Programação textual em linguagem visual para computação musical

MODALIDADE: PÔSTER

SUBÁREA: MÚSICA E INTERFACES

Lucas de Oliveira da Silva
Laboratório de Acessibilidade (LAB) da UNICAMP – e-mail: factoryfiftylucas@gmail.com

Gabriel Mary Dixon Henrique
Laboratório de Acessibilidade (LAB) da UNICAMP – e-mail: gabrielzello@gmail.com

Antonio Fernando da Cunha Penteado
Departamento de Música, Instituto de Artes (IA) da UNICAMP – e-mail: nando@nandopenteado.com

Vilson Zattera
LAB e IA da UNICAMP – e-mail: vilson.zattera@gmail.com

José Fornari
Núcleo Interdisciplinar de Comunicação Sonora (NICS) da UNICAMP – e-mail: tutifornari@gmail.com

Resumo: Este trabalho apresenta uma pesquisa em andamento que trata do desenvolvimento de um método de programação textual para o ambiente visual de programação de performances de música computacional; o Puredata (Pd). Este método é particularmente útil para o músico deficiente visual que, dessa forma, passa a ter total acesso para a criação de composições e performances de música computacional e arte multimodal através do Pd.
Palavras-chave: Computação Musical. Puredata. Acessibilidade.

Textual programming of visual language of computing music
Abstract: This paper presents an ongoing research that deals with the development of a textual programming method for Puredata (Pd), the visual programming language for computer music performances. This method is particularly useful for the visually impaired musician that can now have full access to the creation of electronic compositions, computer music performances and art multimodal using Pd.
Keywords: Computing Music. Puredata. Accessibility.

1. Introdução

Linguagens de programação visual são muitas vezes utilizadas na programação de processos em tempo-real. A música é uma arte fortemente atrelada ao domínio do tempo, não apenas em termos do sentido do seu deslocamento, como também com relação ao seu andamento, à sua percepção e à contextualização do presente aparente (*specious present*); a janela do tempo que, apesar de acusticamente fazer parte do passado, cognitivamente compõe o que sentimos como o "presente" (o momento atual) e intuitivamente separamos do passado óbvio (JAMES, 1893). Como processo, a música ocorre em tempo-real. A notação musical, apesar de estruturada em tempo diferenciado, apenas se torna música no momento da performance. Com os avanços tecnológicos e computacionais, os computadores pessoais (PC) passaram a ser utilizados como ferramentas de criação musical e geração sonora. O barateamento dos PCs possibilitou a criação e acesso à linguagens de programação

especificamente desenvolvidas para a composição musical e a arte sonora. Uma das primeiras linguagens de programação musical é o "Csound" (http://csound.github.io), inicialmente desenvolvido nos anos 80s por Barry Vercoe, no *Massachusetts Institute of Technology* (MIT) descendendo do programa de síntese sonora "MusicN", de Max Mathews. Nos anos 90s surgiu o *Pure Data* (Pd); um novo ambiente computacional para música e artes sonoras em tempo-real. Pd é uma linguagem de programação visual, desenvolvida por Miller Puckette, que é especializada em criar performances de computação musical e intalações multimodais. Pd é uma ferramenta de código aberto e pertence à família de linguagens "*patcher*", tais como: Max (Max/FTS, ISPW Max, Max/MSP, jMax e DesireData. Estas são linguagens de programação em fluxo de dados, onde blocos (mensagens, objetos, átomos) são conectados por segmentos de retas que conectam as saídas (*outlets*) de blocos às entradas (*inlets*) de outros blocos. Estas conexões representam a passagem de dados de controle e mídia (PUCKETTE, 1988).

2. Linguagem textual de programação visual

Um programa em Pd é chamado de *patch* e o formato deste arquivo é textual, mas possui extensão ".pd" (ex: teste.pd). Quando um arquivo com extensão ".pd" é aberto num editor de textos, pode-se observar que existe uma linguagem de programação textual por trás da programação visual (de fluxo de dados) do Pd. Esta corresponde univocamente (do textual para o visual) a todos os elementos do patch (ex: os blocos e suas conexões). O formato de linguagem textual do Pd é "oficialmente não-oficial", ou seja, não existe documentação oficial sobre a linguagem textual que representa a linguagem visual do ambiente Pd. No entanto, existem diversos voluntários e desenvolvedores que mapearam este formato de arquivo e documentaram suas semântica e sintaxe[1]. A figura 1 mostra um patch simples do Pd e o seu arquivo textual correspondente.

