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

Guilherme Lunhani[1] **, Flávio Luiz Schiavoni**[2]

[1]Instituto de Artes e Design – Universidade Federal de Juiz de Fora
Juiz de Fora, MG

gcravista@gmail.com

[2]Departamento de Computação – Universidade Federal de São João Del Rei
São João Del Rei, MG

fls@ufsj.edu.br

Abstract. *This panel reports the development of a sound synthesis program, Termpot, created as an interpretive experience on a historical example of Computer Music. The Web Audio API technology was used in a reinterpretation of a [Mathews and Moore 1970]'s work, GROOVE. The resulting work is still in its infancy, but its potential XXX the possibility of creation and edition of sound functions, during a web system runtime. In this way the improviser, in a web browser, immediately hear the commands that have been written.*
Keywords*: GROOVE; Web Audio API; DSP.*

Resumo. *Este painel reporta o desenvolvimento de um programa de síntese sonora, Termpot, como uma experiência interpretativa, relativa a um exemplo histórico da Computer Music. A tecnologia Web Audio API foi usada em uma reinterpretação de um trabalho de [Mathews and Moore 1970], GROOVE. O trabalho resultante ainda é incipiente, mas seu potencial tange a possibilidade de criação e edição de funções sonoras, durante o tempo de execução de um sistema web. Desta forma a(o) improvisador(a), em um navegador de internet, ouve imediatamente os comandos que foram escritos.*
Palavras-chave*: GROOVE; Web Audio API; DSP.*

1. Introdução

O processamento de sinais digitais de áudio em navegadores de rede, é sumarizado por [W3C 2012, Roberts et al. 2013, Wyse and Subramanian 2014]. [Srikumar 2013] exemplifica a utilização de *nós de áudio*, que podem ser concatenados em um grafo de DSP, como três instâncias de nós diferentes – *OscilatorNode*, *GainNode*, *DestinationNode* –, na Figura 1. Existe um outro nó, *ScriptProcessorNode* que possibilita a customização de funções sonoras. Este último foi utilizado no desenvolvimento do *Termpot*.

2. Trabalho relacionado: GROOVE

Este artigo envolve a utilização tecnologia *Web Audio*, e de seu nó *ScriptProcessorNode*, sob uma perspectiva histórica. Neste sentido, o trabalho de [Mathews and Moore 1970], GROOVE *Generated Real-time Operations On Voltage-controlled Equipment*,foi fundamental para a estruturação do *Termpot*. Um humano digita comandos em um computador, que são renderizados em som. O som, por sua vez, será

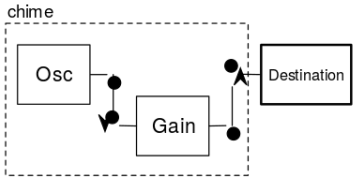

Figura 1: Estrutura básica de síntese da API webaudio, utilizando instâncias das classes *OscilatorNode*, *GainNode*, *DestinationNode*. Fonte: [Srikumar 2013].

percebido pela pessoa, que fornecerá uma nova entrada de dados (através de novos comandos, ou dispositivos, como controles manuais). O processo de síntese sonora considera, portanto, questões performáticas, como a imediaticidade. Um exemplo de música feita com o GROOVE, como apresentado na figura 2, é a obra *The Expanding Universe* de [Spiegel 1975] [1].

Figura 2: Laurie Spiegel configurando a saída analógica do GROOVE, durante a produção de *The Expanding Universe*. Fonte: [Spiegel 1975].

3. Objetivo

Descrever um programa *web* desenvolvido com base no *ScriptProcessorNode*, e estruturado segundo uma interpretação GROOVE.

4. Metodologia de desenvolvimento

Para a implementação, três tarefas foram necessárias: *1)* Customização um emulador terminal: foi utilizada a biblioteca *Ptty.js*[2] *2)* Definição de um ambiente interno customizado, no *ScriptProcessorNode*, como utilizado pelo ambiente *Wavepot*[3]. *3)* Definição de comandos deste ambiente interno: inspeção de funções, definição de

[1]Disponível em `https://www.youtube.com/watch?v=dYUZmsfm4Ww`.
[2]Disponível em `http://code.patxipierce.com/jquery-plugin/ptty/`.
[3]Disponível em `http://www.wavepot.com`.

novas funções, tocar, parar, pausar, gravar e download, criação de controles gráficos (*jQueryUI*) e gravação[4].

