MAC337/5900 – Computação Musical – EP1

Prof. Fabio Kon

Assistente de Ensino: Leandro Ferrari Thomaz

Data de entrega: **9/9/2007**

Instruções: Os EPs devem ser feitos individualmente ou em dupla.

Sintetizador Aditivo Polifônico

Introdução

Neste EP você fará um sintetizador aditivo polifônico, baseado na representação PCM, que gera um sinal de áudio a partir de um conjunto de instruções e o toca. Neste conjunto de instruções você descreverá quais os timbres do seu instrumento, através da síntese aditiva, e a envoltória de amplitude. Em seguida, assim como em uma partitura, você irá descrever os eventos musicais que o seu instrumento irá tocar (nota = freqüência fundamental, duração e amplitude).

O som, como sabemos, é transmitido por uma onda mecânica que se propaga no ar e causa vibrações em nosso aparelho auditivo, que podem ser percebidas como música, fala, ou um ruído qualquer. O *sinal sonoro* corresponde ao gráfico da variação da pressão do ar em função do tempo. Alguns sons correspondem a oscilações periódicas ou quase-periódicas que dão origem à sensação de *altura musical*. Estes sons podem ser caracterizados grosseiramente por alguns parâmetros: a *forma básica de onda* correspondente ao sinal sonoro medido durante um *período completo* da oscilação; a *amplitude* da oscilação que está associada à sensação de volume; a *freqüência* da oscilação, medida em Hertz (Hz), que determina a altura musical percebida e corresponde ao número de repetições da forma básica de onda por segundo (em outras palavras, a freqüência em Hz é o inverso do período, medido em segundos); e a *duração* do som. Tal caracterização fornece uma representação simbólica do fenômeno oscilatório, reduzindo-o a um número pequeno de parâmetros.

A síntese aditiva é uma técnica de áudio com o objetivo de criar um timbre musical a partir da união de ondas sonoras básicas, criando formas de onda mais complexas. Essas ondas básicas podem ter diferentes formas (como ondas senoidais, quadradas, triangulares etc.), e a freqüencia delas é um múltiplo de uma freqüência fundamental. Assim, cada onda que contribui na construção é chamadas de uma parcial. Além disso, a contribuição de cada parcial no som final é controlada através de sua amplitude. Neste trabalho, vamos nos ater a uma senóide como forma de onda básica e a ela adicionaremos multiplos de sua freqüência. Vale lembrar que esses multiplos não precisam ser necessariamente números inteiros (diferente da série harmônica). A Figura 1 mostra um exemplo de síntese aditiva, na qual uma onda é formada a partir da soma de duas ondas mais simples:

Um evento sonoro i do sintetizador aditivo, para um determinado instante t no tempo, vai ser calculado então pela seguinte função, na qual Ampi indica a amplitude da onda, Nparciais o número de parciais que formam a onda, Arelj a amplitude de cada parcial, Frqi a freqüência fundamental e Parj a parcial e tEvei o instante em que se inicia o evento sonoro

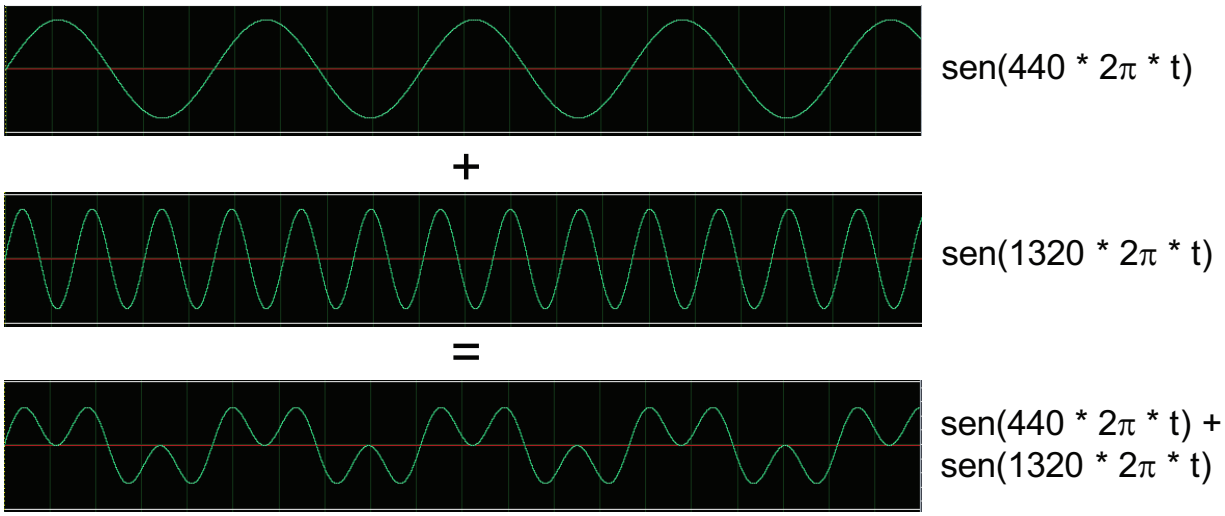

Figura 1: Adição de duas ondas sonoras

$$\text{fi}(t) = \text{Ampi} * \text{env}(t - \text{tEvei}, \text{Duri}) * \sum_{j=1}^{\text{Nparciais}} \text{ARelj} * \cos(2\pi * \text{Frqi} * \text{Parj} * (t - \text{tEvei}))$$

