

UNIVERSIDADE ESTADUAL PAULISTA
INSTITUTO DE ARTES
PROGRAMA DE PÓS-GRADUAÇÃO EM MÚSICA

ALDO DE OLIVEIRA CARDOSO

**“RELAÇÕES ENTRE SOFTWARES E COMPOSIÇÃO
NA PEÇA “HISTÓRIAS OCULTAS OU
METAMÚSICA?””**

Monografia apresentada ao Programa de Pós-Graduação em música do Instituto de Artes da UNESP como Trabalho Equivalente à Dissertação segundo Artigo 35 – Parágrafo 3 da Resolução UNESP n. 81, de 25 de setembro de 2002, como requisito parcial para a obtenção do título de Mestre em Música.

Orientador: Prof. Dr. Florivaldo Menezes Filho

SÃO PAULO

2007

Sumário

Resumo 03

Abstract 04

Introdução 05

1 A composição do material sonoro 06

1.1 A relação da interface intuitiva com o compositor 08

1.2 A relação fenomenológica da interface gráfica intuitiva com o compositor 09

1.3 Síntese e composição via programação 10

1.3.1 O programa SuperCollider enquanto ferramenta de síntese 12

1.3.2 Exemplo de um programa de síntese granular em *SuperCollider* 15

1.4 Programa que funciona como instrumento 18

2 A metáfora do *mouse* 21

2.1 Representações fixas 22

2.2 Representações modificáveis 25

2.3 Modificações em arquivos gráficos 28

2.4 O paradigma Max/MSP 31

3 Composição: *Histórias ocultas ou metamúsica?* 34

3.1 A partitura virtual da composição “Histórias ocultas ou metamúsica?” 34

Conclusão 39

Bibliografia 40

Resumo

A temática que norteia o presente trabalho consiste em estudar as influências dos *softwares* de produção sonora na composição eletroacústica. Os ambientes gráficos demonstram grandes similaridades às partituras convencionais. Como demonstração destes meios tecnológicos na composição eletroacústica acusmática, temos como parte do trabalho uma peça eletroacústica acusmática intitulada “Histórias ocultas ou metamúsica?” (gravada em CD) na qual é evidenciado o processo de composição.

Palavras-chave: música eletroacústica acusmática, *software*, composição, representação.

Abstract

Thematic that it guides the present work consists to study the influences of software used to make sound in the electroacoustic composition. The graphical environments demonstrate similarities to the conventional partitions. As demonstration of these technological ways in the acousmatic electroacoustic composition, we have as part of the academics work an acousmatic electroacoustic composition entitled “Hidden histories or metamusic?” (recorded in COMPACT DISC) in which the composition process is evidenced.

Key-words: acousmatic electroacoustic music, software, musical composition, representation.

Introdução

Os novos meios tecnológicos, notadamente a computação aplicada à música, propiciaram novas maneiras de se fazer e observar a composição quanto ao processo composicional. Nas interfaces destes *Softwares* encontramos sistemas de notação, que, às vezes, se assemelham à notação tradicional. Veremos no decorrer do trabalho que os arquivos que geram sons podem ser compostos, tratados e modificados tanto em ambientes gráficos quanto nos de programação.

Como consequência do estudo destes novos meios tecnológicos, foi proposta aqui uma composição eletroacústica acusmática, que é fruto do estudo dos Softwares evidenciados neste trabalho.

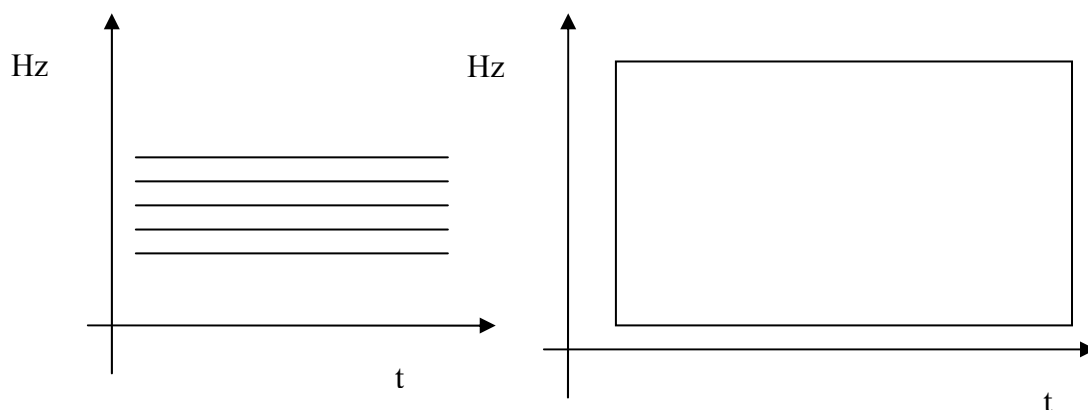
A composição do material sonoro

Se Philippe Manoury difere, em seus textos “La note et le son: un carnet de bord” e “Les partitions virtuelles” (MANOURY, 1998), o conceito de escritura da música instrumental em relação a este mesmo conceito na eletroacústica, é porque na música instrumental o compositor trabalha abstratamente com os parâmetros representados pela notação, a qual demonstra uma ligação estreita entre o objeto musical e seu grafismo (MANOURY, 1998, p. 68).

Quando falamos em material na música eletroacústica, devemos diferenciar duas posições básicas, a saber: o material oriundo de processos de síntese sonora; e o oriundo do tratamento espectral. Um terceiro aspecto do material sonoro pode ser observado nos processamentos que chamamos de *re-síntese*, cujos procedimentos muito se aproximam da idéia de tratamento espectral. Um programa bastante intuitivo que sintetiza, re-sintetiza e faz tratamento espectral de sons importados para seu interior é o *MetaSynth*, que como o próprio nome diz, realiza uma “meta-síntese”.

Nesta linha de *softwares* e também naqueles de programação, podemos observar alguns aspectos importantes no que tange à representação figural nos processos de síntese: a representação visual de um fenômeno acústico ante à sua realização fenomenológica. Os programas de interfaces gráficas classificados como intuitivos possuem uma plataforma que tem como característica dois eixos de coordenadas cartesianas x e y , os quais representam, respectivamente, uma função de tempo e de frequência que muito se assemelha a uma partitura convencional, como podemos ver na figura a seguir.

Fig. 1 Comparação de uma pauta com a representação de uma interface nos programas intuitivos.



Observe que ambas as figuras têm as mesmas características: os eixos de frequência pelo tempo. A primeira figura mostra como a pauta representa as coordenadas cartesianas de tempo por frequência semelhantemente à segunda figura, a qual, por sua vez, representa o campo no qual se inserem os gráficos no programa *MetaSynth*. Para que, neste programa, o compositor possa inserir gráficos que resultarão em eventos sonoros, o *mouse* funciona enquanto metáfora de diversas ferramentas que produzem diversos recursos de construção sonora, tais como granulações, acordes, *clusters*, ruídos, filtros, entre outros.

A questão que aqui queremos elucidar é que o compositor que utiliza esse tipo de programa se depara com um tipo de representação simbólica antes da existência factual do som, o que difere das realizações eletroacústicas feitas, em geral, por meios analógicos. O compositor pode imaginar, a partir dos aparatos oferecidos pelo programa, um som ou evento sonoro, e independentemente da execução pelo programa, o som fica registrado neste tipo de partitura virtual, podendo ser

imaginado ou mentalmente solfejado pelo compositor. É o que lhe permite a representação desta interface gráfica. O programa, por sua vez, tocará o evento sonoro em questão, fazendo o papel de intérprete. Tal fato aproxima o compositor que atua em estúdio da notação musical tradicional.

Em muitos dos processos de síntese sonora que são concebidos a partir deste tipo de programa, é possível armazenar os arquivos em sua forma representacional e reabri-los em outra ocasião para que mínimas partes destes arquivos possam ser modificadas novamente quando o compositor julgar necessário.

A relação da interface intuitiva com o compositor

Diante de uma interface intuitiva, o compositor se depara com um sistema de navegação e de representação bastante simples no sentido de tornar os caminhos de manipulação do som mais curtos e mais fáceis. Os objetos de manipulação sonora costumam estar todos dispostos em uma ordem de agrupamento na qual as ferramentas de tratamento e de síntese podem ser localizadas de acordo com os recursos tipomorfológicos de tratamento ou síntese oferecidos pelo programa, tais como eco, reverberação, ressonadores, controle de parciais, extensão da duração dos sons, granulações, para citar alguns deles.

