista na execução e escolha dos gestos instrumentais seguintes, estabelecendo um atrator entre violonista e sons eletroacústicos.

- *Flautista e Percussionista*

O mesmo ocorreu no comportamento da interação da flautista e do percussionista com o aplicativo. Destaca-se o uso do roem-roem pelo percussionista que, aos 10’56”, atraiu a atenção do performer computacional (a face-sujeito) para escolhê-lo para interação com os *delays*. Além disso, a flautista se adaptou bem a este processamento, utilizando variados gestos instrumentais e contrastes dinâmicos. Havia uma “adaptação” quase constante ao processamento com *delays*, levando o performer computacional a evitar o excesso de redundância nos atratores que eram formados, o que se realizava através da troca de instrumentista para interação e/ou na troca de processos no aplicativo.

4.3.2 *Reverb*

- *Violonista*

Aos 5’17”, o *performer* computacional escolheu o violonista para interagir através da amplitude com o módulo de síntese do canvas “*REVERB*”. Contudo, o instrumentista

conseguiu se adaptar apenas aos 5'56'', momento em que encontrou um gesto mais adequado para o controle da síntese, estabelecendo aí um atrator. Aos 6'10'', o performer computacional decidiu trocar o parâmetro - que é analisado na performance - para o controle da síntese do *canvas* “*REVERB*”, que passou da amplitude para as frequências e ataques, o que permaneceu até 7' e influenciou uma troca de gestos instrumentais do violonista. Estas mudanças realizadas pelo performer computacional que influenciaram as mudanças de decisões do violonista, ocorreram com o objetivo de desfazer o atrator estabelecido logo antes, evitando assim o excesso de redundância. Nesta troca de gestos instrumentais do violonista, ele apresentou trechos melódicos executados em *legato*, influenciando o aplicativo a criar uma textura de *reverb* que muitas vezes sobrepunha alguns dos cinco primeiros harmônicos das notas que tocava.

- *Percussionista*

Depois desse momento, dos 7'01'' até 7'34'', o percussionista controlou os sons desta mesma síntese (*canvas* “*REVERB*”) utilizando do sino, toques com a baqueta na lateral do prato, e principalmente o roem-roem, finalizando este trecho de sua interação com um toque no bumbo-sinfônico.

- *Flautista*

Aos 8'17'', o uso da flauta sopranino, que até então atuava na construção do ambiente sonoro, atraiu a atenção do performer computacional que a escolheu para controlar este mesmo processo de síntese até 8'52''. Percebe-se que nesse trecho (dos 6'10'' aos 8'52'') as respostas do aplicativo aos estímulos enviados pelos instrumentistas tinham um atraso de dois segundos, conforme a programação descrita no Capítulo 3.

4.3.3 Reverb e Synthesises

Aos 8'53'', após romper o atrator entre a flautista e o *REVERB*, o *performer* computacional enviou o comando para o percussionista controlar novamente o “*REVERB*” com parâmetros da análise da amplitude, contudo, poucos gestos instrumentais na percussão contribuíram por um curto momento para estimular o aplicativo (aos 9' e 9'53'', no entanto, percebe-se que houve uma resposta do aplicativo ao vibrafone). Este fator gerou então um ruído informacional que estimulou o *performer* computacional a avançar na estrutura definida *a priori* descrita na seção 4.1 e aplicar dois processos simultâneos aos 9'10'' (*canvas*

“*SYNTHESES*” e “*REVERB*” - amplitude), com dois instrumentistas no controle dos sons eletroacústicos, na busca de novos atratores. Aos 9’17”, o percussionista interagiu com os sons dos módulos de síntese (*canvas* “*SYNTHESES*”) com gestos no vibrafone, buscando utilizar estes sons eletroacústicos como um “modelo auditivo” para imitação, estabelecendo aí um atrator, devido à mudança de processo escolhida pelo *performer* computacional.

Dos 9’09” aos 10” verifica-se a ocorrência de um trecho de grande densidade sonora na performance, com dois processos com interação estática – *canvas* “*SYNTHESES*” e “*REVERB*” – ocorrendo simultaneamente ao eventual disparos de amostras (16ª e 17ª no Quadro 2). Como comentado acima, o percussionista tocou o vibrafone no intuito de buscar a imitação dos módulos de síntese sonora do *canvas* “*SYNTHESES*” aos 9’17”, o que continuou até 9’50”. Neste trecho, é notável um cruzamento de interações na performance do percussionista a partir de 9’39”. Este cruzamento se dá na imitação dos sons da síntese e dos sons da 17ª amostra disparada (Quadro 2), que estabelece um forte atrator.

