Introdução ao Csound

Thilo Koch
Grupo Computação Musical - IME - USP

5 de novembro de 2013

Csound é

- um programa / ambiente / *engine* poderoso para sintetizar som
- um dos programas mais antigos e de maior influência
- estabeleceu padrão seguido por muitos outros software
- estrutura e sintaxe simples $\rightarrow$ incrivelmente flexível
- muito usado por artistas, pesquisadores e músicos ...
- muito bem documentado

A Apresentação dará uma visão geral do Csound e de algumas ferramentas. Além disso vai explicar o básico da sintaxe da linguagem e uns exemplos.

História de Csound

Família de linguagens MUSIC-N

- MUSIC (1) foi a primeira linguagem de programação de som
- inventada por Max Mathews em 1957
- rodava numa IBM 704 (Nova York), escrito em *assembler*
- primeira peça era de 17 segundos
- ideia básica de MUSIC: Uma **orquestra** e uma **partitura** são combinados para produzir a **peça**
- MUSIC foi seguido por MUSIC 2 até MUSIC 5 (fim do anos 60)

História de Csound

- no começo dos anos 60 Max Mathews deu código fonte (da versão 4) para Barry Vercoe
- ele portou para as máquinas que tinha *a mão* (IBM 360 $\rightarrow$ MUSIC 360, PDP 11 $\rightarrow$ MUSIC 11)
- anos 80: com a disponibilidade de microcomputadores baratos e C como língua franca Barry Vercoe reescreveu tudo em C para a melhor portabilidade $\rightarrow$ 1985: Csound versão 1.0
- 1990: apresentação: performance ao vivo com Csound
- desde então foi desenvolvido incrementalmente adicionando cada vez novos *features* (por ex. Csound Extended para utilizar GPU - 2000)
- 2013: Versão 6.0

Estruturas básicas de síntese

Hello World

- Código:

```
<CsoundSynthesizer>
<CsOptions>

-o helloworld.wav

</CsOptions>

<CsInstruments>

     instr    101
a1   oscil    15000, 440
     out      a1
     endin

</CsInstruments>

<CsScore>

; inst   start    duration
i 101        0         3
e
</CsScore>
</CsoundSynthesizer>
```

- Execução:

```
csound <nome_do_arquivo.csd>
```

Exemplo Síntese FM

- proposta 1973 por John Chowning
- modular frequência de um *carrier* dinamicamente
- síntese é controlada pelo indice de modulação (deviação da modulação / frequência da modulação)
- percussão: índice alto no começo depois descendo → muitas *sidebands* não harmônicos no começo, depois reduzem-se
- $f(t) = A_c cos\{2\pi[f_c + A_m cos(2\pi f_m t)]t\}$

