Elementos e idéias para uma metodologia de estruturacão de eventos temporais em Pure data

*Cristiano Figueiró**[1]
*Pedro Kroger **[2]

ABSTRACT: The interactive, real-time eletroacoustic music imposes a series of challenges to the composer when demands careful in the timbric interaction, in the events control and in the flow of data on the same project. The present work is looking for particular aspects on interactive, real-time music creation using the software Pure-data (Pd). Will be discussed topics like the caracteristics of time control and manipulation of the form as also clasical techniques of sound synthesis and sampling on Pd.

KEYWORDS: Pure data; interactive music; composition.

RESUMO: A música eletroacústica interativa em tempo-real, prevê uma série de desafios ao compositor ao exigir o cuidado na interação tímbrica, no controle dos eventos e forma e no fluxo de dados no mesmo projeto. O presente trabalho procura apresentar alguns aspectos particulares da criação de música interativa em tempo-real usando o software Pure-data (Pd). Serão abordados temas como as características de controle temporal e manipulação da forma como também técnicas clássicas de síntese sonora e manipulação de samples no Pd.

PALAVRAS-CHAVE: Pure data; música interativa; composição.

Introdução

O presente artigo visa descrever a fase introdutória do projeto de doutorado do primeiro autor que trata da questão da interação em tempo-real entre músicos e computadores através de um sistema computacional escrito no Pure data (Pd). A música eletroacústica interativa em tempo-real prevê uma série de desafios ao compositor ao exigir o cuidado na interação tímbrica, no controle dos eventos e forma e no fluxo de dados no mesmo projeto. O presente trabalho procura

[1] Mestre em Música pela UFG, doutorando em música pela UFBA, bolsista Cnpq (figocris@gmail.com).
[2] Doutor em Música UFBA, professor/pesquisador UFBA (pedro.kroger@gmail.com).

apresentar alguns aspectos particulares da criação de música interativa em tempo-real usando o Pd.

O Pd é um ambiente de programação gráfico para áudio e imagens orientado a objetos. A programação é escrita em *patches* visuais onde os objetos e seus controles são conectados via caixas com entradas e saídas através de cabos. O Pd pertence a família de programas oriundos do Max que usam a metáfora do fluxograma tradicional de síntese sonora como gramática da criação de *patches*. Apesar de ser uma linguagem de alto nível o Pd aceita alguns comandos textuais encontrados em linguagens de programação como expressões condicionais, comparações, dentre outras. Ele é um software livre e multi-plataforma, além de contar com uma extensa comunidade de desenvolvedores.

Esse artigo tem como objetivo fornecer informações para o desenvolvimento de uma composição escrita integralmente em Pd onde todos os eventos sonoros sejam determinados pela programação do próprio *patch* e não dependam de interação com nenhuma fonte externa como teclado, mouse, entrada de áudio em tempo-real e controle Midi externo. Devido ao fato do Pd não ter uma sequência temporal explícita como outros programas (por exemplo, Csound [Boulanger, 2000]) torna-se necessária uma clareza da organização do fluxo temporal de dados. O principal ponto desse artigo é descrever possibilidades para uma metodologia de organização temporal dos eventos.

Os músicos tem sua formação baseada na notação musical tradicional que estabelece uma cadeia de binômios de parâmetros de estruturação sonora [Zampronha, 2000] como altura versus ataque, duração versus articulação ou métrica versus expressão. O sistema da notação tradicional tem uma estrutura que possui uma sequencialidade temporal dos eventos explícita. A adaptação dos músicos à ambientes de composição e design sonoro como Csound tende a ser natural pela sequencialidade temporal explícita nos dois sistemas (a partitura tradicional e a partitura do Csound). No Csound os eventos temporais são descritos em um arquivo de texto puro chamado “partitura”. No exemplo abaixo o instrumento 1 (i1) toca três notas sequencialmente com frequências de 220, 440 e 660 hertz respectivamente. Todas as notas duram 1 segundo e começam nos tempos 0, 1 e 2 respectivamente.

```
i1 0 1 220
i1 1 1 440
i1 2 1 660
```

Programas como Max e Pd são convidativos por permitirem uma programação intuitiva similar a um fluxograma. Por outro lado, o paradigma do Pd é muito calcado na performance em tempo-real o que pode gerar dificuldades de organização dos dados composicionais como notas, acordes, eventos e

mudanças de parâmetros de síntese devido a falta de uma sequencialidade temporal de eventos explícita.

