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. 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 allows create and edit sound functions at runtime, in a web browser.*
Keywords*: GROOVE; Web Audio API; DSP.*

Resumo. *Este painel reporta o desenvolvimento de um programa de síntese sonora, Termpot. 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 possibilita criar e editar funções no tempo de execução, em um navegador web.*
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. Por exemplo, o *ScriptProcessorNode* pode ser usado para externalizar a(o) improvisador(a) suas próprias customizações de "nós virtuais" (funções matemáticas), semelhantes aos *OscilatorNode* e *GainNode* . Esta abordagem foi utilizada no desenvolvimento do *Termpot*.

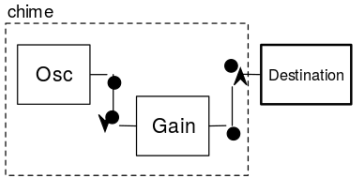

Figura 1: Estrutura básica de síntese da API webaudio, utilizando instâncias das classes *OscilatorNode*, *GainNode*, *DestinationNode*. Fonte: [Srikumar 2013].

2. Trabalho relacionado: GROOVE

Além da tecnologia *Web Audio*, este artigo envolve uma interpretação do GROOVE – *Generated Real-time Operations On Voltage-controlled Equipment*, de [Mathews and Moore 1970]. A compositora Laurie Spiegel (figura 2) sumariza a utilização do GROOVE durante a produção de *The Expanding Universe* (1975) [1]:

> Todas as músicas no GROOVE eram representadas na memória digital como funções abstratas do tempo, séries paralelas de dois pontos, cada ponto sendo um instante no tempo e um valor instantâneo. A taxa de amostragem para essas funções, na maior parte das vezes usada como controle de tensão, era cronometrada por um grande e antiquado oscilador analógico, que era normalmente fixado em 100 Hertz, cada ciclo do oscilador pulsando à frente do código, o computador lia todos dos dispositivos de entrada em tempo real, e reproduzia todas as amostras naquele ponto do tempo em cada uma das funções de tempo. [Spiegel 1975, online] [2]

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

1) Customização um emulador terminal *Ptty.js* (`http://code.patxipierce.com/jquery-plugin/ptty/`). *2)* Definição de um ambiente

[1] Disponível em `https://www.youtube.com/watch?v=dYUZmsfm4Ww`.

[2] Tradução de *All music in GROOVE was represented in digital memory as abstract functions of time, parallel series of point pairs, each point being an instant in time and an instantaneous value. The sampling rate for these functions, which would be used mostly as control voltages, was clocked by a big old-fashioned analog oscillator that was usually set to 100 Hertz, each cycle of the oscillator pulsing one run through the code, the computer reading all of the real time input devices and playing of all of the samples at that time point in each of the time functions.*

interno, baseado no ambiente *Wavepot* (`http://www.wavepot.com`). *3)* Definição de comandos deste ambiente interno: inspeção de funções, definição de novas funções, tocar, parar, pausar, gravar e download, criação de controles gráficos (*jQueryUI*) e gravação[3].

5. Resultados

O Termpot (ver Figura 3)[4] é uma customização do ambiente Wavepot. A(o) improvisador(a) define novas funções, em linguagem *coffeescript* [Burnham 2011][5]. Estas funções podem ser encapsuladas em outras funções e então executadas. Um sumário das possibilidades são apresentadas no Código **??**.

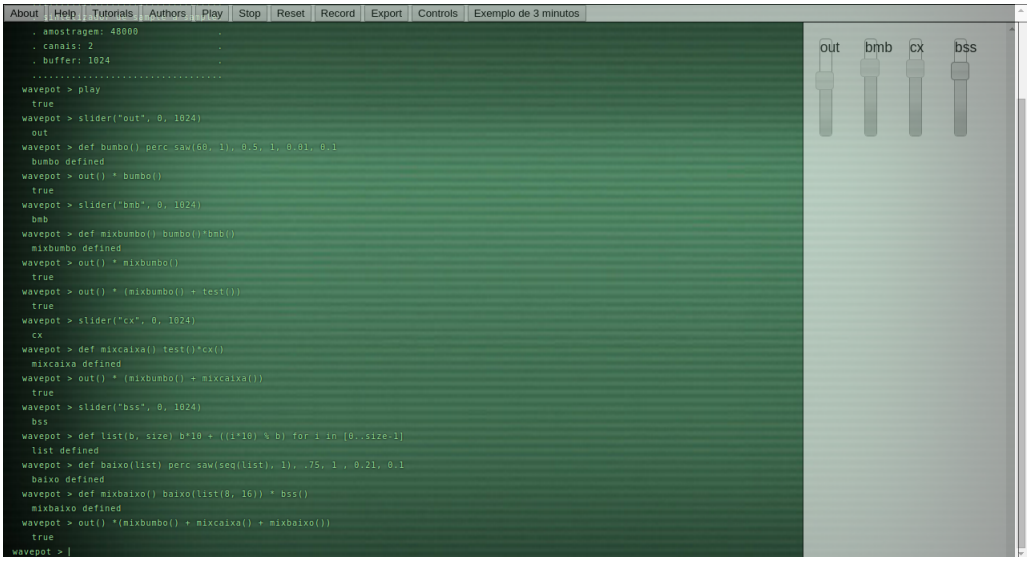

Figura 3: Aplicativo *Termpot*. Fonte: autores.

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*[6]. 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"[7]. 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.

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

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

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

[6]*Google Chrome* 44.0.2403.89 Ubuntu 14.04.

[7]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.*

```
$ |                                                    (1)
$ wavepot 1024                                         (2)
................................
. sintetizador de sample a sample.                     (3)
. amostragem: 44100             .
. canais: 2                     .
. buffer: 1024                  .
................................
wavepot > play                                         (4)
true
wavepot > def AM440(f, a) sin f1, sin(440, a)          (5)
AM defined
wavepot > inspect                                      (6)
funcoes definidas
tau  tmod  mute  stereo  sin  sin2  saw  ramp  ttri
tri  sqr   pulse noise        perc test  seq  bpm   nextevent
AM
wavepot > slider "f", 1, 1025                          (7)
true
wavepot > slider "a", 0, 1024
true
wavepot > record                                       (8)
true
wavepot > AM440 f()*1000, a()                          (9)
true
wavepot > export                                       (10)
```

Código 1: Console do *Termpot* aguardando dados de entrada do improvisador (1).*Boot* do ambiente *wavepot* com um buffer de 1024 amostras por ciclo de DSP (2). Informações diversas do sistema (3). Início do processamento de áudio (4). O improvisador define uma função *AM440* (5). O sistema informa as funções diponíveis (6). Definição de interfaces gráficas controladoras (7). Comando para gravar o processamento em um arquivo (8). Execução da função *AM440* com controles (9). Realizar o download da gravação (10)

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.

[8].

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.

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*.

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

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.