No vídeo, notam-se interessantes cenas de “re-adaptação” das decisões da flautista pela influência dos sons eletroacústicos, baseadas nas trocas que ela fez entre diversos tipos de flautas-doce: sopranino, soprano, contralto, tenor e baixo, tanto relacionadas com suas próprias interações com o aplicativo, quanto suas interações com o ambiente sonoro em geral. Aos 6’28”, ocorreu uma troca para a flauta sopranino provavelmente com o objetivo de interagir com o ambiente sonoro produzido pelo aplicativo “*REVERB*” – que respondia com frequências mais agudas ao controle que era exercido pelo violonista e depois pelo percussionista. Aos 9’03”, a flautista deixou de controlar os sons do módulo “*REVERB*” (ataques/ frequências) por ter visto no monitor outro número que não o seu, e decidiu trocar a flauta sopranino pela flauta baixo. Contudo, como comentado acima, aos 9’10” o *performer* computacional decidiu escolher os módulos de síntese “*SYNTHESES*” para interação com os instrumentistas, o que mudou a decisão da flautista, pois ela desistiu de usar a flauta-baixo e decidiu usar a tenor, provavelmente por considerar seu timbre mais adequado para imitar os processos eletroacústicos iniciados aos 9’10”, contribuindo na manutenção do ambiente sonoro. Esta decisão da flautista em relação à escolha do timbre do instrumento se motivou pela busca de um novo atrator.

Assim, nota-se que no processo de interação estática entre os agentes, o estabelecimento de atratores que garantiam a emergência de coerências locais com resultados musicais coerentes. O grau de redundância dessas coerências locais influenciava as tomadas de decisões do *performer* computacional, que, conseqüentemente, refletiam no grau de ordenação do sistema. Em alguns momentos da performance, observa-se o não estabelecimento de

atratores diante das decisões tomadas pelos agentes, o que elevava o grau de complexidade no sistema. A hegemonia da “face-sujeito” do sistema que regula a “*quase-formação*” dos atratores, torna-se um elemento importante na emergência de comportamentos variados, sem no entanto, exercer uma força dominante, mas sim, executar comandos que podem ter resultados imprevisíveis. Essa imprevisibilidade ocorre, pois, tais comandos, influenciam, em tempo-real, as próprias tomadas de decisão da face-sujeito.

CONCLUSÃO

Esta pesquisa partiu da busca e reflexão de conceitos para a elaboração de um sistema sonoro-interativo com cinco agentes, denominado de SSInPIC. O conceito de auto-organização (DEBRUN, 1996), foi o principal ponto de partida para a concepção poética do sistema. Este conceito se caracteriza pela produção de “comportamento emergente” no sistema, que desta forma se relaciona com as estéticas musicais composicionais da música erudita no Século XX que se utilizaram de diferentes formas de “abertura” em música, descritas no Capítulo 1. A auto-organização de um sistema produz comportamento emergente a partir das próprias interações que ocorrem entre seus elementos. Desta forma, o conceito de interação torna-se um ponto de intersecção entre sistemas “interativos” e sistemas auto-organizados. No primeiro capítulo, foram discutidos tais conceitos que se interrelacionam e promoveram a fundamentação teórica para o objeto de estudo desta pesquisa.

No Capítulo 2, foi feita uma correlação dos temas discutidos no Capítulo 1 com o SSInPIC, de maneira a apresentar seus elementos e suas potencialidades. Através de uma descrição da idealização do sistema, foram identificadas as potencialidades das interações dinâmicas e estáticas nas relações entre seus elementos, além das sonoridades que seriam obtidas com estas relações. Enumerou-se uma série de perguntas que os interagentes humanos deveriam fazer a si mesmos durante uma performance com o SSInPIC para promoverem o comportamento emergente.