5. Resultados

O Termpot (ver Figura 3)[5] é um ambiente adaptado segundo uma interpretação do GROOVE, com uma base técnica do *ScriptProcessorNode*. Comandos utilizam a sintaxe da linguagem *coffeescript* [Burnham 2011][6], que possibilita a improvisação de códigos sonoros com a extensão média de uma linha (ver Código 1).

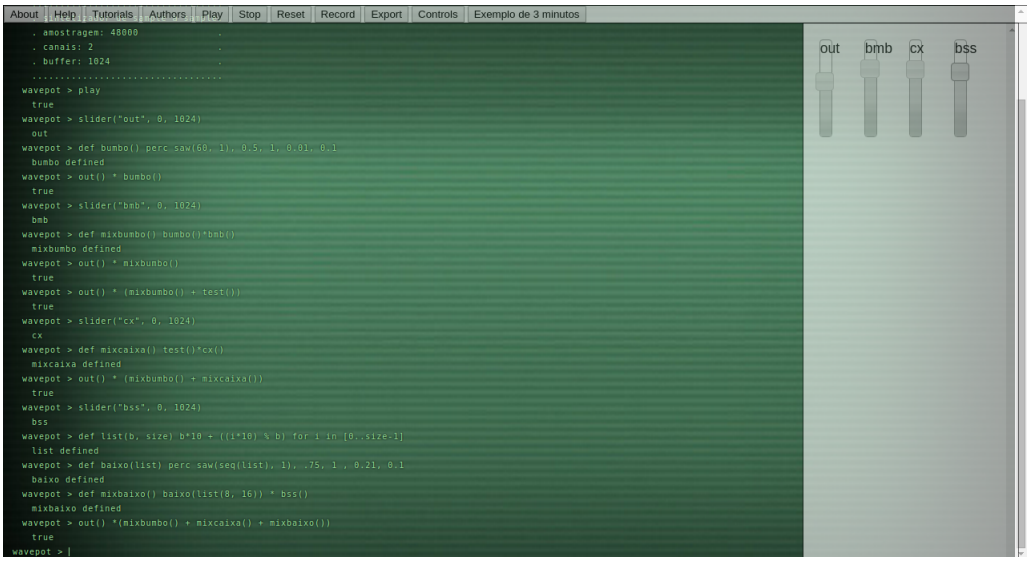

Figura 3: Aplicativo *Termpot*. Fonte: autores.

```
1  $ |                                    (1)
2  $ wavepot 1024                         (2)
3  ..................................
4  . sintetizador de sample a sample.     (3)
5  . amostragem: 44100              .
6  . canais: 2                      .
7  . buffer: 1024                   .
8  ..................................
9  wavepot > play                         (4)
10 wavepot > 0.71 * sin 440               (5)
11 true                                   (6)
```

Código 1: Mensagem de inicialização do sistema (0). Console do *Termpot* aguardando dados de entrada do improvisador (1). O improvisador inicia o ambiente *wavepot* com um buffer de 1024 amostras por ciclo de DSP (2). Informações diversas do sistema de áudio inicializado (3). O improvisador inicia o processamento de áudio (4). O improvisador define o processamento de áudio (5). O sistema informa que o processamento ocorreu sem problemas (6).

Criação e edição de funções em tempo de execução

O *Termpot* é a capaz de definir novas funções em tempo de execução. Do ponto-de-vista dos desenvolvedores, de maneira bastante rápida. Estas funções podem ser encap-

[4]Disponível em `https://github.com/mattdiamond/Recorderjs/blob/master/recorderWorker.js`.

[5]Disponível em `https://jahpd.github.io/termpot` e `https://www.github.com/jahpd/termpot`.