Programadores também são acostumados com linguagens de programação de uso geral como C, Lisp ou Pascal que tem sua gramática baseada na sequencialidade das ações. A vantagem de usar uma ferramenta de síntese embutida em uma linguagem é que se pode ter uma flexibilidade muito maior na criação de relações entre os elementos devido a possibilidade de criar novas abstrações. No exemplo abaixo podemos ver a definição do instrumento simp que implementa um oscilador (osc) com envelope (make-env).

```
(definstrument simp (start dur frequency amp &optional
                (amp-env '(0 0 .5 1.0 1.0 0)))
  (multiple-value-bind (beg end) (times->samples start dur)
    (let ((osc (make-oscil :frequency frequency))
          (amp-env (make-env :envelope amp-env :scaler amp
                             :dur dur)))
      (run
       (loop for i from beg below end do
          (outa i (* (env amp-env) (oscil osc))))))))
```

Esse instrumento pode ser utilizado de uma maneira semelhante ao exemplo do Csound anterior.

```
(with-sound ()
  (simp 0 1 440 1)
  (simp 1 1 660 1))
```

Uma possível implementação do mesmo instrumento (simp) no Pd pode ser vista na figura 1, onde duas notas com frequências de 440 e 660 Hz respectivamente são tocadas. A primeira caixa de mensagem tem o argumento 440 que se refere a frequência da primeira nota. Quando essa caixa é acionada ouvimos a primeira nota e ao mesmo tempo ela aciona o objeto delay com o parâmetro de 1000 ms. Esse objeto agenda no intervalo de 1 segundo a ação da próxima caixa de mensagem com parâmetro de 660 Hz. Para tocar apenas duas notas foram necessárias três caixas (duas de mensagem e uma de objeto) além de cinco ligações. É fácil observar que a complexidade vai aumentar proporcionalmente ao número de notas utilizadas.

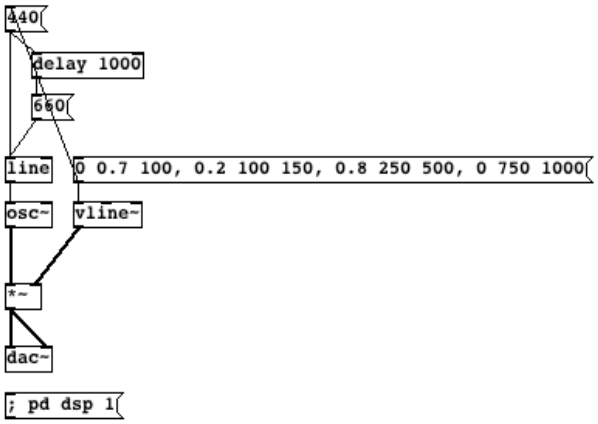

Figura 1: 2 notas num oscilador simples

O fato do Pd ser uma linguagem gráfica possibilita uma facilidade de descrição do processo de síntese sonora pelo fato de que o próprio patch é semelhante ao fluxograma tradicional de representação da síntese. A estrutura de um fluxograma possui uma sequencialidade implícita (da saída final até os parâmetros de cada gerador), mas não explicita como vão ser manipulados os parâmetros nem qual será a duração da música.

Os ambientes Max e Pd são construídos sob o paradigma do envio da mensagem que não necessariamente possibilita que a linguagem seja adequada para o armazenamento e a recuperação de dados. A abordagem do Max e Pd quanto aos dados é ao mesmo tempo simples e evasiva: objetos especiais de armazenagem de dados como table, qlist, dentre outros são disponibilizados. Os dados são essencialmente colocados dentro de objetos contêiner, e para cada tipo de objeto-conteiner uma abordagem particular é colocada para sua armazenagem, edição, interface e comunicação com o resto do patch.

A recuperação de dados (a grande maioria das transações de bases de dados) é a pior qualidade do Pd porque mensagens não tem valores de retorno. Por exemplo, uma caixa de número manipulada com o mouse não retorna os dados da manipulação, que devem ser recuperados com outra sequência de objetos. Os dados recuperados devem ser mandados como uma mensagem separada de retorno. Isso leva muitos programadores a achar soluções diferentes que facilite a interação do *patch* com o nível composicional.

A idéia original por trás da criação do Pd foi remover a barreira entre a computação dirigida por eventos em tempo-real (como no estilo do Max de passagem de mensagens) e dos dados (como em pontos num gráfico ou notas numa partitura). Em Pd caixas de objetos e estruturas de dados podem facilmente coexistir em uma mesma janela. Essa “promiscuidade”, no entanto,

não acaba deixando os objetos funcionais e os dados intimamente conectados. De fato, no design presente, o acesso aos dados tem que ser feito através de uma sequência de objetos como acessórios .