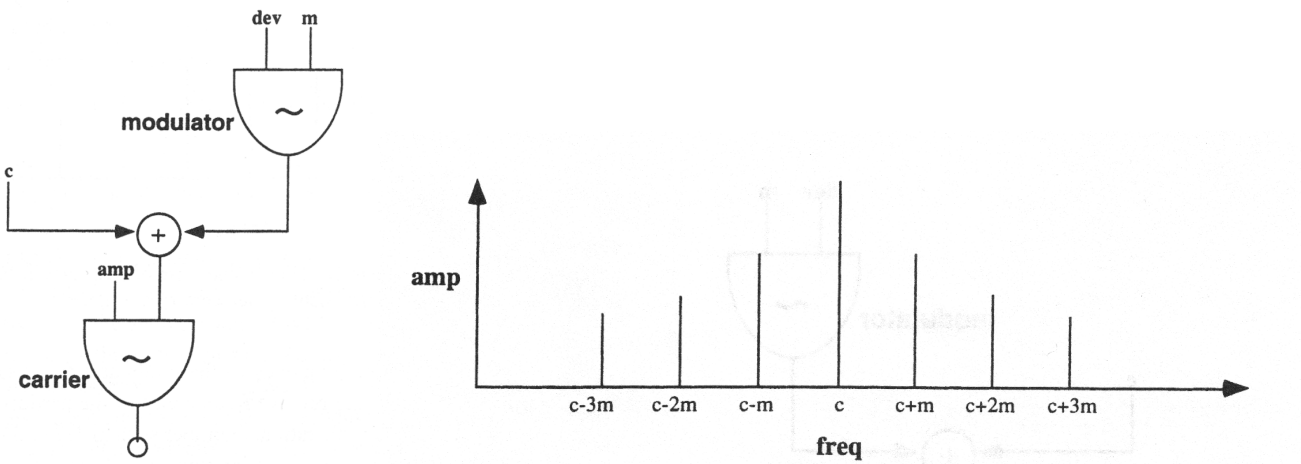

Figura: *FM synthesis: block diagram and spectrum, c = carrier frequency, m = modulation frequency.*

Síntese FM em Csound: O Instrumento

```
<CsInstruments>

sr          =          44100
kr          =          4410
ksmps       =          10
nchnls      =          1

            instr      901
inotedur    =          p3
imaxamp     =          ampdb(p4)
icarrfreq   =          p5
imodfreq    =          p6
ilowndx     =          p7
indxdiff    =          p8-p7
aampenv     linseg     p9, p14*p3, p10, p15*p3, p11, p16*p3, p12, p17*p3, p13
adevenv     linseg     p18, p23*p3, p19, p24*p3, p20, p25*p3, p21, p26*p3, p22
amodosc     oscili     (ilowndx+indxdiff*adevenv)*imodfreq, imodfreq, 1
acarosc     oscili     imaxamp*aampenv, icarrfreq+amodosc, 1
            out        acarosc
            endin

; PARAMETERS DEFINING THE ADSR AMPLITUDE ENVELOPE (TIMES ARE A PERCENTAGE OF p3)
;    attack amp   = p9      attack length  = p14
;    decay amp    = p10     decay length   = p15
;    sustain amp  = p11     sustain length = p16
;    release amp  = p12     release length = p17
;    end amp      = p13

</CsInstruments>
```

Tempo de atualização das variáveis: i<nome>→ uma vez cada nota; k<nome>→ com taxa de controle (kr); a<nome>→ com taxa de amostras (sr).

Síntese FM em Csound: A Partitura (score)

```
<CsScore>

; chowdrum.sco
; DRUM SOUNDS WITH CHOWNING FM

f 1 0 4096 10  1     ; create and fill table 1 at time 0 with 4096 samples with GEN10 and no harmonics


;                                   FINAL AMPLITUDE ENVELOPE                    INDEX(DEVIATION) ENVELOPE
;                                   VALUES                  TIME                VALUES                  TIME
;p1 p2 p3   p4   p5   p6   p7 p8   p9   p10 p11 p12 p13   p14 p15 p16 p17   p18 p19 p20 p21 p22   p23 p24 p25 p26
;IN ST DUR AMP CAR MOD I1 I2  ATK DEC SUS REL END  ATK DEC SUS REL  ATK DEC SUS REL END  ATK DEC SUS REL

i 901   0    0.2 88    80   55 0  5    .75 .8  1.0 .15 .0  .125 .125 .25 .5  1.0 .0 .0  .0  .0  .125 .25 .25
i 901  .2    0.2 88   411  377 0  5    .75 .8  1.0 .15 .0  .125 .125 .25 .5  1.0 .0 .0  .0  .0  .125 .25 .25
i 901  .4    0.2 88   200  161 0  5    .75 .8  1.0 .15 .0  .125 .125 .25 .5  1.0 .0 .0  .0  .0  .125 .25 .25
s
```

Função das linhas:

- f - criar e preencher *lookup tables*
- i - evento de nota
- s - fim de subseção
- ; - comentário (será ignorada na compilação)