A mudança do mapeamento da distribuição de probabilidades como uma potencialidade do sistema no disparo e na reprodução de amostras sonoras pré-gravadas e gravadas da performance, respectivamente, foi prevista como uma solução para o sistema adquirir comportamento emergente nas interações dinâmicas. O uso do conceito de “modelo auditivo” de Nance (2007) também foi utilizado como uma possibilidade de se formarem “atratores” entre sons acústicos e eletroacústicos, ou entre comportamento musical humano e eletrônico. A “*quase-formação* de atratores” obtida com o equilíbrio entre a adaptação ao ruído informacional e o impedimento de um excesso de redundância foram realizados através de determinados procedimentos na programação do aplicativo (Capítulo 3) que reproduzem e disparam amostras gravadas da performance e pré-gravadas, respectivamente, com a utilização de algoritmos de *cadeias de Markov*. As tabelas de probabilidades foram importantes ferramentas que auxiliaram na análise das pontencialidades do sistema no Capítulo 2. Posteriormente, no Capítulo 4, os valores encontrados nas tabelas de probabilidades do Capítulo 2, foram utilizados nos cálculos da entropia e redundância das

amostras sonoras pré-gravadas disparadas - na performance “Atratores Ou-vistos” -, como medidas de complexidade e ordem do sistema, respectivamente. Estes cálculos foram obtidos através de equações da Teoria Matemática da Informação de Shannon (1948), utilizadas pelo cibernético e psiquiatra Ashby (1999) no estudo do comportamento de sistemas auto-organizados.

Segundo Primo (2005, p. 6, grifo do autor), a “[...] reação do sistema digital diante da escolha entre certas alternativas é sim uma forma de interação, mas de tipo limitado. Trata-se na verdade de um processo que gira em torno de *potenciais*”. Desta forma, o SSInPIC se relaciona com o conceito de “auto-organização” secundária, pois, não se pode dizer que haja uma evolução da “complexidade” no comportamento do sistema computacional. O que ocorre é uma variação nos graus da complexidade idealizada - com base em processos potenciais - de antemão no sistema. Apesar disto, as interações que ocorrem no SSInPIC não são previsíveis, pois, além de procedimentos probabilísticos nas interações dinâmicas do aplicativo, foram utilizados procedimentos de abertura em música na performance humana, como o “modelo auditivo”, a “improvisação contemporânea”, e uma supervisão hegemônica da face-sujeito do sistema através do performer computacional. Todos esses aspectos contribuíam na expressão da complexidade do sistema. A supervisão do performer computacional não era dominante, pois ele também se subordinava à busca de um comportamento emergente no sistema, que, muitas vezes, era direcionado pelos outros elementos - os instrumentistas e a parte automatizada do aplicativo (disparos de amostras).

Em relação à programação descrita no Capítulo 3 e aos resultados obtidos no Capítulo 4 com análise da performance “Atratores Ou-vistos”, conclui-se a necessidade de se realizarem mais performances com o SSInPIC com o objetivo de maior exploração de suas potencialidades, como também de ajustes de parâmetros na programação. No vídeo, observa-se que o controle da amplitude dos módulos de síntese através do “*ducking*” não foi muito perceptível, em que, por meio de um “modelo auditivo”, a síntese se prestou mais a uma interação dinâmica do que a uma interação estática com os performers. Este fator demanda novos ajustes e testes na programação para que os *performers* tenham melhor percepção desta interação estática.

Como foi visto na Tabela 4 do Capítulo 4, o aplicativo praticamente captou apenas um nível de dinâmica (nível médio). No vídeo, percebemos sons instrumentais com variabilidade na dinâmica, o que demonstra que este problema está na necessidade de ajustes na programação do aplicativo e não na performance instrumental. Tais ajustes devem se basear em novos testes, com novas oficinas e performances com o aplicativo, pois, como foi

comentado no Capítulo 4, a performance registrada no DVD anexado foi uma das primeiras performances realizadas com o sistema até o momento.

Outros ajustes devem ser feitos na programação responsável pela gravação das amostras durante a performance, pois ocorreram algumas reproduções que não foram perceptíveis. Estas reproduções de amostras sonoras “vazias” apenas causavam uma repentina diminuição da amplitude nos outros processos eletroacústicos, e consequentemente um “ruído informacional”, que não foi adaptado pelos instrumentistas, de acordo com a análise da performance no Capítulo 4. Isto pode ter ocorrido devido ao não conhecimento de qual dos instrumentistas teria sua amostra sonora gravada, o que aumentava o nível da complexidade no sistema.

Nota-se a necessidade de mais oficinas com o SSInPIC, principalmente no que se refere aos fatores comentados acima, e ainda ao controle da síntese sonora do *subpatch* do *canvas* “*REVERB*” através da amplitude da performance instrumental. Observa-se no vídeo que não houve um nível de “adaptação” desejável dos instrumentistas no controle desta síntese pelo parâmetro da amplitude. Contudo, pelos parâmetros dos ataques e frequências, nota-se a criação de situações musicalmente interessantes pelos instrumentistas, aplicativo, e performer computacional, devido às interações sonoras que ocorriam. O processamento com linhas de *delays* aplicadas em tempo-real aos sons dos instrumentos musicais também teve um alto nível de adaptação pelos instrumentistas, cujo excesso de redundância era equilibrado pela variabilidade das técnicas estendidas e contrastes dinâmicos na performance instrumental, ou ainda pela troca de instrumentista efetuada pelo performer computacional.