São esses tipos de programa que podem influenciar “negativamente” o ideário sonoro que o compositor procura, uma vez que tais recursos podem tender a uma qualidade pouco aconselhável para um trabalho criterioso a ser desenvolvido no estúdio. Esses programas geralmente possuem por *default* aspectos sonoros, estruturas sonoras, resoluções de quantização já prontas, nas quais o compositor não poderá interferir, minando, ao menos em parte, seu potencial criativo.

Nestes programas intuitivos é possível, por exemplo, fazer granulações em certas regiões do espectro, mas não é possível ter controle pormenorizado sobre alguns aspectos estruturais dos grãos, já que algumas características já são dadas previamente pelo programa¹.

Esta situação demonstra-se, pois, ambivalente. Por um lado, o compositor parece ter bastante liberdade e facilidade para trabalhar com o seu material, mas, por outro lado, pode encontrar várias limitações em relação aos detalhes estruturais na composição do material sonoro.

A relação fenomenológica da interface gráfica intuitiva com o compositor

A relação fenomenológica do compositor com a interface gráfica e seus recursos visuais e sonoros é bastante diferente da relação que o compositor tem frente às interfaces de programação. Emprestando este raciocínio de outra área onde se estuda, no contexto da educação por ambientes virtuais², a relação de usuários com novos ambientes computacionais, é possível observar quais influências o compositor recebe quando utiliza as interfaces gráficas dos programas intuitivos.

Quando confrontado com esse tipo de interface, o compositor se depara com uma grande gama de possibilidades de ação instantânea. Ele encontra também certa quantidade de ícones que representam cada qual para um conjunto de ações

¹ Obviamente não podemos generalizar esse raciocínio para todos os *softwares* considerados intuitivos, pois sabemos que alguns destes programas não possuem fronteiras rígidas em relação à sua classificação entre intuitivo ou não intuitivo. Sabemos também que alguns programas intuitivos possuem em seu interior ferramentas de programação através das quais o compositor pode trabalhar minuciosamente com parâmetros do som.

² Trata-se de pesquisas relatadas por Dede, Salzman e Loftin (1995) no início do livro *Science Space: virtual realities for learning complex and abstract scientific concepts*.

específicas: um lápis, por exemplo, representa um grupo de ferramentas que “escreve” os processos de síntese na interface; já um símbolo com notas representa um conjunto de filtros harmônicos que podem servir como tratamento espectral, e assim por diante. Existe, pois, uma lógica embutida na distribuição dos ícones pela interface gráfica com a qual o compositor se depara.

Síntese e composição via programação

Com a ferramenta de programação, o compositor tem à sua frente uma imensa gama de recursos de síntese sonora: controles de parâmetros, operações matemáticas, conversores, suporte MIDI, geradores de sinais, modificadores de sinais, entre outros.

Diferentemente dos programas intuitivos, os de programação possuem uma sintaxe baseada em lógicas descritas por comandos, o que configura um grande sistema de rotinas de programação que se correspondem diretamente com os eventos sonoros. Por um lado, a programação oferece uma grande precisão no controle dos fenômenos auditivos, podendo o compositor especificar eventos sonoros com um extraordinário detalhe numérico (ROADS, 1996, p. 786), enquanto que, nos programas gráficos intuitivos, tal controle é determinado pelos chamados *presets*. Por outro lado, em meio a complicadíssimos algoritmos de síntese via programação, em que o som será produzido em tempo diferido, o compositor pode perder o controle gestual de sua construção sonora, uma vez que ele pode distanciar-se substancialmente de um resultado mais palpável em consequência de tais elucubrações.

Pelo viés da programação, temos novamente um código representacional que antecede a produção do som: os *scripts* que gerarão os sons em tempo diferido³.

Temos então, enquanto regra geral, que os *softwares* de programação são os que produzem o som em *tempo diferido* e os *softwares* de interfaces gráficas e intuitivas, o som em *tempo real*⁴.

O programa *CSound* recebe linhas de comando para produzir o som em tempo diferido. Entretanto, o compositor poderá produzir um *script* que tem a finalidade de processar um gesto feito através do *mouse* em tempo real. Temos então duas etapas para a produção do som:

- 1) O *script* que gerará um pequeno programa para processar uma ação em tempo real;

- 2) a ação do compositor através do gesto.

Dessa forma, o compositor constrói seu próprio “programa” para gerar um determinado tipo de som no que diz respeito a seus aspectos tipomorfológicos. A título de exemplo, o compositor já utiliza, na construção desses pequenos programas ou de “meta-instrumentos”, procedimentos composicionais durante a programação. Assim sendo, a escritura também pode estar embutida nesses *scripts*. Um exemplo disso é que, de fato, existem compositores que compõem peças inteiras utilizando exclusivamente o programa *CSound*.

³ Tempo real: qualidade dos programas que produzem som simultaneamente a uma dada ação.

Tempo diferido: qualidade dos programas que produzem o som após a inserção dos dados.

⁴ Muitos programas não têm suas fronteiras bem definidas em relação à sua classificação entre intuitivo ou não intuitivo. Dependerá, por exemplo, da forma pela qual o usuário se utilizará de determinado programa.

Podemos ter em relação à construção sonora dois níveis de representação do evento sonoro. Primeiramente temos o código: programa que gera um tipo de som ou de evento sonoro. A partir deste código, é possível prever quais os tipos de ações serão permitidos pelo programa, mas somente o compositor escolherá quais ações lhe serão de maior interesse composicional.

Analisaremos agora o programa *SuperCollider* que possui a mescla da sintaxe de duas linguagens *orientadas ao objeto*⁵ (a saber; C++ e *Smalltalk*), e as relações que o compositor poderá ter com este *software*.

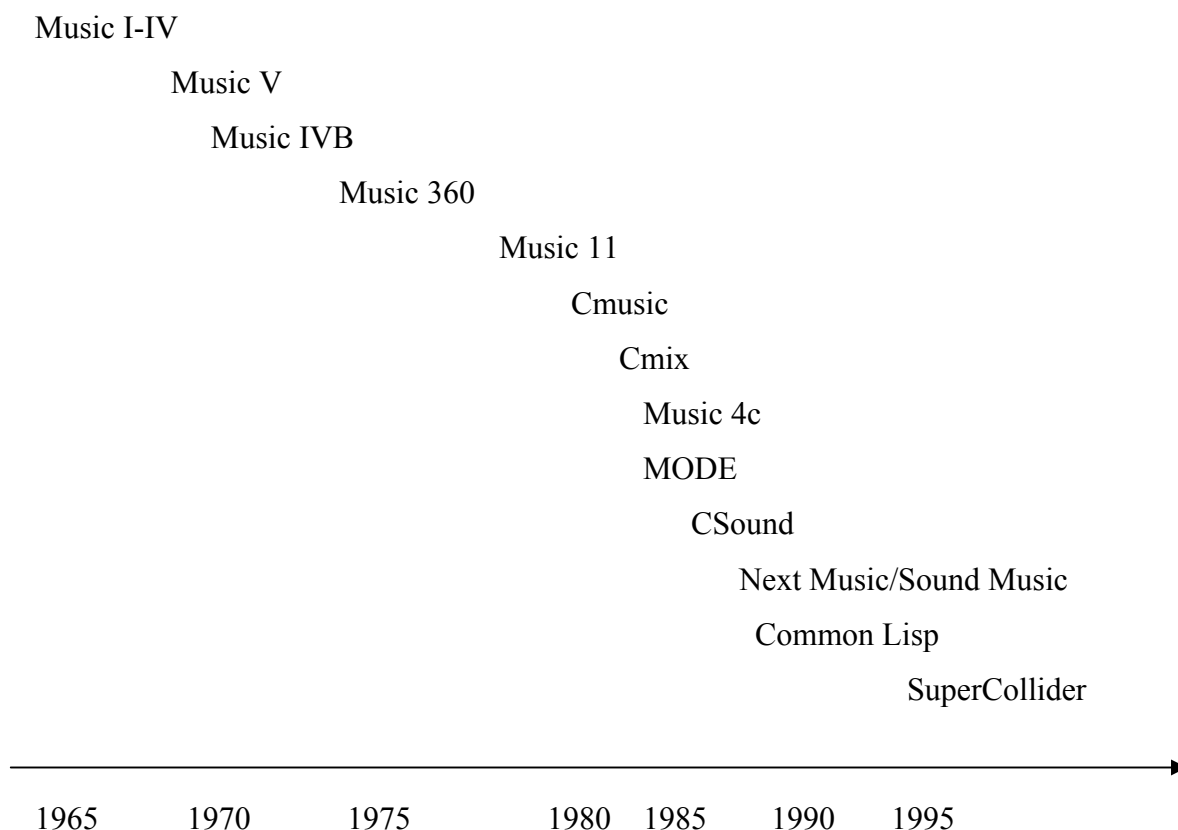
O programa *SuperCollider* enquanto ferramenta de síntese

O programa *SuperCollider* possui uma linguagem bastante flexível e poderosa para síntese e processamento sonoro tanto em tempo diferido quanto em tempo real. Foi desenvolvido a partir de meados da década de 1990 por James McCartney, seguindo a genealogia dos *softwares* de síntese, tal como é ilustrado abaixo⁶:

⁵ Linguagem orientada ao objeto é o tipo de programação que permite ao usuário criar objetos que interagem dentro de um sistema de *Software*.