Em relação a essa divisão do aspecto performático e composicional do Pd, Miller Puckette explica que

> "...in its most succinct form, the problem is that, while we have good paradigms for describing processes (such as in the Max or Pd programs as they stand today), and while much work has been done on representations of musical data (ranging from searchable databases of sound to Patchwork and OpenMusic, and including Pd's unfinished data editor), we lack a fluid mechanism for the two worlds to interoperate." [Puckette, 2004]

Dentro de uma proposta de trabalho de composição de música interativa, o objetivo musical deve ser o mais claro possível de maneira que o *patch* emerja naturalmente dos problemas musicais ao invés de atrapalhar os problemas composicionais.

Abordando o assunto a partir de uma visão didática e metodológica, deve-se compreender a distinção clara entre os aspectos performáticos e composicionais do ambiente e dar conta de fornecer elementos para o estabelecimento de uma metodologia apropriada de controle dos eventos temporais. A proposta desse artigo é a de abordar o uso prático e algumas possíveis implementações em Pd que apontem para a criação de composições interativas mistas de maior porte. O objetivo é dispor de elementos que possam apontar para uma possível poética composicional. Serão abordados temas como técnicas clássicas de síntese sonora e manipulação de samples e também aspectos como controle temporal e macro-estruturas no Pd.

1 . Técnicas de síntese e manipulação

1.1. Síntese Aditiva

A síntese aditiva pode ser entendida como o reverso da análise de Fourier. A figura 2 mostra um exemplo típico de síntese aditiva em Pd onde os osciladores são somados em pares e tem a amplitude final controlada pelo objeto line. Nesse caso cada oscilador poderia ter seus parâmetros de frequência (no caso fixos como argumentos numéricos escritos dentro das caixas de oscil) e amplitude (objetos line, ou outro gerador de envelopes) independentes uns dos outros.

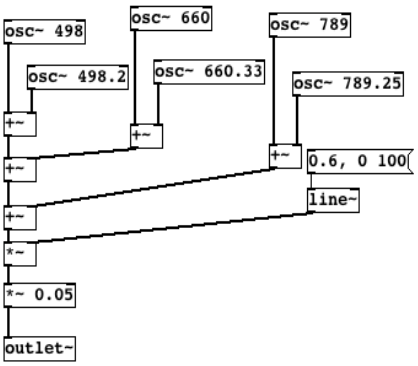

Figura 2: Síntese aditiva

1.2 Frequência e Amplitude modulada

Frequência e amplitude modulada são algumas das formas mais ricas de possibilidades tímbricas, onde a frequência, ou a amplitude respectivamente de um sinal de áudio é modulada por outro sinal de áudio. Na figura 3, abaixo os parâmetros da síntese são controlados por caixas de números e slides. No caso, uma possível interação em tempo-real com um instrumentista poderia prever o endereçamento dos valores de frequência e amplitude do sinal de áudio de entrada para agendar respostas sonoras variando os valores dos parâmetros baseados nos dados do áudio de entrada.

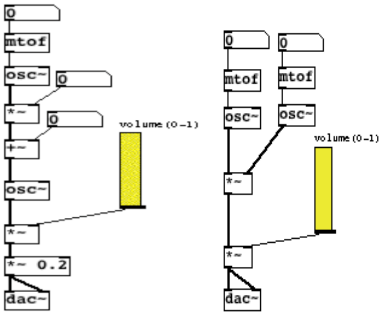

Figura 3: frequência modulada e amplitude modulada

1.3 Manipulação de arquivos de áudio

Sampleamento é uma técnica simples de ser realizada no Pd. O endereçamento de samples externos é feito apenas salvando os samples no mesmo diretório em que o *patch* que os usa está salvo. Na figura 4 o objeto

soundfiler faz a leitura de samples a partir da mensagem read, o sample no caso é endereçado para um array, onde é especificado o tamanho do arquivo que vai ser lido. Os samples são manipulados por três módulos. O primeiro realiza um scratch (surfa pelo sample) através de um line que tem seu parâmetro controlado em tempo real por uma caixa de número (no caso é uma das variáveis dessa abstração e está conectada a um line, controlado por diferentes valores de metro). O segundo módulo realiza um loop com tabplay que é um objeto que manda um bang quando acaba de tocar o sample especificado. O terceiro módulo é um loop com tabread4 controlado por phasor que manda um sinal contínuo de áudio podendo transpôr a altura do sample.

