Laboratórios de Informática 2014-15

Projeto 2: *Virtual Hammond*

Data de Entrega: 7 de Junho de 2015, 23:59

Versões

- v1.0 - Versão Inicial.

1 Enquadramento

Um órgão Hammond é um instrumento musical criado no século passado (cerca de 1934) para ser utilizado em igrejas, mas acabando por ser um dos órgãos elétricos com maior impacto na produção musical Rock, Jazz e Blues, estando presente em muitas músicas conhecidas. Internamente ele era composto por cilindros rotativos capazes de produzir tons praticamente puros, sendo portanto designados por osciladores. Um órgão permitia utilizar até 9 osciladores em simultâneo, cada um produzindo frequências que variam com a nota tocada pelo intérprete. O som produzido consiste na soma das 9 frequências. Em particular, um oscilador irá reproduzir a frequência indicada pela nota, sendo que os seguintes irão produzir frequências múltiplas da nota.

Os osciladores estão organizados na sequência `[1/2, 2/3, 1, 2, 3, 4, 5, 6, 8]`, pelo que se fosse tocada uma nota de 440Hz, o som produzido seria a soma de 9 componentes com frequências de 220, 293, 440, 880, 1320, 1760, 2200, 2640 e 3520. O órgão possui a noção de registos, sendo que é possível especificar a amplitude de cada componente usando valores inteiros entre 0 e 8. O valor 0 significa que o sinal do cilindro não será utilizado, enquanto que o valor 8 significa que este terá amplitude total, 4 significa que a sua amplitude será metade. Registos normalmente utilizados são 88 8800 000, 88 5324 588 ou 88 8000 008, mas pode ser utilizada qualquer outra combinação. Cada registo resulta num timbre distinto do instrumento e, por isso, variando o registo é possível obter sonoridades bastante diferentes para uma mesma música.

2 Objetivo

O objetivo deste projeto é a criação de uma aplicação Web que permita aos utilizadores fornecerem uma especificação de uma música e uma configuração de registos, gerando-se a música correspondente. A aplicação deverá guardar a informação em base de dados e no sistema de ficheiros, permitindo que as músicas sejam ouvidas repetidamente após terem sido criadas.
Esta aplicação deverá ser composta pelos componentes abaixo indicados, ou por outra estrutura funcionalmente equivalente.

Os componentes desenvolvidos, sempre que possível e relevante, deverão ser acompanhados por testes unitários e/ou funcionais.

2.1 Interface Web

Conteúdos `HTML`, `Javascript` e `CSS` que permitam interagir com o sistema, tanto em computadores pessoais como em dispositivos móveis. Esta página tem quatro funcionalidades principais: a) permitir aos utilizadores a adição de uma música, devendo para isto ser fornecida a pauta no formato RingTone Text Transfer Language (RTTTL), o registo e os efeitos a aplicar; b) a listagem das músicas existentes, apresentado a pauta, os registos e os votos; c) para uma música existente na aplicação, a criação de novas versões da mesma música mas com novos registos ou efeitos; d) a possibilidade dos utilizadores atribuírem votos positivos ou negativos a uma interpretação específica de uma música.

A versão móvel da aplicação pode ter funcionalidades reduzidas, nomeadamente apenas de leitura.

Além disto, a página deverá identificar os seus autores e possuir uma hiperligação para o repositório com o código (plataforma CodeUA).

2.2 Aplicação Principal

Irá fornecer interfaces `JSON` para o Interface Web e irá gerir a lógica de funcionamento da aplicação. Deverá permitir a listagem e adição de pautas de música, assim como a obtenção dos ficheiros gerados. Estes ficheiros consistem em ficheiros de som no formato WAV (gerados pelo processador de efeitos) e ficheiros contendo uma visualização das notas reproduzidas.

Os ficheiros com a visualização das notas deverão ser criados por este módulo e irão representar as frequências produzidos em função do tempo. Um exemplo desta imagem é a apresentada na Figura 1, tendo sido criada pelo módulo `matplotlib`.