Embora haja a necessidade (e espaço) para aprimoramentos na programação do aplicativo do SSInPIC, julga-se que os objetivos da pesquisa de criar um sistema sonoro-interativo com auto-organização (secundária) foi atingido satisfatoriamente.

Futuramente, outros projetos de sistema sonoro-interativo utilizando este sistema como um protótipo podem ser realizados. Por exemplo, uma versão do aplicativo independente de um performer humano operando o sistema computacional ao vivo, realizando uma programação que efetue, além dos processamentos e síntese sonora, a troca de instrumentista que interage com o aplicativo, baseados na análise de parâmetros da performance instrumental, interpretada por algoritmos com redes neurais, os quais podem aprender, reconhecer e classificar padrões musicais.

O conceito de auto-organização demonstrou ser uma importante ferramenta em processos criativos que interagem música e ciência do ponto de vista do criador musical. A utilização de formalismos através dos algoritmos computacionais possibilita a manipulação de

estruturas musicais pelo humano em tempo-real que pode viabilizar comportamentos emergentes, aumentando o potencial criativo dos agentes no sistema. O processo criativo descrito neste trabalho buscou explorar o potencial expressivo da música vista como representação de um processo com possibilidades sonoras que só podem ser proporcionadas através da interação entre homem e máquina.

REFERÊNCIAS

ALMEIDA, A. Z. *Forma Lírica e Campos Temporais*: fundamentos das multiplicidades de performance em Klavierstück XI de Karlheinz Stockhausen. Campinas, 2010. 164 f. Tese (Doutorado em Música) – Instituto de Artes, Universidade Estadual de Campinas, Campinas, 2010.

ASHBY, W. R. *Design for a brain*. 2. ed. Frome and London: Butler and Tanner Ltd., 1954. 260 p.

_____. *An Introduction to cybernetics*. Second Impression. Chapman & Hall, London 1956. Internet: 1999. Disponível em: <<http://pcp.vub.ac.be/books/IntroCyb.pdf>>. Acesso em: 3 dez. 2011.

_____. Principles of the self-organizing systems. *Principles of self-organization*: Transactions of the University of Illinois Symposium. VON FOERSTER, H.; ZOPF, G. W. (Org.). Pergamon Press, London UK, p. 255-278, 1962.

BANZHAF, Wolfgang. Self-organizing systems. *Encyclopedia of Complexity and Systems Science*, Heidelberg, p. 8040-8050, 2009.

BARRAUD, Henry. *Para compreender as músicas de hoje*. Tradução de J. J. de Moraes e Mária Lúcia Machado. 3. ed. São Paulo: Perspectiva, 2005. 168 p. Tradução de: Pour comprendre les musique d'aujourd'hui.

BARREIRO, D. L. Coherence and spontaneity in Interferências (2003), for cello and computer. In: *Proceedings of 11th Brazilian Symposium on Computer Music*. São Paulo: IME-ECA/USP, p. 35-46, 2007.

_____. Sonic image and acousmatic listening. *Organized Sound*, Cambridge University Press, v. 15, n. 1, p. 35-42, 2010.

BARREIRO, D. L.; FORNARI, J. Glossário. In: _____. *Criação musical e tecnologias: teoria e prática interdisciplinar*. Goiânia: ANPPOM, 2010. vol. 2. Disponível em: <<http://www.anppom.com.br/editora/>>. Acesso em: 22 de mar. 2011.

BARREIRO, D. L.; TRALDI, C. A.; CINTRA, C. L. de A.; MENEZES JÚNIOR, C. R. F. de. Instrumentos acústicos e meios eletrônicos em tempo real: estratégias de improvisação coletiva. *Revista Eimas*, Juiz de Fora, v. 1, n. 1, p. 2-16, set. 2010. Disponível em: <<http://www.eimas.net/revi.html>>. Acesso em: 18 abr. 2011.

CAMPESATO, Lílian. Dialética do ruído. *Anais do Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música*, Florianópolis, p. 1389-1393, 2010. 20º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, Florianópolis, 2010.