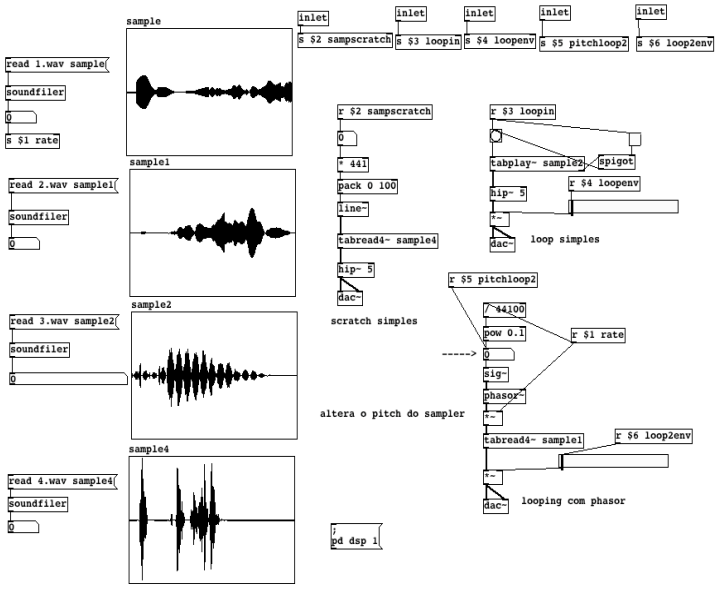

Figura 4: arquivos e 3 módulos de controle dos samples.

2. Fluxo de dados

2.1 Objetos de controle temporal

Uma das partes mais importantes de um projeto composicional com Pd é o controle do fluxo de dados, pelo fato de que o programa não tem uma sintaxe uniforme de descrição de eventos, e cada objeto específico vai ter uma gramática própria de controle de dados. Na figura 5 vemos um gerador osc sendo modulado por um gerador de envelopes de sinal de áudio vline. Esse gerador de envelopes agenda segmentos lineares e possui uma sintaxe que aceita trios de números, sendo respectivamente: amplitude (objetivo de chegada do segmento), tempo inicial (tempo de começo do segmento) e tempo de duração do segmento.

A recuperação dos dados da performance de vline é feita pelo objeto snapshot que transforma dados de sinal de áudio em números. O objeto snapshot por sua vez é acionado pelo objeto metro que é um controlador temporal que manda bangs (mensagem: faça) regulares no andamento especificado pelo seu argumento descrito em milisegundos. O seja, vline vai envelopar a amplitude de osc enquanto snapshot vai fazer leituras do fluxo de áudio numa frequência de 10 vezes por segundo.

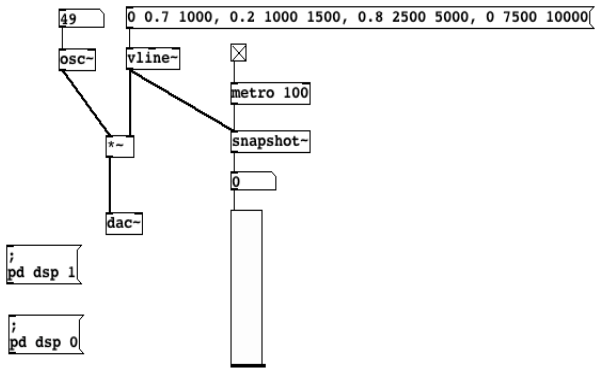

Figura 5: vline, metro e snapshot

2.2 Abstrações e subpatchs

Na figura 6 está exposta uma abstração que tem a função de mandar um bang no tempo pré-estabelecido (argumento de delay). Essa abstração está sendo usada na peça para "desligar" os módulos geradores no tempo especificado. Abstrações são patchs que assumem funções de objetos externos e podem ser usados por vários patchs diferentes ao mesmo tempo. Além disso possibilitam um certo controle gráfico, definido pelo programador. No caso, vemos que no *patch* temos um bang e uma "caixa de número" contidos num retângulo, consequentemente quando essa abstração for chamada num *patch* vai assumir a forma da figura 7.