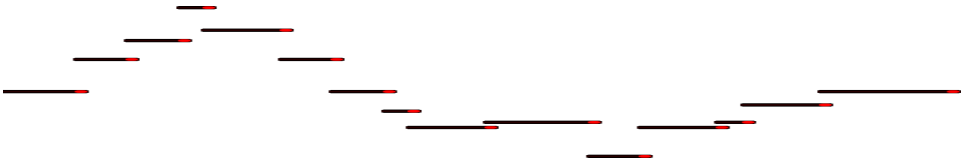

Figura 1: Visualização de notas gerada com `matplotlib`

Do ponto de vista do interface HTTP, considera-se que serão necessários os seguintes serviços, ou outros funcionalmente equivalentes:

- `/createSong?name=text&notes=text` Cria uma nova música com o nome e notas fornecidas.
- `/createInterpretation?id=number&registration=text&effects=text` Indica que se deve gerar uma nova interpretação da música com o registo e efeitos especificados.
- `/listSongs` Devolve a lista de músicas existentes.
- `/listSongFiles?id=number` Devolve a lista de ficheiros de uma música. Estes ficheiros podem ser imagens apresentando uma visão gráfica das notas da música e ficheiros WAV com a música.
- `/getNotes?id=number` Devolve a pauta de uma música.
- `/getWaveFile?id=wfileNumber` Devolve um ficheiro WAV contendo a interpretação da música.
- `/getWaveForm?id=wformNumber` Devolve uma imagem contendo uma representação gráfica das notas.

Toda a informação relativa às pautas e registos deverá estar contida em base de dados, à exceção dos ficheiros com as músicas (WAVE) e as imagens, que deverão existir no sistema de ficheiros.

2.3 Interpretador de pautas

O interpretador recebe uma pauta e deve devolver uma lista de pares `[(t1,f1), (t2,f2), ...]`, onde cada par indica a duração (em segundos) e a frequência fundamental (em Hz) de uma nota da pauta.
A pauta consiste numa sequência de carateres no formato RTTTL e é composta por 3 partes separadas pelo caráter ':'. Um exemplo de uma música será:

```
"The Simpsons:d=4,o=5,b=160:c.6,e6,f#6,8a6,g.6,e6,c6,8a,8f#,
8f#,8f#,2g,8p,8p,8f#,8f#,8f#,8g,a#.,8c6,8c6,8c6,c6"
```

A primeira parte define o nome da música, neste caso `The Simpsons`.

A segunda parte `"d=4,o=5,b=160"` define os parâmetros de referência para a música, nomeadamente:

- `d=4` define o *pulso* ou batida, que estabelece a duração de referência das notas, neste caso uma semínima (ver abaixo).
- `o=5` indica a *oitava* (ou escala) de referência das notas.
- `b=160` fixa o *andamento* em número de batidas por minuto.

Se um ou mais dos parâmetros não forem defindos, presume-se `d=4`, `o=6` e `b=63`.

A terceira parte contém as notas da música separadas por vírgulas. Um exemplo de nota é `4c#.7`. Por ordem de aparecimento, os carateres possuem o seguinte significado:

- `4` - *Valor* da nota, que indica a sua duração como fração de uma nota completa (semibreve). Os valores possíveis são: 1 (𝅝 semibreve), 2 (𝅗𝅥 mínima), 4 (♩ semínima), 8 (♪ colcheia), 16 (semicolcheia) ou 32 (fusa). O valor 1 corresponde à nota mais longa, 2 tem metade da duração, 4 dura $1/4$ e assim sucessivamente. Se não for especificado, presume-se um valor de duração igual a `d`.
- `c#` - *Tom* da nota. Os valores possíveis são, do mais grave ao mais agudo: `c, c#, d, d#, e, f, f#, g, g#, a, a#, b`, que correspondem aos 12 (semi)tons de uma oitava: Dó, Dó sustenido, Ré, ..., Lá sustenido, Si. Pode ainda ser `p` para indicar uma pausa (silêncio), ou `h` que é equivalente a `b` (Si). Este é o único elemento obrigatório da nota.
- `.` - Caso exista, um ponto indica que a duração deve ser aumentada em 50%. Por exemplo, `4c#.` deve ter a duração de $1/4 + 1/8$ de uma semibreve.
- `7` - Oitava (ou escala) da nota, que juntamente com o tom determina a *altura* absoluta da nota. Poderá ser um número entre 0 e 8, onde 4 indica a oitava média de um piano. O Lá médio (`a4`) é afinado para ter uma frequência fundamental de 440Hz, `a5` corresponde a 880Hz, `a6` corresponde a 1760Hz, etc. Se não for especificada, presume-se igual à oitava de referência `o`.