⁶ Adaptado do tutorial do Programa *SuperCollider 1* de Stephen Pope (1997)

Fig. 2 Linha do tempo dos *softwares* de síntese (POPE, 1997, p. 8).



Este programa tem sua linguagem baseada nas sintaxes das linguagens C++ e *Smalltalk*, sendo que ambas são orientadas ao objeto. Em seu ambiente para desenvolvimento, há um editor de texto, compiladores, sistema para processamento em tempo real, ferramenta de GUI (do inglês, *Graphical User Interface*), e suportes para comandos e controladores MIDI em tempo real. As vantagens que este programa tem em relação aos programas da linha *MUSIC N* consistem nos processamentos em tempo real de sons ou de entradas (*inputs*) com suporte MIDI. Tais processamentos podem se dar pelo uso de uma linguagem orientada ao objeto e de ferramentas GUI, pela qual o compositor pode construir, através de estruturas gráficas, seu “meta-instrumento”, seu programa específico

ou interfaces gráficas interativas para controlar instrumentos externos, sejam eles acústicos ou periféricos acoplados ao computador. Logo abaixo temos um exemplo simples de um oscilador em *SuperCollider*:

```
Defaudioout L, R;          -- define saída dos canais de audio

start{                     -- Função inicial chamada automaticamente

  var osc, outval;         -- Declara nomes de variáveis para o oscilador

  osc = Asinosc(440, 0)    -- Cria um objeto oscilador senoidal de 440 Hz

  dspAdd( {                -- Cria a iteração do sinal (loop contínuo)

    outval = value(osc);

    out(ouval, L);         -- Envia o sinal para o canal esquerdo

    out(ouval, R);         -- Envia o sinal para o canal direito

  });                      -- Finaliza o loop do DSP

}
```

Note que este programa é baseado no conceito de *geradores de unidades*⁷, mas não possui as sub-linguagens “orquestra” e “partitura” que são derivadas desses geradores (ROADS, 1996, p. 788).

⁷ Os geradores de unidades são módulos de processamento de sinal, tais como os osciladores, os filtros, os amplificadores etc. Tais módulos podem ser interconectados para formar instrumentos de síntese que geram sinais sonoros (ROADS, 1996, p. 788).

Exemplo de um programa de síntese granular em *SuperCollider*

No que tange aos aspectos escritos para a constituição do material composicional da música eletroacústica, podemos classificar dois grandes níveis nos quais o compositor pode agir para construir seu aparato sonoro:

- 1) A construção de um objeto que resulta diretamente em uma síntese sonora;
- 2) A construção de um objeto concebido para construir a síntese a partir de controles de gestos na interface. Tal objeto funciona enquanto instrumento ou “compositor”, dependendo da dimensão de sua estrutura.

Temos abaixo um exemplo de objeto que resulta em uma síntese granular, no qual o compositor pode programar diretamente o resultado sonoro esperado.

Script do programa de síntese granular concebido pelo autor para suas peças “Point, ligne et plan sur les temps e Histórias ocultas ou metamúsica?”:

```
defaudioout L, R;                                -- Declara a saída.

deftable tabl1, env1;                             -- Declara tabela de onda para
envelope.

start {                                           -- Toca a grade na função n.º 1
```

--Argumentos são: time

```
instrument dur pitch amp  
[0.00, `chorus_instr, 0.10, 48, 0.2].sched;  
[0.10, `chorus_instr, 0.10, 50, 0.2].sched;  
[0.20, `chorus_instr, 0.10, 52, 0.2].sched;  
[0.30, `chorus_instr, 0.10, 53, 0.2].sched;  
[0.40, `chorus_instr, 0.10, 55, 0.2].sched;  
[0.50, `chorus_instr, 0.10, 55, 0.2].sched;  
[0.60, `chorus_instr, 0.10, 55, 0.2].sched;  
[0.60, `chorus_instr, 0.10, 59, 0.2].sched;  
[0.70, `chorus_instr, 0.10, 63, 0.2].sched;  
[0.80, `chorus_instr, 0.10, 67, 0.2].sched;  
[0.90, `chorus_instr, 0.10, 70, 0.2].sched;  
[1.00, `chorus_instr, 0.10, 73, 0.2].sched;  
[1.10, `chorus_instr, 0.10, 76, 0.2].sched;  
[1.20, `chorus_instr, 0.10, 79, 0.2].sched;  
[1.30, `chorus_instr, 0.10, 80, 0.2].sched;  
[1.40, `chorus_instr, 0.10, 85, 0.2].sched;  
[1.60, `chorus_instr, 0.10, 86, 0.2].sched;  
[1.80, `chorus_instr, 0.10, 88, 0.2].sched;  
[3.00, `chorus_instr, 4.00, 90, 0.2].sched;  
[4.50, `chorus_instr, 3.00, 93, 0.2].sched;  
[5.00, `chorus_instr, 2.50, 95, 0.2].sched;  
[6.00, `chorus_instr, 2.00, 98, 0.2].sched;
```

```

[6.00, `chorus_instr, 1.00, 99, 0.2].sched;

[7.00, `chorus_instr, 0.50, 100, 0.2].sched;
}

chorus_instr {                                     --Faz nova função chorus.

    arg dur, pitch, amp;                            --Declara    argumentos    do
instrumento.

                                                    --Argumentos são: table, freq,
beats/sec

    osc = Acoscili(tabl1, pitch.midi cps, 80);

                                                    -- Envelope de função transiente
                                                    --Argumentos  são: table, dur,
amp, bias, completionFunction
amp = Ktransient(env1, dur, amp, 0, `dspRemove);

                                                    --DSP loop -- send (osc * env)
para as duas saídas de áudio.

    { (osc.value *! amp.value).out(L).out(R) }.dspAdd;
}

```

Note que os grãos sonoros são adicionados pouco a pouco no decorrer tempo, cujas frequências (no argumento *pitch*) são dadas pelos valores padronizados do suporte MIDI.

É interessante salientar que ao compositor é permitido, a partir do programa *SuperCollider*, compilar e executar o som por diversas vezes, podendo fazer mudanças permitidas nos parâmetros sonoros diretamente no *script*, cujo procedimento poderá ser compilado e ouvido em uma próxima etapa. Uma outra faceta permitida pelo programa *SuperCollider* que veremos agora é a possibilidade de se construírem programas que funcionam como instrumento.

Programa que funciona como instrumento

Independente da discussão acerca da estética musical em torno da música contemporânea envolvendo a computação com programas, sistemas de interação, inteligência artificial, robótica, entre outros, sabemos que existem programas computacionais que podem funcionar como “compositores”, tendo como exemplo clássico a Illiac Suíte de 1957 (contemporânea ao *Music I*), composta para quarteto de cordas com as regras de contraponto modal, mas que usa um algoritmo construído, sobretudo, por Lejaren Hiller para o computador Illiac I. Nesse sentido, temos ainda, em contexto tecnológico bem distinto, um exemplo brasileiro: o RoBoser, desenvolvido nos últimos anos no NICS (Núcleo Interdisciplinar de Comunicação Sonora) da UNICAMP, entre outros, pelo Professor Jônatas Mazolli. Trata-se, *grosso modo*, de um robô que compõe música ao se movimentar e interagir com o meio ambiente.

Porém, essa discussão foge ao escopo deste trabalho. Trataremos aqui somente do conceito de um programa funcionando enquanto instrumento que pode ser construído e tocado pelo próprio compositor.