CAMPOS, C. S.; TRALDI, C. A.; MANZOLLI, J. Improvisação e interpretação mediada. In: SIMPEMUS3-Simpósio de Pesquisa em Música, 2006, Curitiba. *Anais do Simpemus3*. Curitiba : Editora Deartes, p. 206-211, 2006.

CHAGAS, P. C. A Música de Câmara Telemática: A Metáfora de Flusser e o Universo da

Música Eletroacústica. *Revista de Comunicação, Cultura e Teoria da Mídia*, São Paulo, v. 1, n.11, mar. 2008. Disponível em: < <http://www.revista.cisc.org.br/ghrebh/index.php/gh/article/viewfile/7/5>>. Acesso em: 16 dez. 2009.

CHION, Michel. *Guide to sound objects*. Pierre Schaeffer and musical research. Versão digital. 2009. Disponível em: <<http://modisti.com/news/?p=14239>>. Acesso em: 20 dez. 2009.

CINTRA, C. L. de A. *Formas de utilização do acaso na composição musical contemporânea*. 2004. 148 f. Dissertação (Mestrado em Música) Universidade Estadual Paulista, São Paulo, 15 abr. 2004.

_____. Imponderável: as concepções de acaso e suas influências na composição musical contemporânea. *Ouvirouver*, Uberlândia, v. 6, n. 2, p. 302-319, jul./dez. 2010.

COSTA, R. L. M. O ambiente da livre improvisação: referências para um campo de consistência. *Anais do Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música*, Porto Alegre, p.1-7, 2003. 14º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, Porto Alegre, 2003.

_____. A ideia de corpo e a configuração do ambiente na improvisação musical. *Opus: Revista da Associação Nacional de Pesquisa e Pós-Graduação em Música – ANPPOM*. Goiânia, v. 14, n. 2, p. 87-99, dez. 2008.

_____. A idéia de jogo em obras de John Cage e no ambiente da livre improvisação. *Per Musi* (UFMG), Belo Horizonte, v. 19, p.83-90, 2009.

CYCLING'74. *MSP Tutorials: table of contents*. 2012. Disponível em: <<http://cycling74.com/docs/max5/tutorials/msp-tut/mspindex.html>>. Acesso em: 20 jun. 2012.

DEBRUN, Michel. A ideia de auto-organização. *Auto-organização: estudos interdisciplinares em filosofia, ciências naturais e humanas, e artes*. DEBRUN, M.; GONZALES, M. E. Q.; PESSOA JÚNIOR, O. (Org.). Coleção CLE, Campinas, n. 18, p. 3-23, 1996.

DELEUZE, Gilles. *Cinema 1: a imagem movimento*. Tradução de Stella Senra. [S.l.]: Editora Brasiliense, 1983. 268p.

_____. *Diferença e repetição*. Tradução de Luiz Orlandi e Roberto Machado. Rio de Janeiro: Graal, 1988. 126 p.

_____. *Lógica do Sentido*. Tradução de Luiz Roberto Salinas. 4. ed. São Paulo: Editora Perspectiva S.A., 2003. Tradução de Logique du sens.

DIAS, H. P. G. *A “Querela dos Tempos”*: um estudo sobre as divergências estéticas na música eletroacústica mista. 2006. 132 f. Dissertação (Mestrado em Música) Universidade Estadual Paulista, São Paulo, ago 2006.

DOBRIAN, Christopher. Strategies for continuous pitch and amplitude tracking in realtime interactive improvisation software. *Proceedings of the Sound and Music Computing*

Conference, Paris, 2004. Sound and Music Computing Conference 2004. Disponível em: <http://www.music.arts.uci.edu/dobrian/PAPER_051.pdf> Acesso em: 26 maio 2010.

DRUMMOND, Jon. Understanding interactive systems. *Organized Sound*, Cambridge University Press, v. 14, n. 2, p. 124-133, 2009.

ECO, U. *Obra Aberta*. 8.ed. São Paulo: Ed.Perspectiva, 1991. 284 p.

EMMERSON, Simon. *Living electronic music*. England: Ashgate Publishing Limited, 2007. 195 p.

FERRAZ, Silvio. *Música e repetição: a diferença na composição contemporânea*. São Paulo: Educ, 1998. 273 p.

_____. Livro das sonoridades (notas dispersas sobre composição): um livro de música para não-músicos ou de não-música para músicos. Rio de Janeiro: 7Letras, 2005. 127 p.