e a função envoltória de amplitude é dada por

$$\text{env}(x, \text{Dur}) = \begin{cases} 0 & x < 0 \text{ ou } x > \text{Dur} \\ \frac{x}{\text{Atk}} & 0 \leq x \leq \text{Atk} \\ 1 & \text{Atk} < x < \text{Dur} - \text{Rel} \\ \frac{\text{Dur}-x}{\text{Rel}} & \text{Dur} - \text{Rel} \leq x \leq \text{Dur} \end{cases}$$

na qual Atk indica o tempo do ataque em que a onda vai de uma amplitude 0 até a Ampi, Dur a duração do evento sonoro e Rel o tempo decaimento que leva para o som ir de Ampi até 0.

Lembre-se que o computador não trabalha com o tempo contínuo, portanto é necessário discretizar as funções que envolvem tempo baseado na freqüência de amostragem com que você está trabalhando. Além disso, o sintetizador é polifônico, portanto vários eventos sonoros podem acontecer ao mesmo tempo, as ondas resultantes dos eventos sonoros devem ser somadas.

Formato de arquivos WAVE

Um arquivo de áudio, por outro lado, é uma representação explícita do sinal sonoro que permite seu armazenamento utilizando espaço de memória (digital) finito. A representação aqui descrita é conhecida como *Pulse Code Modulation* (PCM) e utiliza a técnica de amostragem, que consiste em medir o sinal sonoro regularmente a intervalos de tempo pequenos obtendo *amostras* do sinal. O número de amostras obtidas para cada segundo do som é chamado de *taxa de amostragem* do sinal e está associado à precisão da representação do sinal sonoro no eixo horizontal (do tempo); quanto maior a taxa de amostragem, melhor a representação do som original e maior a fidelidade obtida na

hora de reproduzir o sinal armazenado[1]. O *número de bits* utilizados para armazenar cada amostra está associado à precisão da representação do gráfico no eixo vertical (da variação de pressão do ar). Um arquivo de áudio pode armazenar vários *canais* diferentes contendo sinais sonoros que serão reproduzidos simultaneamente em um sistema com várias caixas acústicas. Como exemplo, se armazenamos 44100 amostras por segundo, com 16 bits por amostra, em um arquivo de áudio estéreo, então cada segundo de som ocupa $44100 * 16 * 2$ bits = 176400 bytes deste arquivo[2].

Arquivos WAVE seguem o padrão RIFF (*Resource Interchange File Format*) e consistem de blocos (*chunks*), cada um dos quais possui um cabeçalho (identificador + tamanho dos dados) e uma seqüência de dados. Todos os valores inteiros são armazenados no formato *little-endian*(ou seja, uma seqüência de bytes $b_0, b_1, \ldots, b_k$ corresponde ao número inteiro $b_0 + 256b_1 + \ldots + 256^k b_k$).

Posição	Conteúdo	Observações
0	"RIFF"	identificador do bloco
4	tamanho do bloco	= 36 + tamanho do bloco de amostras
8	"WAVE"	identificador do bloco
12	"fmt "	identificador do bloco
16	tamanho do bloco de formato	= 16
20	formato das amostras	= 1 (PCM)
22	número de canais (n)	= 1 ou 2
24	taxa de amostragem em Hz (t)	
28	bytes por segundo	= t*n*(b/8)
32	número total de bytes por amostra	= n*(b/8)
34	número de bits por amostra (b)	= 8 ou 16
36	"data"	identificador do bloco
40	tamanho do bloco de amostras	= n.amostras*n*(b/8)
44	amostras	

Observações:

1. Os tamanhos não incluem os bytes do cabeçalho. Por exemplo: se as amostras ocupam 1204 bytes, então este é o número que aparece na posição 40 (e não 1204-8).

2. Para arquivos com mais de um canal, as amostras aparecem na ordem

   $$\text{amostra}_1\text{-canal}_1, \text{amostra}_1\text{-canal}_2, \ldots, \text{amostra}_1\text{-canal}_K,$$

   $$\text{amostra}_2\text{-canal}_1, \text{amostra}_2\text{-canal}_2, \ldots, \text{amostra}_2\text{-canal}_K, \ldots$$

   A convenção para arquivos estéreo é canal_1=esquerdo e canal_2=direito.