Tomemos como exemplo um instrumento concebido em *SuperCollider* que possui 6 *faders* (potenciômetros) que funcionam como filtros de um evento sonoro oriundo de um conjunto de osciladores acoplados uns aos outros, o que se condiciona segundo o algoritmo escolhido. Tais *faders* podem ser manipulados através de gestos com o *mouse* de acordo com a idéia de material que o compositor tem em mente. Segue abaixo o exemplo do código do programa intitulado “Pipe” e adaptado do tutorial do *SuperCollider* para a composição Histórias ocultas ou metamúsica?:

“Instrumento Pipe”

```
defaudioin Lin Rin;

defaudioout L R;

defdelay pipe(0.05) pipe2(0.05) pipe3(0.05);

init {

}

Start {

instr1(1.0, pipe1);

instr1(0.5, pipe2);

instr1(0.75, pipe3);

}

instr1 { arg freq pipe;

n1 = Anoise;

n2 = Asinosc;

o1 = Asiosc;
```

```

k1 = Klag;
k2 = Klag;
k3 = Klag;
k4 = Klag;
k5 = Klag;
k6 = Klag;
f1 = Alpf;
{
    (o1.(k6.(6.getItemValue, 0.4))
        *! n1.(k1.(1.getItemValue, 0.4), .(k2.(5.getItemValue, 0.4)))
    .out (pipe);
    f1.(pipe.tapi (k3.(2.getItem value * freq, 0.4)).out(L).out(R)
        *! k4.(3.getItemValue.neg, 0.4), k5.(4.getItemValue, 0.4)).out(pipe);
    }.dspAdd;
}

```

Trata-se de um pequeno programa que funciona como instrumento, já que temos um conjunto de osciladores ligados a algumas ações e seis *faders* declarados pela letra “k”. Nas linhas finais do *script*, são construídas as ações consignadas a esses botões. No caso da peça Histórias ocultas ou metamúsica, este *script*, com ligeiras mudanças, é responsável pelos sons de ataque de ressonância.

Como vimos, o compositor torna-se um *luthier* e elabora seu próprio instrumento para tocá-lo em seguida. Entretanto, surge uma questão: o compositor compõe seu instrumento? A resposta, neste caso, pode ser *sim*, se pensarmos que ao construir

seu instrumento o compositor já elabora os filtros, os osciladores, e que quando tal *script* é compilado e executado, surgirá simultaneamente na tela do computador – no caso do “Instrumento Pipe” – um som estático e contínuo, assim como um conjunto de *faders* para manipulação em tempo real pelo gesto do compositor na tela do computador. Se o compositor não realiza nenhuma ação, o som permanece contínuo. Dessa maneira, ao fazer seu instrumento, o compositor também já está compondo uma realidade sonora mesmo que seja do ponto de vista do material sonoro que, por sua vez, também está ligado à idéia de escritura musical.

Com tal conjunto de *faders*, o compositor poderá fazer diversas transformações do evento sonoro através de gestos. Os resultados obtidos através dos gestos poderão ser armazenados em um arquivo de som, o qual poderá constituir parte do material de uma futura composição eletroacústica. É possível ainda, com mais recursos de programação, registrar e gravar visualmente os gestos procedidos com o *mouse* na interface gráfica, juntamente com os efeitos sonoros dados pelo instrumento.

A metáfora do *mouse*

Como já dissemos, o *mouse* tornou-se metáfora da mão que manipula muitas outras ferramentas na interface gráfica de alguns programas. Inventado em 1968 pelo americano Douglas Engelbart, mas somente incorporado, modificado e popularizado na computação pela *Apple* em 1982 com o sistema de “apontar” e “clique”, o *mouse* tornou-se um importante periférico, não somente para a computação em geral, mas também para o controle gestual sonoro em inúmeros

programas de música. O programa *Music Mouse* de Laurie Spiegel, concebido em meados dos anos 1980, foi um exemplo clássico para a escrita baseada em um piano virtual controlado pelas ações do *mouse*.

Todavia, inúmeros periféricos, desde *joysticks*, teclados, até “mesas” de sons, podem ser acoplados aos programas, permitindo ações simultâneas e diacrônicas de controle do som por diversos aparatos gráficos que o compositor pode construir.

Os seis *faders* do programa “Pipe” poderiam ser acoplados a uma mesa para serem controlados simultaneamente pelos dedos do compositor, e unicamente pelo uso do *mouse* seria impossível de se controlar o resultado sonoro em tempo real.

Nesse contexto, a ação do compositor sobre esses instrumentos virtuais não corresponde diretamente ao efeito oriundo da ação. A diferença desses instrumentos em relação aos tradicionais está na relação de causalidade⁸.

Nesse sentido, o aspecto da causalidade constituiu-se numa crucial diferença para os novos conceitos de instrumento: a causalidade explícita dos meios analógicos é modificada nos meios digitais, pois há diversas camadas simbólicas entre um objeto ou uma ação e sua representação em dígitos.

Representações fixas

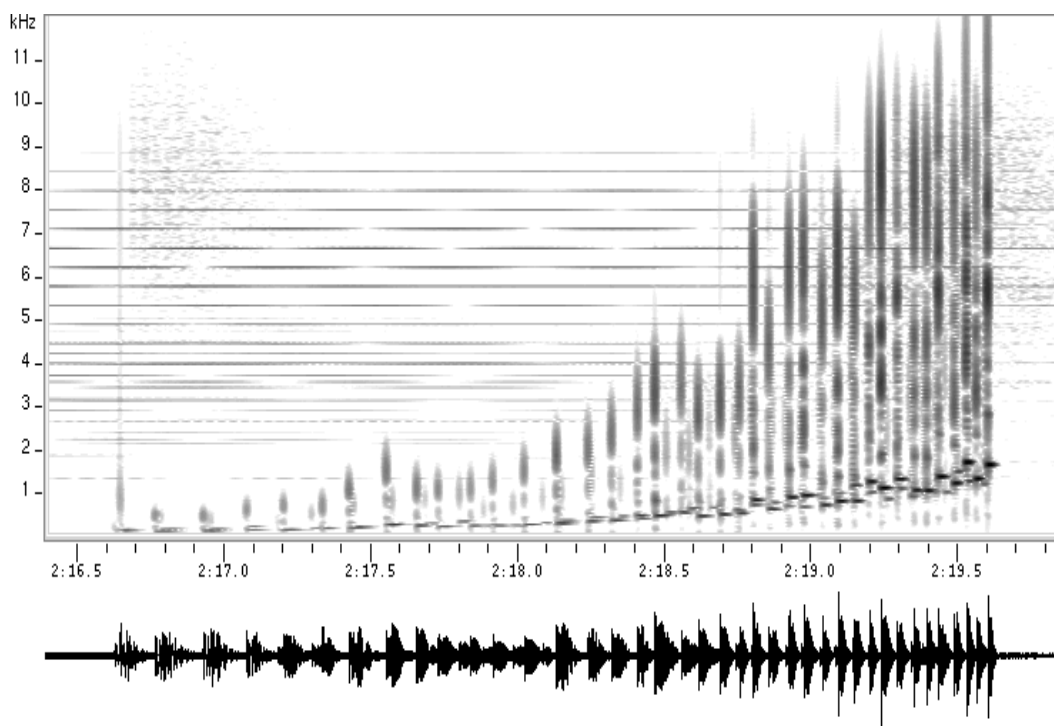
Temos atualmente uma parcela muito grande de *softwares* direcionados à produção e manipulação do som. Alguns deles podem gerar fonogramas dos

⁸ Com exceção do órgão de tubos, no qual a ação é intermediada pelos mecanismos de produção de ar. Ao pressionar uma tecla com diversas intensidades o músico tem, teoricamente, o mesmo resultado. Diferentemente do piano, onde se tem o controle de intensidade e até mesmo de timbre (segundo Cláudio Richerme (1996) em seu livro “A técnica pianística: uma abordagem científica”).

parâmetros sonoros ou ainda imagens gráficas de manipulação vetorial. Contudo, estes fonogramas, diferentemente das imagens que podem ser manipuladas na interface gráfica, são imagens fixas do evento sonoro, no sentido em que o compositor não altera o som a partir da manipulação direta sobre a imagem na interface gráfica, diferentemente dos sonogramas ou de outros tipos de imagens que podem ser manipuláveis, como no programa *AudioSculpt*.

Temos aqui um exemplo de fonograma sincronizado com seu padrão *waveform* logo abaixo. Os fonogramas podem ser extraídos a partir da análise de um *sample*, ou ainda a partir do resultado espectral de uma síntese sonora através da FFT.