A duração real da nota depende do seu valor, da presença de ponto, mas também do pulso `d` e do andamento `b`. Por exemplo, se $d = 4$ e $b = 160$,

uma nota de valor 4 terá a duração de $\frac{60}{160}$ segundos. Já uma nota de valor 8 (colcheia) terá a duração $t = \frac{4}{8}\frac{60}{160}$ segundos. Se tiver ponto, a duração será $1.5\,t$.

A frequência fundamental da nota (em Hz) é dada por $440 \times 2^{\frac{n}{12}}$ com n a variar entre -28 e 21. Cada conjunto de 12 tons representa uma oitava.

Sugestão: Para evitar cálculos demorados, poderá usar uma Lookup Table (LUT) [1]. Assim, se inicializar adequadamente uma lista (ou dicionário) `freqs` com todas as frequências possíveis, poderá obter a frequência de uma nota do órgão fazendo simplesmente $freqs[12 \times (oitava - 4) + tom]$. Para as durações das notas poderá usar uma técnica semelhante.

2.4 Sintetizador

O sintetizador irá receber um registo do orgão e os pares (duração, frequência) do interpretador, e deve produzir a sequência de amostras que representam a forma de onda de cada nota, com uma dada resolução e frequência de amostragem.
A forma de onda é obtida pela soma das várias componentes sinusoidais de frequências múltiplas da fundamental, ponderadas de acordo com o registo. Nesta fase devem ser tomadas precauções para evitar *clipping* da onda. [2]
Se a pauta tiver 10 notas, este módulo deverá devolver uma lista com 10 dicionários contendo a frequência e as amostras da nota, no formato `{'freq': 440, 'samples': [2332, 1223, ...]}`.

2.5 Processador de efeitos

Este sistema aceita uma lista de sons já gerados (vindo do sintetizador) e uma lista de efeitos. Cada som da lista deverá indicar qual a frequência da nota que o gerou e os dados que o compõem, tal como definido na secção anterior.

Este componente aplicará os efeitos indicados de forma individual a cada som, ou de forma global à sequência final, gerando no final um ficheiro WAVE para armazenamento no sistema de ficheiros.

Juntamente com este ficheiro é já fornecido uma estrutura básica e funcional deste componente, sem a aplicação de qualquer efeito.

- `Echo`: Soma do sinal atual atenuado a um sinal futuro (ex. 0.1 segundos depois). A fórmula genérica será $v[i + d] += a \times v[i]$, em que v é uma

[1] `http://en.wikipedia.org/wiki/Lookup_table`
[2] `http://en.wikipedia.org/wiki/Clipping_(audio)`

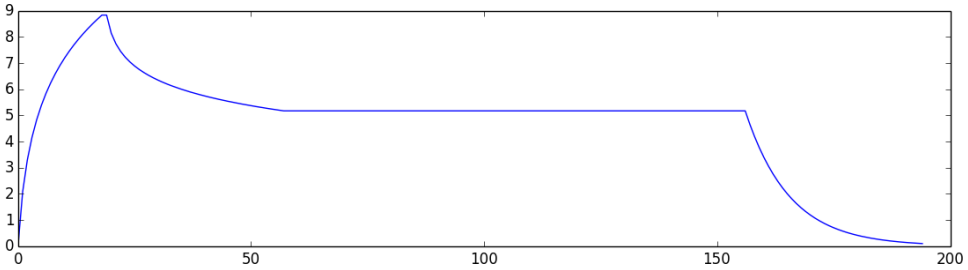

Figura 2: Envelope típico de uma nota

lista de valores de amplitude, a a atenuação da cópia, i o número do valor atual e d o atraso.

- `Tremolo`: Variação periódica, de pequena magnitude ($<5\%$), da amplitude do sinal. A fórmula genérica será $v[i] += a \times sin(2 \times \pi \times f_t \times \frac{i}{r}) \times v[i]$, em que a é a magnitude do efeito, f_t a frequência do efeito, r o número de amostras por segundo, v uma lista de valores de amplitude e i o número de um dado valor.

- `Distorção` Amplificação exagerada com ocorrência frequente de *clipping*. A fórmula genérica será $v[i] = v[i]^n$ sendo que depois os valores devem ser limitados ao intervalo `-32768` e `32767`. O valor n define a magnitude do efeito.

- `Percursão` No início da música, ou depois de uma pausa, adiciona uma harmónica (múltiplo da frequência), com uma amplitude que decai ao longo do tempo, até se anular.