FIGUEIRÓ, C. S.; KROGER JÚNIOR, P. R. Elementos e ideias para uma metodologia de estruturação de eventos em Pure Data. *Anais do Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música*, São Paulo, v. 17, p. 1-11, 2007. 17º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, São Paulo, 2007.

FIGUEIRÓ, C. S. *Estratégias composicionais na interação homem-máquina*. RUA Revista Universitária do Audiovisual, v. 2, p. 605, 2008. Disponível em: <<http://www.ufscar.br/rua/site/?p=605>>. Acesso em: 13 dez. 2009.

FREE SOFTWARE FOUNDATION, Inc. *Floss manuals: free manuals for free software* (Pure Data). 2. ed. Boston: General Public License, 1991. Disponível em: <<http://flossmanuals.net/puredata/>>. Acesso em: 29 mar. 2010.

FUBINI, Enrico. *Estética da música*. Tradução de Sandra Escobar. 2. ed. Lisboa: Edições 70, 2008. 183 p. Tradução de: *Estética della musica*.

FURLANETE, F. P.; MANZOLLI, J. Síntese Sonora com Agentes Distribuídos. In: SEMINÁRIO MÚSICA, CIÊNCIA E TECNOLOGIA, 2., 2005, Campinas. *II Seminário Música, Ciência e Tecnologia*. Campinas: 2005, p. 1-5.

FURLANETE, F. P. *Modelagem de interações musicais com dispositivos informáticos*. Campinas, 2010. 143 f. Tese (Doutorado em Música) – Instituto de Artes, Universidade Estadual de Campinas, Campinas, 2010.

GARNETT, G. E. The aesthetics of interactive computer music. *Computer Music Journal*, v. 25, n. 1, p. 21-33, 2001.

GRIFFITHS, Paul. *Aleatory*. In: THE NEW GROOVE dictionary of music and musicians. Washington: Macmillan, v. 01. p. 237-242, 1980.

IAZZETTA, F. H. de O. *Formalization of computer music interaction through a semiotic approach*. 1996. Disponível em: <<http://www.eca.usp.br/prof/iazzetta/papers/inter.htm>>. Acesso em: 10 abr. 2011.

KOSTKA, S. M. *Materials and techniques of twentieth-century music*. Englewoods Cliffs, New Jersey: Prentice Hall, 1989. 337 p.

KREIDLER, Johannes. *Programming Electronic Music in Pd*. 27 jan. 2009. Disponível em: <<http://www.pd-tutorial.com/english/index.html>>. Acesso em: 08 mar. 2010.

LEWIS, G. E. Too many notes: complexity and culture in Voyager. *Leonardo Music Journal*, v. 10, p. 33-39, 2000.

LIPPE, Cort. A composition for clarinet and real-time signal processing: using Max on the IRCAM Signal Processing Workstation. *Proceedings of Italian Colloquium on Computer Music*, Milão, p. 428-432, 1993. 10th Italian Colloquium on Computer Music, Milão, 1993. Disponível em: <<http://www.music.buffalo.edu/faculty/lippe/pdfs/milan.pdf>>. Acesso em: 5 jan. 2012.

_____. *Real-time interaction among composers, performers, and computer systems*. In: Information Processing Society of Japan SIG Notes. V. 2002. p. 1-6. Disponível em: <<http://www.music.buffalo.edu/faculty/lippe/pdfs/Japan-2002.pdf>>. Acesso em: maio de 2010.

LOCHHEAD, J.; AUNER, J. *Postmodern music/postmodern thought*. Studies in Contemporary Music and Culture. New York: Routledge, 2002. Disponível em: <http://books.google.com/books?id=hqiSnCkVU4cC&printsec=frontcover&dq=postmodern+music/postmodern+thought&hl=en&src=bmrr&ei=bqi1TZa9OKfz0gGus5T9CA&sa=X&oi=book_result&ct=book-preview-link&resnum=1&ved=0CDIQuwUwAA#v=onepage&q&f=false>. Acesso em: 25 abr. 2011.

MANZOLLI, Jônatas. Auto-organização: um paradigma composicional. *Anais do Encontro Anual da Anppom*, João Pessoa, 1995. 8º Encontro Anual da Associação Nacional de Pesquisa e Pós-Graduação em Música, João Pessoa, 1995. Disponível em: <http://www.anppom.com.br/anais/anaiscongresso_anppom_1995/comppairelat2.htm>. Acesso em: maio 2011.

MENEZES, Flo. *A acústica musical em palavras e sons*. Cotia: Ateliê Editorial, 2003. 308 p.

