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, ttonmeister@gmail.com

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

fls@ufsj.edu.br

Abstract. *TODO ...*

Resumo. *Neste artigo, descrevemos um programa de síntese sonora, com base em uma releitura do GROOVE de Max Mathews, sob a ótica de uma tecnologia chamada* Web Audio API*. Nomeamos este programa* Termpot*.O programa utiliza a mesma tecnologia que um outro,* Wavepot*, para criar, e encapsular, diversas definições matemáticas em um único vetor de dados. O processo de criação e edição das funções se dá no tempo de execução do sistema. Neste sentido realizamos uma experiência preliminar de* livecoding*, com reflexo em um exemplo que consideramos paradigmático. Por fim, descreveremos problemas e implementações futuras.*

1. Introdução

O processamento de sinais digitais de áudio em navegadores de rede, como por exemplo o Mozzilla Firefox, Google Chrome ou Apple Safari, é sumarizado por [Roberts et al. 2013, Wyse and Subramanian 2014]. Utilizando a *Web Audio API* [W3C 2012] *nós de áudio* podem ser concatenados em um grafo de DSP. Um exemplo desta concatenação é demonstrado na Figura 1. [Srikumar 2013] apresenta três instâncias de nós diferentes (*OscilatorNode*, *GainNode*, *DestinationNode*), em sequência, para construir um simples sintetizador.

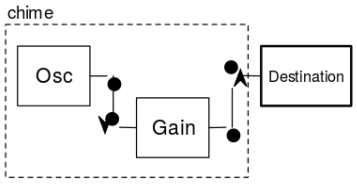

Figura 1: Estrutura de síntese da API webaudio. Fonte: [Srikumar 2013].

1.1. Script Processor Node

Um sintetizador mais complexo contêm diversos nós concatenados. Este sintetizador complexo pode ser simplificado com um outro nó, chamado *ScriptProcessorNode*. Uma instância deste nó permite o cálculo customizado de cada *sample* em um *buffer*, à semelhança de uma função matemática.

1.2. GROOVE

Este artigo envolve uma pesquisa sobre prática *livecoding* [Collins 2014, Di Prospero 2015, Di Prospero 2015].

O resultado deste artigo não é o objetivo da pesquisa, mas relaciona uma proposta histórica de Max Mathews, [Mathews and Moore 1970], GROOVE. O estudo deste trabalho de Mathews foi fundamental para o entendimento de um paradigma do *livecoding*: ao digitar comandos em um console de computador, é possível ouvir os os resultados, e modificá-los através de novos comandos, e/ou com controles externos.

Para que este modelo de paradigma fosse aplicado em um navegador de internet, recorremos à implementação do *ScriptProcessorNode*, da equipe desenvolvedora do programa *Wavepot*[1].

2. Termpot

O Termpot (ver Figura 2) é um ambiente de luteria composicional [Iazzetta 2009, Soares 2015], que utiliza o *livecoding* como paradigma de programação. É também uma adaptação do ambiente *Wavepot* (apresentado na nota-de-rodapé 2 da Seção 1) .

Comandos utilizam a sintaxe da linguagem *coffeescript*[Burnham 2011][2], que possibilita otimizações diversas, entre elas, o tempo de produção do próprio código. Neste sentido, o ambiente possibilita a improvisação de códigos com a extensão média de uma linha (ver Código 1).

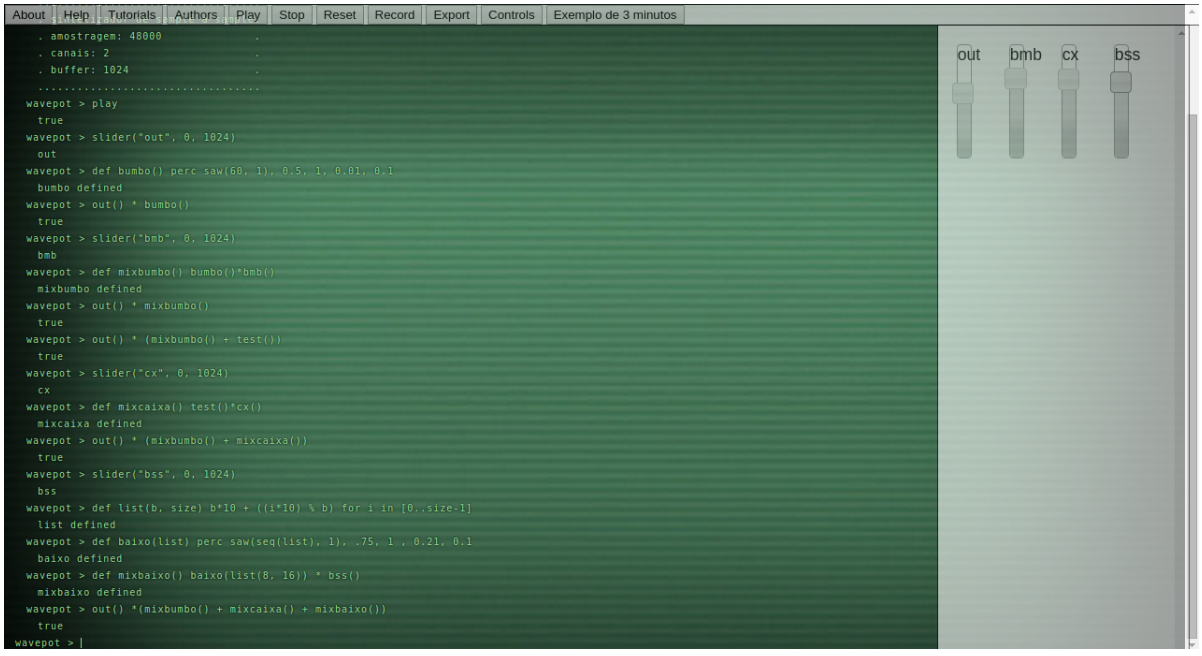

Figura 2: Aplicativo *Termpot*. Fonte: autores.