- `Chorus` Adiciona um outro tom numa frequência ligeiramente superior à frequência principal presente. Consideram-se variações menores a $+50$Hz, que podem ser constantes ou variar igualmente (ex, entre 30Hz e 50Hz).

- texttttEnvelope Manipula a amplitude de cada nova nota de acordo com uma forma específica, simulando características do instrumento. Nomeadamente uma fase crescente de amplitude seguida de um pequeno decaimento, correspondendo à ativação da nota. Uma fase em que a amplitude se mantém praticamente constante e finalmente um decaimento correspondendo à desativação da nota. Um exemplo será o apresentado na Figura 2.

3 Regras

O projeto deve ser realizado por um grupo de **4 alunos**. Em casos excecionais serão permitidos grupos com dimensão diferente. A avaliação terá em conta

a dimensão do grupo.

Os alunos deverão criar um projeto na plataforma CodeUA, com o identificador no formato `labi2015-proj2-g#`, em que `#` representa um número inteiro positivo. Para escolher o número os alunos devem tentar criar um projeto com o identificador `labi2015-proj2-1`, incrementando o valor até que a criação seja autorizada pela plataforma. Devem ser incluídos os 4 docentes da disciplina: `jpbarraca@ua.pt`, `jmr@ua.pt`, `dgomes@ua.pt` e `mario.antunes@ua.pt`.

Assume-se que todos os conteúdos entregues são da autoria exclusiva dos alunos membros do grupo. A incorporação de elementos ou ideias externas ao grupo, sem a devida indicação, constitui plágio, levando à não avaliação do trabalho.

A entrega final consiste no envio de um relatório através da página da disciplina e do código presente na plataforma Code.UA.

A avaliação irá focar-se nas funcionalidades da aplicação criada e no seu processo de desenvolvimento. A aplicação será avaliada pelo cumprimento dos objetivos enunciados, a sua estrutura e implementação, e aparência

O desenvolvimento será avaliado pela utilização da plataforma Code.UA, nomeadamente as funcionalidades de atribuição de tarefas e sistema de controlo de versões. Devido ao facto deste projeto ser realizado por um grupo de 4 alunos é vital a distribuição efetiva de tarefas, sendo que a nota final poderá ser ponderada pelas tarefas implementadas por cada aluno. Se for devidamente identificado quem é o autor, ou contribuidores, dos componentes e testes, será possível avaliar de forma independente os vários membros do grupo.

Caso se verifiquem discrepâncias relevantes entre a participação dos elementos do grupo, a nota atribuída a cada elemento do grupo não será uniforme.

Irá existir uma penalização de 2 valores por cada 24 horas de atraso na entrega, aplicada em múltiplos de 1 minuto.

Considera-se que o projeto pode ser entregue sem que se implemente o processador de efeitos. Neste caso a nota estará limitada a 17 valores (por outras palavras, o componente vale 3 valores). Para notas superiores a 17 será necessária a implementação deste componente. No entanto, recomenda-se que os alunos apenas iniciem a sua implementação após a implementação correta dos restantes módulos.

4 Datas importantes

- **8 de Maio** - indicação dos membros do grupo e número do projeto.
- **17 de Maio** - entrega de um documento descrevendo que módulos serão criados, como interagem e quem ficará responsável pela sua implementação. Deverá ser indicada a estrutura das mensagens (p.ex, JSON) trocadas, métodos e parâmetros. O responsável poderá ser alterado e vários alunos poderão contribuir para o mesmo componente, sendo que esta informação tem de estar contida no relatório final. Esta entrega terá um peso de 2 valores da nota final do projeto.
- **7 de Junho** - entrega final do código desenvolvido e de um relatório que enquadre, descreva e avalie a solução criada. A solução criada terá um peso de 16 valores, enquanto o relatório terá um peso de 2 valores.

5 Ligações de Interesse

- Hammond Organ, http://en.wikipedia.org/wiki/Hammond_organ
- RTTTL Tutorial, http://www.mobilefish.com/tutorials/rtttl/rtttl_quickguide_specification.html
- Músicas RTTTL, http://www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/
- MatplotLib Tutorial, http://cs.smith.edu/dftwiki/index.php/MatPlotLib_Tutorial_1
- MatplotLib, http://matplotlib.org
- Synthesizing Tonewheel Organs, http://www.soundonsound.com/sos/nov03/articles/synthsecrets.htm