Nesta figura observa-se a correspondência entre o *patch* – a programação visual, na forma de um diagrama de fluxo de dados – (à esquerda) e a linguagem textual (à direita). A primeira linha desta (*canvas*) define o tamanho e a posição da janela deste patch. O primeiro bloco (*message* ou *msg*) possui um argumento numérico (o valor 440). Ao ser clicado, este transmite este valor para a caixa de baixo (*floatatom*) que o transmitirá para a caixa de baixo (*osc~*, que gera um sinal senoidal contínuo) onde este valor irá representar a frequência desta senóide. Este objeto gera um sinal senoidal de áudio que é transmitido (por conexões com linhas mais grossas, que representam "dados de áudio") para a última caixa (*dac~*, que

representa os 2 canais da saída de áudio). As conexões entre as caixas são representadas pelos comandos *connect.*


```
#N canvas 577 134 450 300 10;
#X msg 210 37 440;
#X floatatom 210 66 5 0 0 0 - - -;
#X obj 210 94 osc~;
#X obj 201 163 dac~;
#X connect 0 0 1 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 2 0 3 1;
este é apenas um comentário;
```

Figura 1. Patch do Pd (esquerda) e o seu correspondente arquivo de texto (direita).

3. Implementação

Existem três maneiras de tornar um software acessível: 1) Reprogramação: modificação do código fonte (desde que disponível); 2) Interação através de ferramentas de acessibilidade, como é o caso dos chamados leitores de tela, e 3) Uso de *software* alternativo para criar ou editar o arquivo de armazenamento do *software* original. Optou-se neste estudo pelo terceiro item, utilizando-se um editor de texto simple (como o "bloco de notas" do Windows) para editar diretamente o arquivo salvo no formato do Pd. Este método é conhecido como Engenharia Reversa, que, de acordo com CHIKOFSKY (1990: 15), é "o processo de analisar um sistema para identificar seus componentes e como se interrelacionam, e criar representações do sistema em outra forma ou nível mais alto de abstração".

O Pd é também um *software* livre. Segundo STALLMAN (2010: 3) todo *software* livre deve garantir a seus usuários a liberdade de estudá-lo e modificá-lo, oferecendo livre acesso ao seu código fonte. Código Fonte[2]. O código fonte do Pd[3], é de livre acesso e portanto permite ser analisado. Verificou-se assim que os arquivos do Pd possuem um formato textual simples, que pode ser facilmente compreendido. Em um arquivo do Pd, as informações são salvas em registros. Cada registro é escrito em uma linha do arquivo. Cada linha inicia sempre com uma cerquilha (#) e finaliza com um ponto e vírgula (;), como pode ser observado na figura 1.

Logo após a cerquilha existe um caractere descritor do tipo do registro, o qual pode ser um entre estes três tipos de descritores: X, N ou A. Tais descritores identificam respectivamente: Elementos, Janelas e Listas de dados. Cada descritor é seguido por um elemento. No exemplo da figura 1, na quarta linha, o elemento é um objeto (obj), seguido de

parâmetros, que são: a posição horizontal do objeto na tela (210), sua posição vertical (94), um parâmetro indicando que o elemento é um oscilador senoidal de áudio (osc~).

Os objetos do Pd são interconectados, na interface gráfica por segmentos de retas, que são automaticamente desenhadas em traço fino (para dados de controle) ou grossas (para dados de áudio). Estas conexões são registradas no arquivo textual através do elemento *connect*. Na figura 1 (nas linhas 6, 7, 8 e 9) tem-se a presença destes elementos.

A sintaxe do elemento *connect* obedece à seguinte regra: a cerquilha (#), o descritor de tipo (X), o descritor de elemento (*connect*) e 4 parâmetros. Os números dos parâmetros obedecem a lógica da linguagem C, onde o primeiro número é o 0 e o n-ésimo é o N-1 (assim, o parâmetro de valor n é representado por n-1). As conexões são sempre feitas da saída (*outlet*) de um elemento para a entrada (*inlet*) de outro. Exemplificando, na figura 1, linha 8, os parâmetros são (2, 0, 3, 0). O primeiro parâmetro indica qual elemento terá seu inlet conectado. No caso, o terceiro objeto (osc~), representado pelo valor 2 (3-1=2). O segundo parâmetro representa qual outlet deste objeto será conectado. No caso, o primeiro (e único), representado por 0 (1-1=0). O terceiro parâmetro representa qual objeto terá seu inlet conectado. No caso, o quarto elemento (dac~), representado por 3 (4-1=3). Por fim, o último parâmetro representa qual inlet deste objeto receberá a conexão. No caso, o primeiro, representado por 0 (1-1=0). A linha abaixo (linha 9) cujos parâmetros são (2 0 3 1), representa praticamente a mesma conexão entre os mesmoe objetos, com a exceção do último parâmetro, que representa a conexão ao segundo inlet (2-1=1) do objeto dac~. Assim, no arquivo textual do Pd, os objetos não possuem um nome específico. Estes são apenas numerados sequencialmente (conforme aparecem ordenados na lista de definições dos elementos).

