Computação Musical – 1º Exercício Programa
Prof. Marcelo Queiroz
Data de entrega: **29/4/2005**

Instruções: Os EPs devem ser feitos individualmente. A entrega será feita por e-mail para mim até a data indicada.

Sintetizador por Síntese Aditiva

O objetivo deste EP é aprender a manipular diretamente um formato específico de arquivo de áudio, o RIFF/WAVE. Escreveremos um programa em C que lê um arquivo texto com instruções específicas de síntese aditiva e produz uma saída .wav correspondente.

A interface (em linha de comando) do sintetizador será

```
sintetiza R B C entrada.txt saida.wav
```

onde 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. O arquivo entrada.txt consiste em uma seqüência de números (reais) positivos:

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

onde Atk e Rel (em milissegundos) controlam a envoltória dinâmica de cada evento, 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 tEvej e uma duração Durj (em segundos), bem como uma amplitude Ampj (entre 0 e 1). A lista de eventos deve obedecer tEve1≤tEve2≤ $\cdots$ ≤tEveM.

O evento i corresponde a uma função

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

onde MAXAMP é o maior valor permitido de amplitude (32767 ou 127 dependendo do número de bits) e a função envoltória é dada por

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

Seu programa não deve fazer verificações de consistência dos parâmetros de entrada, mas deve observar a possibilidade de estouro de valores das amostras na síntese (um valor acima de MAXAMP deve ser transformado em MAXAMP, bem como um valor abaixo de -MAXAMP deve ser transformado em -MAXAMP).

Formato de arquivos WAVE

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 aparece na posição 40 (e não 1204-8).

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

   amostra$_1$-canal$_1$, amostra$_1$-canal$_2$,..., amostra$_1$-canal$_K$,

   amostra$_2$-canal$_1$, amostra$_2$-canal$_2$,..., amostra$_2$-canal$_K$,...

   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,

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/`