MIRANDA, E. R. *Computer sound design: synthesis, techniques and programming*. 2. ed. Woburn: Focal Press, 2002. 263 p.

MONTEIRO, A. C.; MANZOLLI, J. Análise computacional de texturas sonoras via mapas de Poincaré. *Anais da Associação Nacional de Pesquisa e Pós-Graduação em Música*, Uberlândia, p. 1500-1506, 2011. 21º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, Uberlândia, 2011.

NANCE, R. W. *Compositional explorations of plastic sound*. Leicester, 2007. 65f. Tese (Doutorado em Música). De Montfort University.

OLIVEIRA, B. de M.; PEREIRA, C. A. A.; LUZ, J. F. R.; RIBEIRO, S. T. da S.; TRALDI, C. A. Improvisação livre mediada por dispositivos tecnológicos: reflexões para a educação musical. *Anais da Associação Nacional de Pesquisa e Pós-Graduação em Música*,

Uberlândia, p. 303-309, 2011. 21º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, Uberlândia, 2011.

OSHO. *Inteligência: a resposta criativa ao agora*. São Paulo: Cultrix, 2007. 200 p.

PEREIRA, C. A. A. Estratégias para a concepção de um sistema sonoro-interativo para performance com indeterminação musical. *Anais da Associação Nacional de Pesquisa e Pós-Graduação em Música*, Uberlândia, p. 73-79, 2011. 21º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, Uberlândia, 2011a.

_____. *Atratores Ou-vistos: performance interativa com o SSInPIC*. Video. 2011b. Disponível em: <http://www.youtube.com/watch?v=htWkwEOy_hQ>. Acesso em: 9 dez. 2011.

PESSOA JÚNIOR, O. F. *Auto-organização e complexidade: uma introdução histórica e crítica*. [2001?]. Disponível em: <<http://www.fflch.usp.br/df/opessoa/opessoa-tex.htm>>. Acesso em: 6 out. 2011.

PESTOVA, Xenia. *Models of Interaction in works for piano and live electronics*. Montreal, Quebec, 2008. 90f. Tese (Doutorado em Música). Schulish School of Music, McGill University.

PORRES, A. T. *Tutorial de Pure Data por Alexandre Porres*. 03 fev. 2010. Disponível em: <<http://sites.google.com/site/porres/TutorialPd.zip>> Acesso em: 08 mar. 2010.

PRIMO, Alex. *Enfoques e desfoques no estudo da interação mediada por computador*. 404NotFound, n. 45, 2005. Disponível em: <http://www.facom.ufba.br/ciberpesquisa/404nOtf0und/404_45.htm>. Acesso em: 20 abr. 2010.

PUCKETTE M.; LIPPE C. *Score following in practice*. In: Proceedings of the ICMC92, 1992, San Francisco. International Computer Music Association. p. 182-185, 1992.

PUCKETTE, M. S.; APEL, T.; ZICARELLI, D. Real time audio analysis tools for Pd and MSP. *Proceedings, International Computer Musical Conference*, San Francisco, p. 109-112. 1998. International Computer Music Association, San Francisco, 1998.

PUCKETTE, M. S. *The theory and technique of electronic music*. World Scientific Publishing Co. Pte. Ltd., 2007. Disponível em: <<http://crca.ucsd.edu/~msp/techniques.htm>>. Acesso em: 9 jul. 2010.

_____. *Pure Data*. Pd version 0.41.4-extended. Boston: GNU General Public License, (1996). Disponível em: <<http://puredata.info/downloads>>. Acesso em: 14 nov. 2009.

QUARANTA, D. E. A escrita como meio para a abertura conceitual da obra musical. *Anais da Associação Nacional de Pesquisa e Pós-Graduação em Música*, Salvador, p. 445-449, 2008. 18º Congresso da Associação Nacional de Pesquisa e Pós-Graduação em Música, Salvador, 2008.

ROCHA, F. de O. *A improvisação na música indeterminada: análise e performance de três*

obras brasileiras para percussão. Belo Horizonte, 2001. 133 f. Dissertação (Mestrado em Música). Escola de Música da Universidade Federal de Minas Gerais.

_____. *Works for percussion and computer-based live electronics: aspects of performance with technology*. Montreal, Canadá. Abril, 2008. 61 f. Tese (Doutorado em Música). Schulich School of Music. McGill University.

ROWE, Robert. *Interactive music systems*. Cambridge MA: The MIT Press, 1993.