[6]Disponível em `http://coffeescript.org/`.

suladas em outras funções. Existem também funções de base, bem como a possibilidade de criar funções especiais, para controle manual. São apresentadas no código **??**.

```
wavepot > def AM(f1, f2) sin f1, sin(f2, 0.5)              (1)
AM defined
wavepot > inspect                                          (2)
funcoes definidas
tau  tmod  mute  stereo  sin  sin2  saw  ramp  ttri
tri  sqr   pulse noise         perc test  seq   bpm  nextevent
AM
wavepot > slider "f1", 1, 1025                             (3)
true
wavepot > slider "f2", 1, 1025
true
wavepot > slider "a", 0, 1024
true
wavepot > AM f1()*1000, f2()*1000                          (4)
true
wavepot > def AM(f1, f2, a) sin f1, sin(f2, a)             (5)
AM redefined
wavepot > AM f1()*1000, f2()*1000, a()                     (6)
true
```

Código 2: Definição de uma nova função, *AM*, que utiliza a função pré-definida *sin* (1). Inspeção das funções definidas no sistema (2). Definição de interfaces gráficas controladoras (3). Execução da função *AM*, controlados por interfaces gráficas (4). Redefinição da função *AM* (5). Reexecução da função *AM*, com um novo controle (6).

6. Conclusão

Embora em estágio incial de desenvolvimento, o *Termpot* permitiu revisita questões relativas à performance, e linguagem de programação textual simplificada para músicos. Entretanto, apontamos o seguinte problema técnico: apontamos a biblioteca jQuery como uma provável fonte de *xruns*[7]. O conceito é original do ALSA (Linux), mas é semelhante do ponto de vista perceptivo. Segundo [User:Markc 2013] um *xrun* "(...) pode ser um estouro de buffer ou de uma saturação de um buffer. Um aplicativo de áudio ou não foi rápido o suficiente para transmitir dados (...) ou não rápido o suficiente para processar os dados"[8]. Por outro lado, a criação musical baseada em funções matemáticas permite ao músico prototipar técnicas tradicionais de síntese, bem como explorar construções em cascata. Neste sentido, a ferramenta desenvolvida apresenta uma abordagem pedagógica para o ensino de música eletroacústica.

6.1. Trabalhos Futuros

i) criar um servidor; *ii)* otimizar o emulador, talvez substituindo o Ptty ou propondo melhorias; *iii)* suporte para amostras pré-gravadas.

[9].

6.2. Agradecimentos

Os autores agradecem ao Guilherme Rafael Soares por ter apresentado as biblioteca Ptty.js, aos desenvolvedores dos projetos investigados por disponibilizarem seus códigos e a FAPEMIG por subsidiar a pesquisa.

[7] *Google Chrome* 44.0.2403.89 Ubuntu 14.04.

[8] Tradução nossa de *An "xrun"can be either a buffer underrun or a buffer overrun. In both cases an audio app was either not fast enough to deliver data (...) or not fast enough to process data.*

[9] Disponível em `https://jahpd.github.io/termpot`.

Referências

Burnham, T. (2011). Coffeescript: Accelerated javascript development. *Pragmatic Bookshelf*.

Mathews, M. V. and Moore, F. (1970). GROOVE a program to compose, store, and edit functions of time. *Bell Telephones Laboratories*, page 7.

Roberts, C., Wakefield, G., and Wright, M. (2013). The web browser as synthesizer and interface. page 6.

Spiegel, L. (1975). The expanding universe: 1970s computer music from bell labs by laurie spiegel. disponível em `http://www.retiary.org/ls/expanding_universe/`. *Retiary*.

Srikumar, S. (2013). Tamming the script processor node. disponível em `http://sriku.org/blog/2013/01/30/taming-the-scriptprocessornode/`.

User:Markc (2013). Xruns: From the alsa wiki. disponível em `http://alsa.opensrc.org/Xruns`.

W3C (2012). Web audio API.

Wyse, L. and Subramanian, S. (2014). The viability of the web browser as a computer music platform. *Computer Music Journal*, 37(4):10–23.