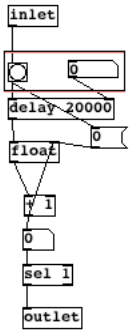

Figura 6: abstração delay: conteúdo

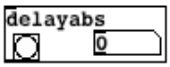

Figura 7: abstração delay: aparência externa

3. Compondo com Pd

Uma vez tendo prontos os geradores sonoros e os controladores dos dados desses geradores - em formatos de abstrações e/ou sub-patchs, podemos pensar em controlar a macro-forma da composição. Na figura 8 está exposto o controle da macro-forma da peça. Essa maneira visual de controlar a forma é mais familiar aos músicos pelo fato de termos uma linha do tempo mostrando a ordem de acontecimento dos eventos. O que permite uma maior controle de justaposição e superposição dos macro-eventos. A abstração "contador", cria um novo número no andamento específico e o objeto select cria pontos temporais de ataque que são endereçados a diferentes geradores sonoros.

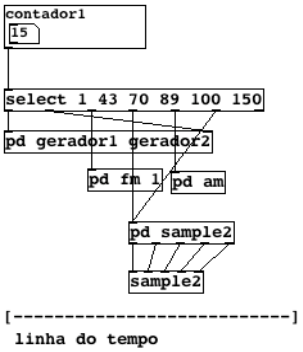

Figura 8: forma geral da peça

A figura 9 mostra o *patch* encapsulado pd gerador1 gerador2, que apareceu na figura 8 e tem seu gerador sonoro baseado em síntese aditiva. O controle é feito por uma combinação dos objetos line e metro, onde o line controla o parâmetro de tempo de metro, causando linhas de acelerando ou desacelerando. O momento de "desligamento" de cada gerador é agendado pela abstração delayabs.

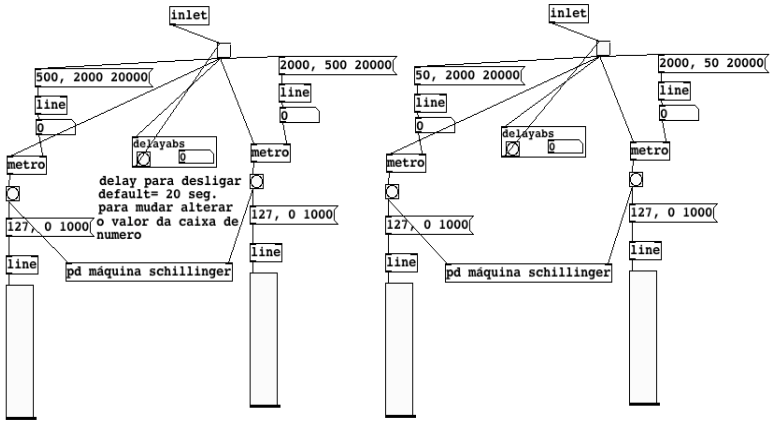

Figure 9: gerador1 gerador 2

Considerações finais

Esse artigo mostrou a necessidade do estabelecimento de uma boa metodologia de controle dos eventos temporais na programação com Pd. Foram mostrados diferentes objetos de controle temporal com sintaxes próprias e foram descritas algumas possibilidades de uso combinado desses objetos dentro de um projeto composicional.

A composição em Pd tem como característica a facilidade do controle de síntese, porém exige um tratamento especial na metodologia empregada no controle de eventos temporais. Segundo Puckette "the Pd patch looks more complex than the C code. One possible reason for the complexity is the dificulty of sequencing actions in Pd patches, which lack the natural sequentiality of a text programming language like C." [Puckette, 2004] O fato de cada objeto de controle temporal apresentar uma sintaxe diferente pode ser antes um convite a criatividade composicional do que um problema estrutural da música resultante.

Referências bibliográficas

Boulanger, R. (Org). *The Csound Book*. Massachussets: MIT Press, 2000.

Farnell, A. *Composition in puredata 1*, 2007. Disponível em: <http://www.obiwannabe.co.uk/html/music/musictuts/Composition-001/Composition-001.html>. Acessado em 25 de maio de 2007.

Puckette, M. (2004). *A divide between 'compositional' and 'performative' aspects of pd*, 2004. Disponível em : <http://crca.ucsd.edu/~msp/Publications/graz-reprint.pdf>. Acessado em 20 de maio de 2007.

Roads, C. *The Computer Music Tutorial*., Massachussets: MIT Press, 1996

Winkler, T. *Composing Interactive Music - Techniques and ideas using Max*. Massachussets: MIT Press, 1993.

Zampronha, E. *Notação, Representação e Composição - Um novo paradigma da escritura musical*. São Paulo: Annablume/Fapesp, 2000.