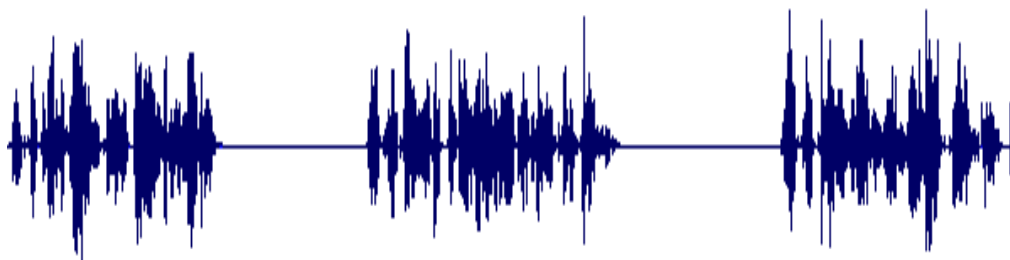
Fig. 3 Representação do som por um fonograma.



Vemos a partir da figura que este gráfico consiste em um plano cartesiano de frequência por tempo. As regiões mais escuras representam sons com maior amplitude; as mais claras, regiões de menor amplitude. É importante salientar que mesmo sem ouvir esse evento sonoro, é possível fazer mentalmente uma imagem acústica desse evento. Vemos, por exemplo, sons contínuos mais acima e ataques mais abaixo que vão crescendo tanto em densidade temporal e espectral quanto em amplitude.

Alguns *softwares* de edição, de masterização e de mixagem do som também possuem representações fixas do som, como os padrões de *waveform*, nos quais as ondas sonoras são representadas em gráficos geralmente bidimensionais de amplitude por tempo.

Fig. 4 Representação do som por *waveform*.



Esse gráfico representa o som através da somatória de todos os parciais que compõem esse evento sonoro resultante, tornando mais difícil conceber uma imagem acústica do som antes de sua escuta. Embora denominemos estes arquivos visuais de fixos, existem alguns recursos no ambiente dos programas de edição que possibilitam algum tipo de tratamento acústico dos sons: são os *plugins*, que nada mais são que programas menores que se acoplam aos programas

maiores, assumindo usualmente uma função específica. Outros recursos, tais como envelopes sonoros, *pan* (para escrita do espaço ou espacialização), entre outros, permitem certas manipulações acústicas nesses eventos sonoros.

Todavia, a manipulação acústica desses eventos sonoros, a partir do próprio *software* de edição via *plug-ins*⁹, pode deixar a desejar no que tange à qualidade estética dos processamentos dados por tais *plug-ins*. Trata-se da questão da realização técnica de uma peça eletroacústica.

O compositor pode preferir reabrir os arquivos em seus programas de origem ou em outros programas especializados para realizar um certo tratamento sonoro, preterindo tratamentos realizados em programas de edição.

Quando se abrem estes arquivos em seus programas de origem, eles se tornam manipuláveis. Assim sendo, o compositor poderá manipulá-los novamente em seus pormenores com todos os vários recursos oferecidos pelos programas nos quais esses sons foram originalmente concebidos.

Representações modificáveis

São inúmeros os tipos de arquivos de som que podem ser modificáveis. De um modo geral, os arquivos produzidos por um determinado programa podem ser modificados quando reabertos neste mesmo programa.

Os arquivos de programação, bastante usados em processos de síntese sonora, são representações modificáveis por excelência, já que o compositor pode modificar o código em seus mínimos detalhes.

⁹ No contexto da computação são pequenos programas que se acoplam aos programas maiores.

As modificações podem transformar o evento sonoro sem mudar a estrutura sintática do programa ou alterando completamente a estrutura do *script*, uma vez já programado.

Temos abaixo um oscilador que pode ser modificado sem que se altere sua estrutura de sintaxe.

Exemplo da modificação de um *script* em *SuperCollider*.

```
Defaudioout L, R;  
  
Start {  
  
  Var, osc, outval;  
  
  Osc = Asinosci (220,0)  
  
  dspAdd ( {  
  
    outval = value(osc);  
  
    out (outval, L);  
  
    out (outval, R);  
  
  } )  
  
}
```

Observe que tal *script* consiste em um simples oscilador de uma senóide, no caso um *lá* 220Hz. Contudo, se observarmos novamente o mesmo *script* abaixo, veremos que este se encontra ligeiramente modificado. Sua frequência sonora é modificada sem que se altere qualquer aspecto de sua estrutura sintática.

```
Defaudioout L, R;
```

```

Start {
  Var, osc, outval;
  Osc = Asinosci (440,0)
  dspAdd ( {
    outval = value(osc);
    out (outval, L);
    out (outval, R);
  } )
}

```

O leitor pode observar, agora, que nossa frequência “lá” 220 Hz passou para uma oitava acima, tendo 440 Hz com a mesma estrutura do programa. Salientamos que nem todos os geradores de unidades destes programas aceitam qualquer valor. Por um lado, há limitações impostas em determinadas estruturas de *scripts* que combinam geradores de unidades trabalhando com um certo limiar de compatibilidade numérica entre si. Por outro lado, o compositor deverá ter noções no âmbito da acústica para que as modificações e construções de seus programas funcionem corretamente.

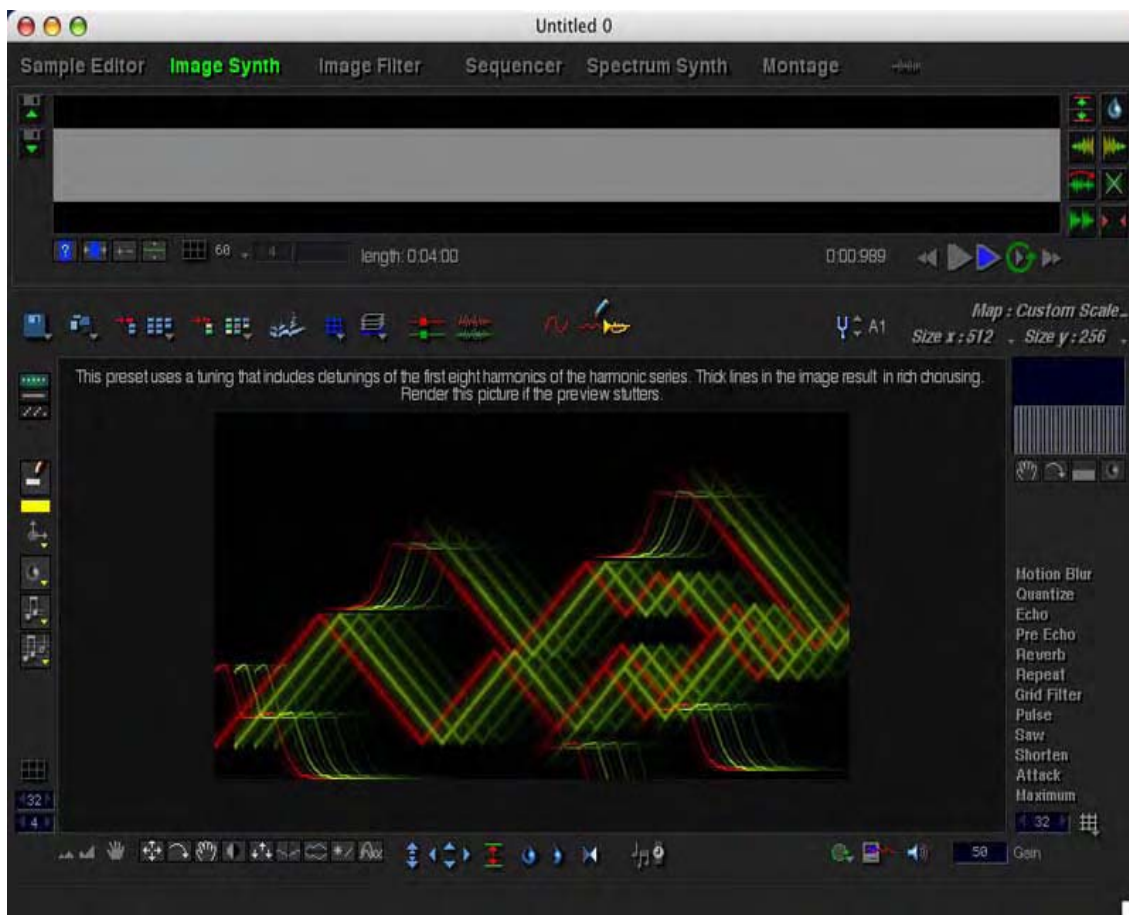
Vamos a um exemplo bastante simples: se substituíssemos a frequência de 220 Hz por uma de 30.000 Hz, o programa compilaria e executaria a mudança, mas o compositor não ouviria o som, pois este estaria acima do nosso limiar auditivo.