3. Amostras de 8-bits são armazenadas como `unsigned`. Assim os valores $0, \ldots, 127, 128, \ldots, 255$ representam os inteiros $-128, \ldots, -1, 0, \ldots, 127$.

4. Amostras de 16-bits são armazenadas como `signed` usando complemento de 2. Os códigos (em notação *little-endian*) 00000000 00000000, 00000001 00000000, ... , 11111111 01111111,

[1] A taxa de amostragem define não apenas a qualidade da representação da onda sonora como também a máxima freqüência sonora representável, que é conhecida como freqüência de Nyquist e vale $R/2$ Hz.

[2] Estes valores correspondem à qualidade de CD; outro padrão comum é 8000 amostras de 8 bits por segundo em um único canal, que é aproximadamente a qualidade do telefone.

00000000 10000000, 00000001 10000000, ... , 11111111 11111111 representam, respectivamente, os inteiros 0,1, ... ,32767,-32768,-32767, ... ,-1. Pode-se obter a representação do inteiro $-n$ tomando-se o complemento bit-a-bit da representação de n e somando-se 1 ao resultado. Por exemplo: 10101010 01010101 = 170 + 85*256 = 21930, e assim -21930 = 01010101 10101010 + 00000001 00000000 = 01010110 10101010.

5. Escreva sempre um byte de cada vez na saída; isso permite um controle melhor dos números inteiros em notação *little endian*, bem como das amostras de 16-bits.

Para mais informações sobre o formato WAVE:

`http://sox.sourceforge.net/AudioFormats.html`
`http://ccrma-www.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/`

O que seu programa deverá fazer

A interface (em linha de comando) do sintetizador será

```
java Sintetizador R B C entrada.txt
```

na qual `R`>0 (inteiro) é a taxa de amostragem, `B`=8 ou `B`=16 é o número de bits de cada amostra e `C`=1 ou `C`=2 é o número de canais, `entrada.txt` é o arquivo que contém as instruções para sintetização do som. O arquivo `entrada.txt` consiste em uma seqüência de números (reais) positivos:

```
NInstr
Inst1 Atk Rel
NParciais
Par1 ARel1
Par2 ARel2
...  ...
ParN ARelN
Inst2 Atk Rel
Par1 ARel1
Par2 ARel2
...  ...
ParN ARelN
...
MEventos
tEve1 Inst1 Dur1 Frq1 Amp1
tEve2 Inst2 Dur2 Frq2 Amp2
...  ...  ...  ...
tEveM InstM DurM FrqM AmpM
```

no qual `NInstr` indica o número de instrumentos que serão descritos, `Atk` e `Rel` (em milissegundos) controlam a envoltória dinâmica de cada evento para esse instrumento, `NParciais` é o número de parciais, `Parj` e `ARelj` são o índice (pode ser fracionário) e a amplitude relativa (entre 0 e 1) do parcial j. `MEventos` é o número de eventos gerados e cada evento é descrito por um tempo

inicial `tEve`, o instrumento que será utilizado `Inst` e uma duração `Dur` (em segundos), bem como a freqüência fundamental da onda `Frq` e uma amplitude `Amp` (entre 0 e 1). A lista de eventos deve obedecer `tEve1` $\leq$ `tEve2` $\leq$ ... $\leq$ `tEveM`.

Seu programa não precisa fazer verificações de consistência dos parâmetros de entrada, mas deve observar a possibilidade de estouro de valores das amostras na síntese. Para isso, utilize uma constante MAXAMP, que é o maior valor permitido de amplitude (32767 ou 127 dependendo do número de bits). Um valor acima de MAXAMP deve ser transformado em MAXAMP, bem como um valor abaixo de -MAXAMP dever ser transformado em -MAXAMP.

Você pode gerar o áudio de duas formas: criar um arquivo de áudio (WAVE) a partir das informações do arquivo de entrada e o tocar; ou então escrever diretamente os bytes do seu sinal na saída de áudio do computador. Fica a seu critério escolher. Na página da disciplica você poderá encontrar dois exemplos de arquivo de entrada e suas correspondetes saídas, em WAVE, para que possa conferir o funcionamento do seu programa.

Recomenda-se que o exercíco seja desenvolvido na linguagem Java, uma vez que ela disponibiliza a biblioteca Java Sound[3] para manipulação de áudio.

Vocês deverão entregar um arquivo comprimido contendo os arquivos .java, um exemplo interessante de arquivos de entrada que foi usado como teste e um arquivo LEIAME que explica resumidamente a organização do código desenvolvido e como ele deve ser executado.

Bom trabalho e divirta-se!

[3] http://java.sun.com/products/java-media/sound/