É interessante notar aqui um princípio de compilação *Just In Time* [Aycock 2003]. Antes que uma função de áudio específica seja executada, o sistema é executado, à princípio, com silêncio. Ao se definir uma função de saída, o sistema automaticamente executa a função para os alto-falantes. O processo de síntese é semelhante ao *Wavepot*.

[1]Disponível em `http://www.wavepot.com`.

[2]`http://coffeescript.org/`, acessado em 4 de novembro de 2015.

```
.....................................
. Virtual machine started at          .
. Thu Sep 03 2015 13:32:15 GMT-0300 (BRT).
. type help for instructions          .
.....................................
$\$$ |                                          (1)
$\$$ wavepot 1024                               (2)
...............................
. sintetizador de sample a sample.              (3)
. amostragem: 44100           .
. canais: 2                   .
. buffer: 1024                .
...............................
wavepot > play                                  (4)
wavepot > 0.71 * sin 440, sin(330, sin(220, 0.5) (5)
true
```

Código 1: Console do *termpot* aguardando dados de entrada do improvisador (1). O improvisador inicia o ambiente wavepot com um buffer de 1024 pontos flutuantes (2). Informações diversas do sistema de áudio (3). O improvisador inicia o processamento de áudio (4). O improvisador define o processamento de áudio (5).

2.1. Criação de funções em tempo de execução

Uma das potencialidades do *software* é a capacidade de definir novas funções em tempo de execução, do ponto-de-vista dos desenvolvedores, de maneira bastante rápida. Estas funções podem ser encapsuladas em outras novas funções. Existem também funções de base, como *noise* (Ruído branco), *sin* (senóide), *saw* (dente-de-serra), *tri* (triangular), *pulse* (pulso), *env* (envelope) e *sequenciamento* (seq)..

```
wavepot > def tresAM(f, f1, f2, a) a * sin f, doissin(f1, f2)
true
wavepot > def doissin(f1, f2) sin f1, sin(f2, 0.5)
true
wavepot > inspect tresAM
var tresAM;

tresAM = function(f, f1, f2, a) {
  return a * sin(f, doissin(f1, f2))
};
wavepot > inspect doissin
var doissin;

doissin = function(f1, f2) {
  return sin(f1, sin(f2, 0.5))
};
wavepot > tresAM 440, 330, 220, 0.71
true
```

Além da prototipação de funções, habilitamos a criação dinâmica de GUIs de controle (*sliders*, ver Código 2)

```
wavepot > slider "left", 0, 1024
true
wavepot > slider "right", 0, 1024
true
wavepot > stereo sin(440,0.5)*left(), sin(330, 0.5)*right()
true
```

Código 2: Exemplo de código do Wavepot

Uma característica singular do *Wavepot* original, é a possibilidade de separação da programação-partitura em dois arquivos, muito semelhante ao método *Instrumento-Orquestra* descrito por Max Mathews e utilizado no CSound [Mathews 1963,

Di Nunzio 2010]. Isso é possível adicionando um marcador $@module$ aos comentários iniciais de um código. Desta forma, serão reconhecidos dois arquvivos durante a performance de improvisação: $index.js$ e $test.js$. O primeiro permite elaborar os instrumentos, enquanto o segundo realiza o DSP (teste).

Já no Termpot, buscamos utilizar outro método de codificação focado em uma abordagem mais performática.

Arriscamos a comentar uma inspiração no GROOVE de [Mathews and Moore 1970, Nunzio 2010], quando este propõe a criação de novas funções em tempo de execução. Ao mesmo tempo em que utilizamos a biblioteca *Ptty.js* dando ao *Termpot* as características de um emulador de terminal no que tange a utilização de comandos em tempo de execução, como em um terminal, integrado com controles manuais. Por esta razão, acreditamos que esta ferramenta é inspirada no conceito de compor, memorizar, editar e controlar funções do tempo, algoritmicamente e manualmente [Mathews and Moore 1970].

Referências

Aycock, j. (2003). A brief history of just-in-time. *ACM Computer surveys*, pages 97–113.

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

Collins, N. (2014). Origins of live coding.

Di Nunzio, A. (2010). Genesi, sviluppo e difusione del software "MUSIC n"nella storia della composizione informatica.

Di Prospero, C. (2015). Social participation. In *ICLC2015 Proceedings*, pages 68–73.

Iazzetta, F. (2009). *Música e mediação tecnológica*. Ed. Perspectiva-Fapesp.

Mathews, M. V. (1963). The digital computer as a musical instrument. *Science*, 142(3592):553–557.

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

Nunzio, A. d. (2010). Groove.

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

Soares, G. R. (2015). Luteria composicional de algoritmos pós-tonais v1.1final.

Srikumar, S. (2013). Tamming the script processor node.

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.