Modificações em arquivos gráficos

Muitos arquivos gráficos que, para constituição visual da representação sonora, necessitam do *mouse* enquanto metáfora, também podem ser modificados ou reconstruídos de acordo com a ferramenta que o *mouse* representa. Todavia, a modificação é feita de acordo com o *default* do programa, ou seja, não se pode modificar uma estrutura mínima que já é dada prontamente pelo programa, pois estas estruturas mínimas já estão programadas dentro do *software*.

Este é um exemplo de arquivo gráfico no programa *MetaSynth* 4.0 com a janela gráfica chamada de *Image Synth*, na qual o usuário constrói sínteses sonoras através de gráficos utilizando ferramentas específicas:

Fig. 5 Gráfico no *Image Synth*, programa *MetaSynth* 4.0.



O compositor pode agir diretamente no gráfico através de diversas ferramentas, podendo apagar ou substituir as linhas dispostas em um plano cartesiano com as coordenadas de frequência por tempo.

São muitos os programas que possuem tipos de arquivos gráficos modificáveis. Tratamos aqui apenas de um deles, mas que representa uma grande classe de programas que geram tais arquivos.

Os sonogramas consistem em um tipo de gráfico semelhante aos fonogramas, entretanto os sonogramas são resultados de uma análise do som e, no ambiente do

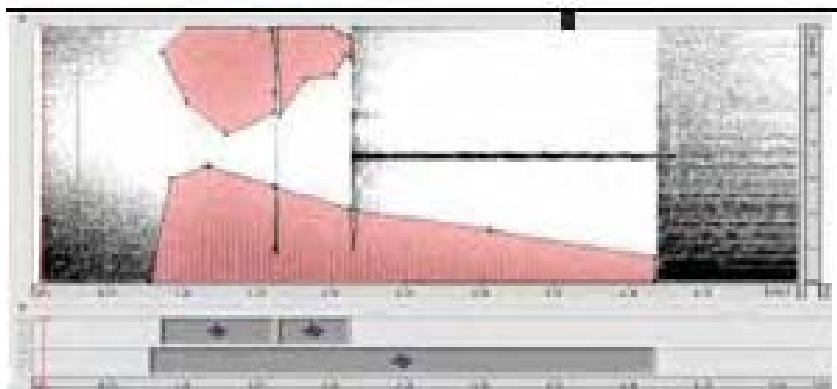
software que os produzem, demonstram-se maleáveis. Neles o compositor pode verificar cada componente do espectro sonoro por meio de ferramentas via *mouse*.

O programa *AudioSculpt* (concebido no IRCAM) gera este tipo de arquivo modificável. O compositor pode trabalhar diretamente sobre a imagem do espectro sonoro através de inúmeras formas possibilitadas pelo programa. São elas: cortar o som, selecionar tanto partes específicas quanto um único parcial do espectro, filtrar o espectro, entre muitas outras ações. Ou seja, ele permite ao compositor “esculpir” visualmente o som.

Este programa é baseado no tipo de análise chamada *Vocoder*¹⁰, permitindo também modificações de escalas temporais do tipo compressão e dilatação temporal do material sonoro, transposição, sínteses cruzadas, entre outros. Logo abaixo temos um exemplo de sonograma modificado:

¹⁰ Nome derivado de “voice coder”, trata-se de um algoritmo criado para analisar e sintetizar a voz humana, bastante usado nos sistemas digitais de telecomunicações.

Fig. 6 Exemplo de sonograma no programa *AudioSculpt 2*, Bogaards, Röbel, Rodet (2004, p. 4).



Neste caso, temos uma região de filtros que atenua a área de sons indesejados, trabalho feito a partir de um espectro aberto no programa.

O paradigma Max/MSP

O programa *Max/MSP* consiste em uma linguagem de programação gráfica e foi desenvolvido no “Institut de Recherches et Coordination Acoustique/Musique” (IRCAM) em Paris em 1986, tendo como principal autor Miller Puckette. Seu desenvolvimento se deu durante toda a década de oitenta, tendo uma grande variedade de influências. Seu principal antecessor foi o programa RTSKED de Max Mathews (PUCKETTE, 2002, p. 31) (daí o nome Max, em homenagem a Max Mathews).

Em 1985 o programa *Max* rodou pela primeira vez, tendo como suporte o programa *Synthetic Performer* desenvolvido por Barry Vercoe e M. Puckette no

IRCAM. Os recursos para formar os *patches*¹¹ eram constituídos por somente quatro objetos para manipulação do som (PUCKETTE, 2002, p. 34). Finalmente em 1987 surge a versão do programa *Max* concebida para os computadores 4x, tendo ocasionado a estréia de importantes obras, tais como “Jupiter” de Manoury. Posteriormente *Max* é reescrito da linguagem Lisp para a linguagem “C” para o computador *Macintosh*, ganhando a extensão MSP (Musical Signal Processing), o que o tornou bastante próximo da versão atual.

Este *software* consiste em um ambiente de programação gráfica com herança de programação modular em que o usuário pode programar, utilizando enquanto metáfora os sintetizadores analógicos.

Neste programa, os módulos são conectados uns aos outros de acordo com regras sintáticas específicas. A junção destes módulos forma o que chamamos de *patch* e estes, por sua vez, possuem funções específicas.

O programa *Max*, sob o ponto de vista de construção via programação modular, pode ser considerado intuitivo (WINKLER, 1998, p. 109). É possível fazer diversos tipos de construções modulares de acordo com as necessidades de quem o utiliza. Temos exemplos de *patches*: um sintetizador, um instrumento, um hiperinstrumento, um “compositor” (um algoritmo que pode gerar uma cadeia de eventos sonoros a partir ou não de eventos exteriores ao programa, seja via MIDI ou através de outra interface), um controlador ou disparador de eventos, entre outros. Outra característica importante é a possibilidade de se esconderem as estruturas complexas dos *patches*, deixando somente visível as estruturas essenciais, ou seja, é possível visualizar os *patches* sob pelo menos duas formas: a forma visual (a interface gráfica que consiste no programa representado pelo

¹¹ Objetos conectados uns aos outros obedecendo a regras sintáticas formam o que chamamos de *patch*, veremos com mais detalhes neste tópico.

patch); e a forma estrutural (a interface modificável na qual são mostradas as conexões entre os módulos ou objetos que formam o *patch*).

Os *patches*, por sua vez, podem se conectar uns aos outros e sob este ponto de vista, são construídos por objetos menores. Nesse sentido, passam por uma mudança qualitativa, ou seja, um *patch* feito de objetos menores ligados a outros *patches* torna-se ele mesmo um “objeto” maior que pode tornar-se um “link” a ser aberto em outras “janelas” com interfaces gráficas. Para o leitor não familiarizado com este ambiente, podemos fazer uma analogia com uma página da *internet*, na qual o usuário se depara com a página principal (designada geralmente por *home*) que, ao seu turno, nos remete a vários *links*, os quais, por sua vez, nos remetem a novas páginas em novas janelas do navegador (nesse caso, as chamadas *pop-ups*). Estas possuem novos *links*, e assim por diante. Todavia, no ambiente Max os novos *patches* são susceptíveis a modificações em sua forma estrutural.¹²

¹² Ressaltamos que essa visualização na qual os *patches* podem formar outro objeto maior, trata-se de uma abstração, já que, um mesmo *patch* pode ser visto sob diversos aspectos.

Composição: *Histórias ocultas ou metamúsica?*

A primeira idéia mais geral que surgiu para a composição desta peça foi uma metáfora de “portas que abrem”, de onde os eventos surgem e acabam de acordo com a abertura e fechamento de portas no decorrer do tempo. Na abertura da última porta, ao final da peça, o primeiro evento que antecede a abertura da primeira porta é repetido com o intuito de se construir uma metalinguagem, ou seja, a música “falando” dela mesma.

A peça acusmática aqui apresentada envolve tanto processos de síntese quanto os de tratamento do som. Tal peça possui apenas caráter didático com o intuito de expor as idéias aqui apresentadas.