Uma outra necessidade, importante para facilitar o processo de programação e depuração do código, é a inserção de linhas de comentários. Estas funcionam como lembretes ao programador acerca do que está sendo realizado em cada parte do código. Não encontrou-se uma indicação, na documentação disponível, para a sintaxe de inserção de uma linha de comentário, na programação textual do Pd. No entanto, após diversos experimentos, o primeiro autor deste artigo conseguiu uma solução adequada; introduzir uma linha de comentário no arquivo textual bastando para isso não utilizar o cerquilha inicial e finalizar o comentário com um ponto e vírgula (como mostra a linha 10 da figura 1).

A inclusão de comentários claros e relevantes, segundo JONES (2010: 107), é parte essencial das boas práticas de programação. O formato de inserção de comentários, testado na versão 0.46 do Pd, consistindo em inserir uma linha que não comece com a cerquilha e que termine com um ponto e vírgula, gera um aviso (*warning*) na janela principal

do Pd, porém, até onde testamos, este não corrompe o arquivo, nem tampouco impede o funcionamento correto e robusto do *patch*.

4. Acessibilidade à Música Computacional

A possibilidade de programar patches de Pd de modo textual traz grandes vantagens para o usuário deficiente visual. Através de ferramentas de software que lêem arquivos de texto (os leitores de tela[4]), o usuário deficiente visual pode criar e editar arquivos de texto. Se estes forem devidamente elaborados, de acordo com as regras brevemente exemplificadas na figura anterior, um usuário deficiente visual pode programar um patch em Pd, o que, de forma tradicional (a programação visual do Pd), seria impossível de ser realizado. O objetivo deste trabalho é assim propiciar a acessibilidade e a inserção do músico deficiente visual na programação do Pd.

Um caso de utilização deste procedimento é o relato do Prof. Vilson Zattera, co-autor deste trabalho, doutor em etnomusicologia, músico (violonista) e deficiente visual. Segundo ele: "...venho buscando acessibilidade computacional para o meu aprimoramento profissional e pessoal em música. Atualmente existem alguns softwares que são razoavelmente acessíveis ou que possuem uma certa acessibilidade (alguns software livres, como, o Audacity[5]). No entanto, os softwares que oferecem alguma acessibilidade são comerciais, portanto com um alto custo. Um dos mais importantes recursos de acessibilidade para deficientes visuais são os leitores de telas, que lêem textos de uma maneira eficiente. Porém esses mesmos programas apresentam bastante limitações para a leitura de outros arquivos, como páginas da internet, tornando-se quase que ineficientes na questão da leitura de gráficos e imagens. Como a maioria dos softwares possuem uma interface gráfica, alguns softwares, como Pd, se demonstravam até então totalmente inacessíveis. O Pd, que é um ambiente computacional de programação visual, se mostrava impossível ou de extrema dificuldade para que eu conseguisse operá-lo. Começamos assim a investigar como acessar esse programa juntamente com o leitor de tela. Conseguimos abrir o menu móvel do Pd, porém nenhum dos comandos que tentamos executar se mostraram eficiente, pois, por esse método não é possível conectar os objetos do Pd. Posteriormente abrimos um patch do Pd através do editor de texto "bloco de notas". Inicialmente isso se mostrou bastante confuso e complexo, porém começamos a mapear os comandos, números e expressões que aparecem no arquivo de texto e assim a fazer uma relação com a parte visual. Isso abriu para nós, como pesquisadores (e para mim, como deficiente visual) uma possibilidade inédita de poder trabalhar com um programa como o Pd que é, em teoria, totalmente visual. Através do

mapeamento e da identificação dessas expressões, verificamos que podemos fazer toda e qualquer atividade que é possível realizar no Pd de modo visual, através do texto. Como um músico deficiente visual essa nova possibilidade de trabalhar com o Pd certamente irá ampliar o meu conhecimento dentro da música computacional, em termos de criação sonora, performance e composição musical, bem como será um grande incentivo para continuar essa pesquisa de ampliação da acessibilidade para músicos deficientes. A partir dessa pesquisa pretendemos tornar realidade a acessibilidade para músicos com deficiência visual que hoje em dia se encontra em estagio precário, se comparado aos músicos videntes".

5. Discussão

O procedimento utilizado para garantir acessibilidade ao Pd para o músico deficiente visual foi a edição textual do arquivo de programação do *patch*. A principal vantagem deste método é permitir que um músico com deficiência visual crie *patches* com o Pd, que é um software principalmente visual, porém através da edição de seus arquivos de textos correspondentes, sem a necessidade de utilizar a *interface* gráfica do editor padrão do Pd. Arquivos texto são facilmente acessíveis pelo músico deficiente visual, seja através do uso de leitores de tela (que convertem textos em fala) ou através de dispositivos conhecidos como *displays braille*[6] que são capazes de converter dinamicamente linhas de texto em caracteres braille.