_____. *The aesthetics of interactive music systems*. In : *Contemporary Music Review*, v. 18-3, p. 83-87, 1999.

_____. *Machine musicianship*. Cambridge: Massachusetts Institute of Technology, 2001.

SANTOS, F. B. B. *Cadeias de Markov: práticas e aplicações*. Tutorial desenvolvido por Fabrício Bueno. CEFET/SC: Araranguá, 2008.

SCHAEFFER, Pierre. *Tratado dos objetos musicais: ensaio interdisciplinar*. Tradução de Ivo Martinazzo. Brasília: Ed. UnB, 1993. 517 p. Tradução de: *Traité des objets musicaux: essai interdisciplines*.

SHANNON, C. E. The mathematical theory of communication. *The Bell System Technical Journal*, [S.l.], jul./out. 1948. vol. 27, p. 379-423, 623-656.

SMALLEY, Denis. The Listening Imagination: Listening in the Electroacoustic Era. Amsterdam: *Contemporary Music Review*, v. 13, p. 77-107, part 2, 1996.

_____. Spectromorphology: explaining sound-shapes. *Organized Sound*, v. 2, n. 2, p. 107-126, 1997.

SUTTON, J. P. *The invisible handshake: an investigation of free musical improvisation as a form of conversation*. Ulster, 2001. 320 f. Tese (Doutorado em Phd). Faculty of Arts, School of Media and Performing Arts (Music), University of Ulster.

TRALDI, C. A.; MANZOLLI, J. Reflexões sobre sistemas sonoros e auto-organização. *Anais do Simpósio de Pesquisa em Música*, Curitiba: Editora da UFPR, v. 1, 2008. 5º Simpósio de Pesquisa em Música, Curitiba, 2008.

TRALDI, C. A. *Percussão e interatividade PRISMA: um modelo de espaço instrumento auto-organizado*. Campinas, 2009. 157 f. Tese (Doutorado em Música). Universidade Estadual de Campinas, Instituto de Artes.

VILLAVICENCIO, C. M. *The discourse of free improvisation: a rhetorical perspective on free improvised music*. Tese (Doutorado em Música) – School of Music, University of East Anglia, Norwich, Reino Unido, 2008.

WISHART, Trevor. *On Sonic Art*. Amsterdam: 2006. 2. ed. *Contemporary Music Studies*, v. 12.

XENAKIS, Iannis. *Formalized Music: thought and mathematics in composition*. New

expanded edition. Pendragon Press, 1992. 389 p.

GLOSSÁRIO

Auto-organização: Segundo Debrun (1996, p. 13) “Há auto-organização cada vez que, a partir de um encontro entre elementos realmente (e não analiticamente) distintos, desenvolve-se uma interação sem supervisor (ou sem supervisor onipotente) – interação essa que leva eventualmente à constituição de uma "forma" ou à reestruturação, por "complexificação", de uma forma já existente”.

Abertura: Eco (1991, p.89) classifica dois tipos de abertura, a de primeiro e a de segundo grau. A abertura de primeiro grau se encontra em qualquer obra artística, pois, segundo o autor “[...] é a condição de toda fruição estética, [...] mesmo quando o artista visa uma comunicação unívoca e não-ambígua.” Em música, pode-se dizer que a cada escuta de uma mesma obra pode haver um novo elemento de estruturação poética. A abertura de segundo grau se diferencia da de primeiro grau pois, vai além da apreciação subjetiva da obra, convidando espectadores a participarem de sua construção poética, ou ainda, utilizando-se de elementos estruturantes tais como o acaso.

Adaptação: Acomodação conveniente e espontânea de um elemento à uma nova informação, o que dá a qualidade de emergência a um sistema.

Atrator: Forças de atração entre as ações dos elementos de um sistema que estabilizam estruturas que emergem de suas interações na busca de uma auto-organização do sistema.

Entropia: É uma medida da complexidade de um sistema, da incerteza da informação de uma mensagem transmitida por um emissor a um receptor.

Redundância: É uma medida quantitativa de ordem em um sistema.

Ruído Informacional: Interferência imprevista na mensagem enviada por um emissor a um receptor.

Sistema: De acordo com a Teoria Geral de Sistemas de Bettarlanfy, sistema é um conjunto de elementos interdependentes - com determinadas funções - que interagem entre si, constituindo um organismo.

ANEXOS

A. DVD: Vídeo da performance “Atratores Ou-vistos”; *Patch* interativo do SSInPIC; Registro de gravação de áudio de duas performances de improvisação com o SSInPIC.