Os principais programas utilizados na peça foram o *SuperCollider* para síntese, o *SoundMaker*, para tratamento e o *Pro Tools* para edição. A peça, apesar de possuir um caráter híbrido possui maior referencialidade no terço final, onde é colocado alguns eventos simultâneos, que foram apresentados inicialmente, lembrando a exposição de uma sonata.

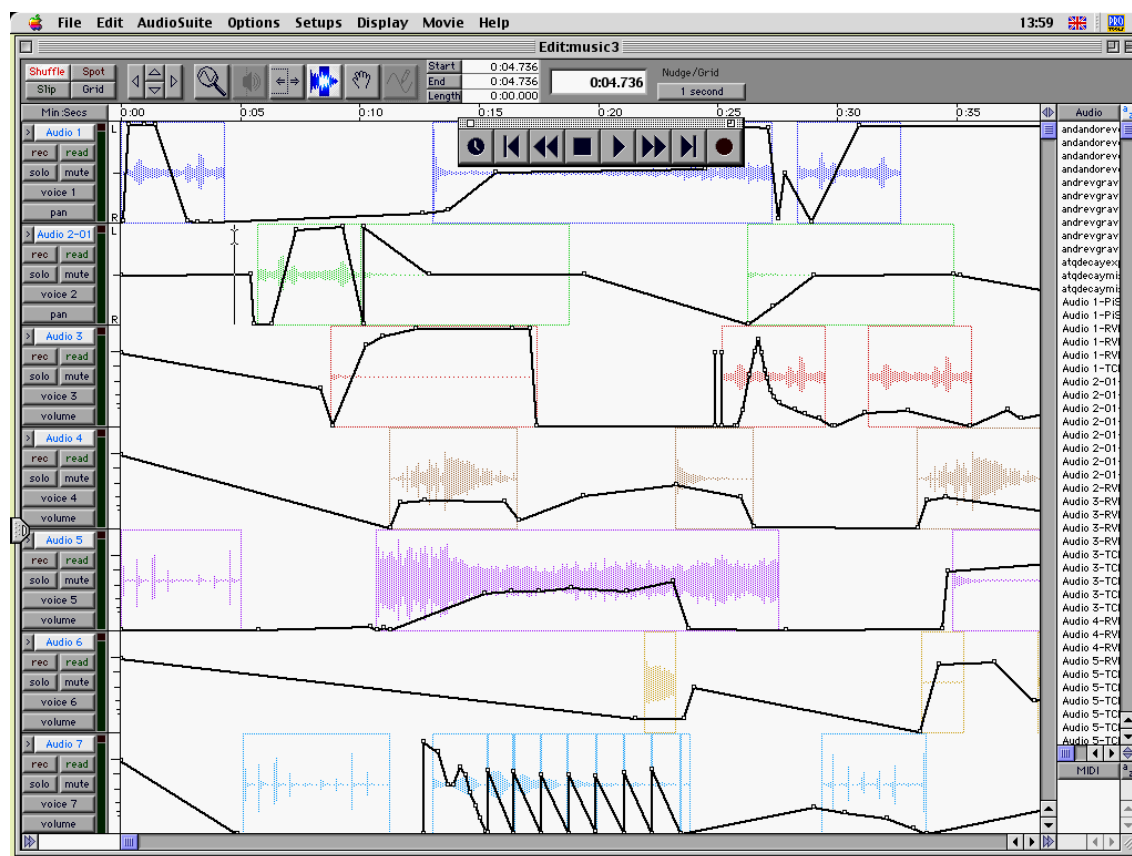
O aspecto formal pode ser observado sob dois aspectos; constituição em blocos de variados tamanhos e forma A, B, A’ já que as seções são mais ou menos delimitadas pela diferença tipomorfológica do material.

A partitura virtual da composição “Histórias ocultas ou metamúsica?”

Temos logo abaixo a figura extraída da seção no programa de edição *Pro Tools* com várias “pistas” que compõem a peça. Essa interface possui várias camadas de representação. Nessa primeira, podemos observar no primeiro bloco da primeira

“pista” algumas linhas que são um tipo de contorno. Estas linhas, nesse caso, correspondem à espacialização estereofônica da peça.

Fig. 7 Início da peça na seção no programa *Pro Tools*

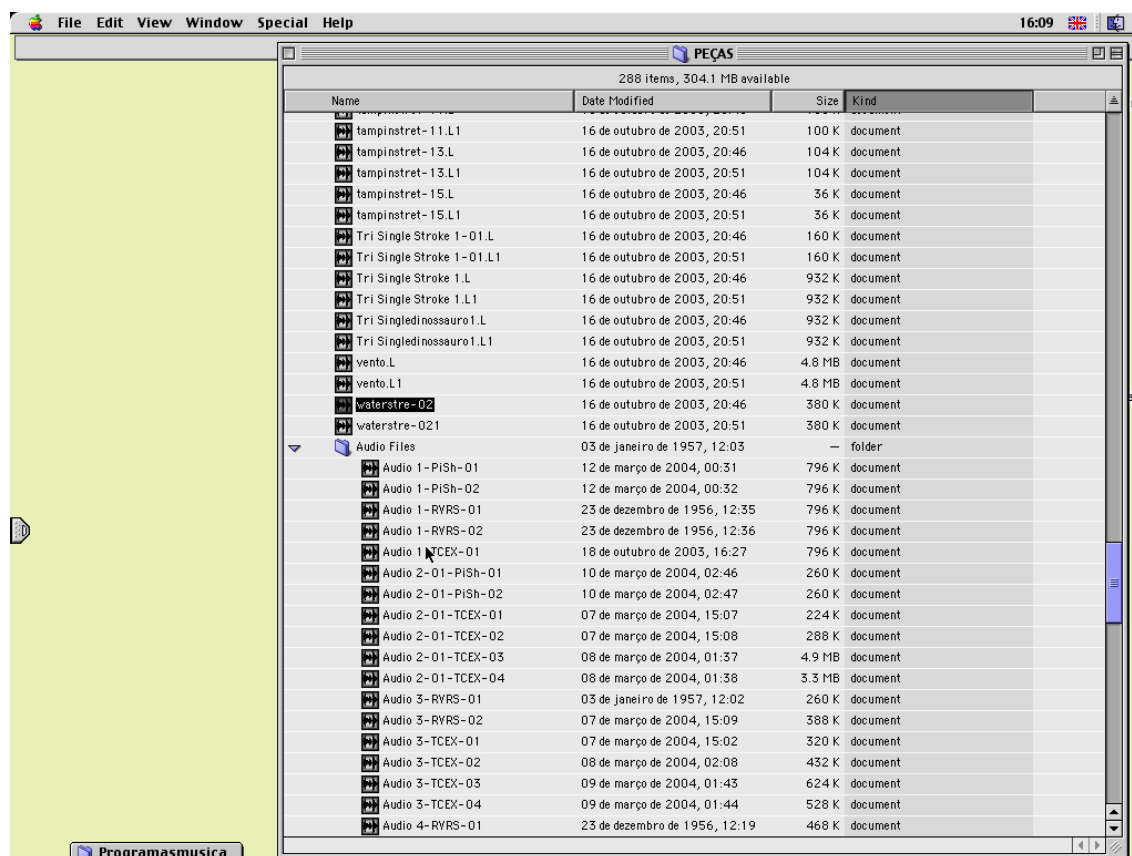


Na próxima figura podemos ver o mesmo bloco, desta vez com a linha de volume.

The screenshot displays the Audacity 2.0.5 software interface. At the top, the menu bar includes File, Edit, AudioSuite, Options, Setups, Display, Movie, and Help. The title bar indicates the current project is 'Edit: music3'. Below the menu bar is a toolbar with various editing tools, including a selection tool, a zoom tool, a pan tool, and a solo tool. The main workspace shows seven audio tracks, each with a volume envelope and a waveform. The tracks are labeled Audio 1 through Audio 7. The top status bar shows the current time as 0:04.736. The right sidebar displays a list of audio tracks and their associated audio files.

Todavia, esse mesmo arquivo pode ser aberto no programa que o gerou.