No entanto, deparou-se com duas questões deste método textual de criação de *patches*. A primeira é a inexistência de um formato oficial para a inclusão de comentários no arquivo de armazenamento, em formato ".pd". A solução encontrada funcionou na versão 0.46 e em algumas anteriores, mas não se pode garantir que terá suporte em versões futuras. Não é de fato uma boa prática de programação quando um programa, ao ser executado, gera avisos (*warnings*), mesmo que estes não sejam erros que impeçam ou interfiram com o seu bom funcionamento. A segunda dificuldade foi a ausência de um modo de nomear os elementos, que são referenciados por números, o que em geral não costuma ser considerado amigável. GOODLIFE (2007: 40) afirma que um nome marca um elemento como uma entidade distinta, transformando este, de um conceito etéreo, para uma realidade bem definida[7].

A utilização de nomes amigáveis para identificar elementos da programação textual poderia ser realizada através da utilização de um software intermediário, especialmente desenvolvido para este fim. A vantagem deste método, além do uso para nomear objetos, seria também permitir a inclusão de comentários no código. No entanto, a

principal desvantagem seria a curva de aprendizagem que o músico deficiente visual teria que passar para aprender e dominar um novo software, além da possível inclusão de erros (*bugs*) no código traduzido, situação esta que qualquer novo desenvolvimento está sujeito.

6. Conclusão

Este trabalho apresenta uma pesquisa em andamento, que está sendo realizada no Laboratório de Acessibilidade (LAB) da UNICAMP por alunos do PIBIC EM, intitulado: "Ferramentas de software e hardware livre para a acessibilidade musical do deficiente visual". Este projeto visa o auxílio na utilização, desenvolvimento e apoio à utilização de ferramentas de software para músicos deficientes visuais, em processos de análise, composição e performance musical. Este artigo tratou da viabilização da acessibilidade do músico deficiente visual ao Pd, uma das mais importantes ferramentas de programação para música computacional e instalações multimodais operando em em tempo-real. Após cumprida a etapa de mapeamento da semântica e da sintaxe que compõem a linguagem textual por trás da programação visual do Pd, o próximo passo será redigir um documento, em texto acessível, para o ensino desta metodologia, e distribuí-la gratuitamente aos músicos deficientes visuais interessados em aprender a programar em Pd. Feito isto, este documento deverá passar por diversas revisões (vindas dos próprios usuários) até que este se torne suficientemente completo e coerente. Este projeto se estenderá desde a questão da utilização e implementação de ferramentas computacionais de auxílio ao acesso à música, para compositores e instrumentistas com deficiência visual, até a exploração da interatividade musical, dada pela excelência aural destes músicos, no que tange à sua destacada capacidade de localização espacial sonora, e que pode assim constituir uma vantagem da percepção sonora do deficiente visual, podendo esta ser utilizada na análise, criação e performance musical contemporânea.

Referências:

CHIKOFSKY, Elliot J.; CROSS, James H.. *Reverse engineering and design recovery*: A taxonomy. *IEEE Software,* v. 7, n. 1, p. 13-17, 1990.

GOODLIFFE, Pete. *Code craft*: the practice of writing excellent code. No Starch Press, 2007.

JAMES, W. The principles of psychology. New York: H. Holt and Company. Page 609. 1893.

JONES, Caper. *Software Engineering Best Practices:* Lessons from Successful Projects in the Top Companies. New York, NY, USA: McGraw-Hill, Inc, 2010

PUCKETTE, Miller. The patcher. In Proceedings of International Computer Music Conference. 1988.

STALLMAN, Richard; LESSIG, Lawrence. *Free software, free society :* selected essays of Richard Stallman. Boston, MA: Free Software Foundation, 2010.

[1]https://puredata.info/docs/developer/PdFileFormat (acessado em abril de 2016)

[2] Instruções escritas em uma linguagem facilmente legível por humanos e que posteriormente são convertidas para o chamado código binário que será executado por um computador

[3] https://sourceforge.net/p/pure-data/pure-data/ci/master/tree/src (acessado em abril de 2016)

[4]https://pt.wikipedia.org/wiki/Leitor_de_tela (acessado em abril de 2016)

[5]http://www.audacityteam.org (acessado em abril de 2016)

[6] https://en.wikipedia.org/wiki/Refreshable_braille_display (acessado em abril de 2016)

[7]Goodlife dedicou o capítulo 3 inteiro de sua obra "Code Craft" para chamar a atenção à importância deste fato, o título do capítulo é uma resenha de seu próprio conteúdo: "O que há em um nome - dando nomes significativos para coisas significativas"