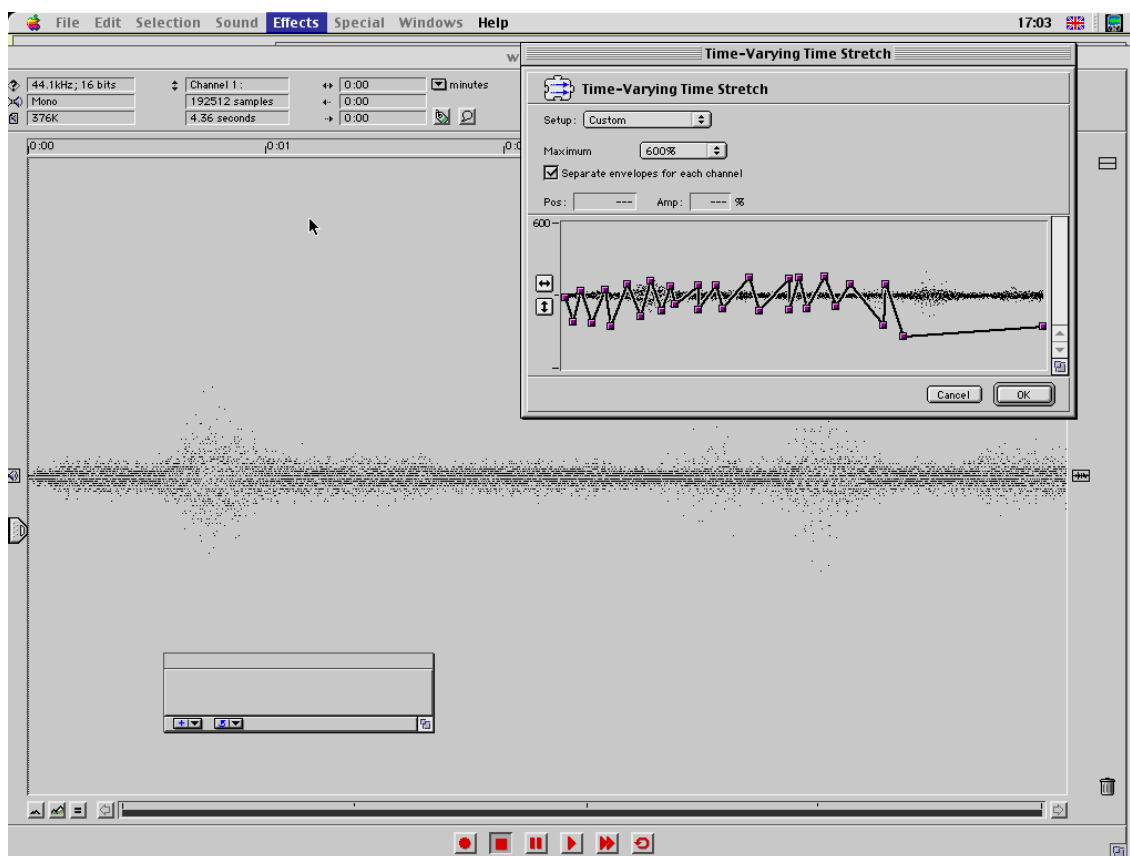
Fig. 9 Arquivo dentro da pasta do programa de edição, ao todo os arquivos que totalizam a peça são mais de trezentos.



Temos aqui o arquivo no programa inicial de tratamento, a saber; o *SoundMaker*.

A figura a seguir trata-se de apenas uma das possibilidades do programa e no nosso caso mostra como o bloco sonoro foi esticado no tempo através das linhas dentro do bloco.

Fig. 10 Arquivo aberto no programa de tratamento, *SoundMaker*, as linhas sobre o arquivo na janela menor consistem em um “time stretch” variável.



Notamos aí um nível mais aprofundado de representação do arquivo sonoro. Nesse tipo de representação, podemos trabalhar diretamente sobre o gráfico novamente para enviá-lo ao programa *Pro Tools*.

Esse tipo de arquivo é modificável, como vimos anteriormente, já que podemos interferir na estrutura do som através do *mouse*. Desse modo, os arquivos da peça, em sua maioria, são arquivos modificáveis.

Conclusão

A música eletroacústica composta em estúdios com recursos digitais ganha outros níveis de representações figurais que são oriundas dos *softwares* de síntese e tratamento sonoros. Tais representações permitem ao compositor utilizar as vantagens destes recursos simbólicos para compor o material de sua peça. A composição da peça aqui apresentada “Histórias ocultas ou metamúsica?” foi amparada por esses recursos simbólicos que os *softwares* nos permitiram. Vimos como exemplo um arquivo representado em meios diferentes; do *software* de edição ao de síntese ou tratamento.

Assim sendo, alguns recursos no âmbito dos *softwares* de manipulação sonora (programação, recursos gráficos, recursos tecnológicos etc.) apresentados neste trabalho puderam ser vistos na composição enquanto resultados técnico e estético: uma peça eletroacústica acusmática híbrida, mesclando sons sintéticos aos sons concretos tratados via *softwares*.

Bibliografia geral e específica

BAYLE, François. **Musique Acousmatique, propositions... positions**, Paris Buchet/Chastel, Bibliothèque de Recherche Musicale, 1993.

BOGAARDS, Niels et al. **Sound analysis and processing with AudioSculpt 2**, Paris, Mediatheque IRCAM, 2004.

BOULANGER, Richard. et. al. **The Csound Book. Perspectives in software synthesis, sound design, signal processing and programming**, London. The MIT Press, 2000.

CAESAR, Rodolfo. **Novas interfaces e a produção eletroacústica**, In: IV Simpósio Brasileiro de Computação e Música, Brasília, 1997.

CHION, Michel. **Guide des objets sonores**, Buchet/Chastel, Paris, 1995.

DEDE, Cris et al. **ScienceSpace: Virtual realities for learning complex and abstract scientific concepts**. 1995.

Disponível em:< <http://www.virtual.qmu.edu/vriaspdf.htm> >. Acesso em: 07 abr. 2006.

DODGE, Charles. et al. **Computer music, synthesis, composition and performances**, ed. Schirmer, USA, 1997.

EMMERSON, Simmon et al., **The Language of Electroacoustic Music**, The Macmillan Press, London, 1986.

FERRAZ, Silvio. **Composição com suporte tecnológico**. Relatório apresentado no Encontro Anual da Anppom. Salvador, 1999.

_____. **Composição e Ambiente de Composição**. Comunicação apresentada no "II Encontro de Música Eletroacústica". Brasília, UnB, 1997.

FURLANETE, Fábio. **Escritura sismográfica**: interação entre compositor e suporte digital. São Paulo, PUC/SP, 2000.

GARCIA, Denise. **Modelos perceptivos na música eletroacústica** (Doutorado). PUC/SP, 1998.

IAZZETTA, Fernando. **Música: Processo e Dinâmica**, AnnaBlume, 1993.
Doutorado. Programa de Comunicação e Semiótica, PUC-SP, 1997.

_____. **Material, forma e processo na Música Eletroacústica**. Apresentado no "VI" Simpósio Brasileiro de Computação e Música", Rio de Janeiro, RJ, julho de 1999.

KIENTZLE, Tim. **A programmer's guide to sound**, Addison-Wesley, USA, 1998.

MANOURY, Philippe. **La note et le son**: Écrits et entretiens 1981-1998, Paris, Editions L'Harmattan, 1998.

MATTHEWS, Max. **The Technology of Computer Music**, M.I.T. Press, Boston, 1969.

_____. **A Espacialidade na Música Eletroacústica**. In: revista ARTUnesp, vl. 11, p. 56- 61, 1995.

_____. (Org) **Música Eletroacústica – História e Estéticas**, São Paulo, Edusp, 1996.

_____. **Atualidade estética da música eletroacústica**, ed. Unesp, São Paulo, 1998.

MIRANDA, Eduardo. **Computer Synthesis for the Electronic Musician**. Oxford, Focal Press, 1998.

POHLMANN, Ken. **Principles of digital audio**, McGraw Hill, N.Y., USA, 2000.

POPE, Stephen. **Sound and music processing in SuperCollider**. Ed. Univ. of California, Santa Barbara, USA, 1997.

PUCKETE, Miller. **Max at seventeen**, Cambridge, In: Computer Music Journal, Vol. 26, no. 4, MA: MIT Press, p. 31-43, 2002.

ROADS, Curtis. **The Computer Music Tutorial**. Cambridge, MA, The MIT Press, 1996.

SCHAEFFER, Pierre. **Tratado dos Objetos Musicais**, Brasília, ed. UniB, 1993.

_____. **Traité des objets sonores**, Paris, Seul ed. 1966.

_____. **Solfège de l'Objet Sonore**. Ed. du Seuil, Paris, 1967.

WINKLER, Todd. **Composing Interactive Music**: Techniques and Ideas Using
Max, Cambridge, The MIT press, Massachussets, 1998.