Síntese FM em Csound: A Partitura (score)

```
i 901   0   0.2     88 200 161   0   25   .75  .8  1.0  .15   .0  .125  .125  .25  .5   1.0   .0   .0   .0   .0  .125  .25  .25
i 901   +   0.2005  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2007  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2011  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2016  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2022  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2031  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2044  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2062  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2088  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2125  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2177  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2225  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2353  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.25    <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.2707  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.3     <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.3414  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.4     <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.4828  <   <   <   <   <    <    <   <    <    <   <     <     <    <    <     <    <    <    <    <     <    <
i 901   +   0.6     88  440 440   0   5    .0   1.0  .75  .66  .0   .17   .17   .49  .17  .0    1.0  .75  .66  .0   .17   .17  .49

</CsScore>
```

Função dos símbolos:

- $<$ - interpolar valores
- $+$ - conectar notas
- . - copiar valores

Opcodes

- Opcodes são essenciais na definição do instrumento
- muitos usam tabelas f - *lookup tables* - para achar valores
- tem milhares para todo tipo de aplicação
- podem ser definidos pelo usuário

Exemplos:

- geradores de sinais (oscil, lfo, foscil, phasor, tablei)
- operadores matemáticos (+, -, max, ampdb)
- entrada e saída (in, diskin, readk, dumpk)
- modificadores de sinais (convolve, delay, pan, reson)
- controle de instrumento (if ... else ..., widgets, ihold)
- processamento pelo espectro (pvoc, dnoise)
- network (OSClisten, OSCsend, socksend, sockrecv)

GENerators

- GENs (geradores) preenchem tabelas de valores (tabelas f - *lookup tables*)
- instrumentos varrem as tabelas com indices (acesso rápido)
- tem dezenas para todo tipo de aplicação
- podem ter 2 dimensões

Exemplos:

- GEN09, GEN10, GEN19: composição de senoídes (com adição de harmônicos e não-harmônicos de amplitudes diferentes)
- GEN05, GEN06, GEN07: segmentos (lineares, cúbicos, exponentiais)
- GEN04: waveshaping / normalização - analisa outra tabela f para gerar função adequada
- GEN01: preencher tabela com amostras (por exemplo do disco)
- GEN20: janelamento (Hamming, Hanning, Bartlett ...)

Configuração para Performance ao vivo

```
<CsOptions>

; Select audio/midi flags here according to platform
-odac     ;;;RT audio out

;-iadc    ;;;uncomment -iadc if RT audio input is needed too

; For Non-realtime ouput leave only the line below:
; -o buzz.wav -W    ;;; for file output any platform

; -d                ;;; less verbose
; -+rtmidi=portmidi ;;; load midi
; -M1               ;;; midi-device 1 for input
; -Q2               ;;; midi-device 2 for output

</CsOptions>
```

Ferramentas

GUIs

Como o input para Csound é somente texto estruturado, tornou-se comum gerar código com uma outra ferramenta e *dar de comer* a Csound → usar Csound como *backend sound engine*.

CsoundQT:

- vem junto com Csound
- editor multi-aba com sistema de *help* para OPCODES
- *widgets* para controlar performance ao vivo

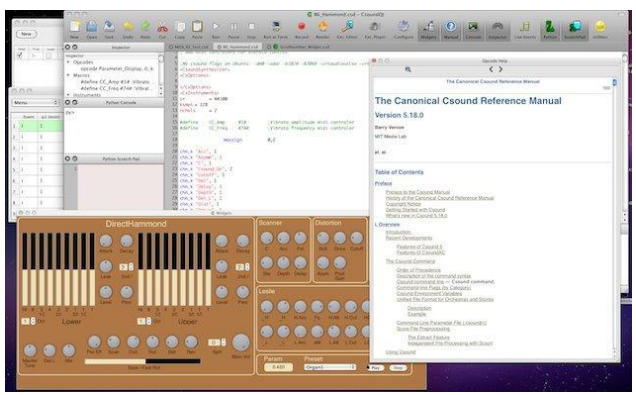

Figura: *Screenshot: CsoundQT with help and widgets.*

Ferramentas

GUIs

blue

- editor multi-aba, orientado a composição
- arranjar score numa *timeline*
- trabalhar com abstrações (SoundObject, PolyObject, NoteProcessor)
- *mixer interface*, *graphical instruments*

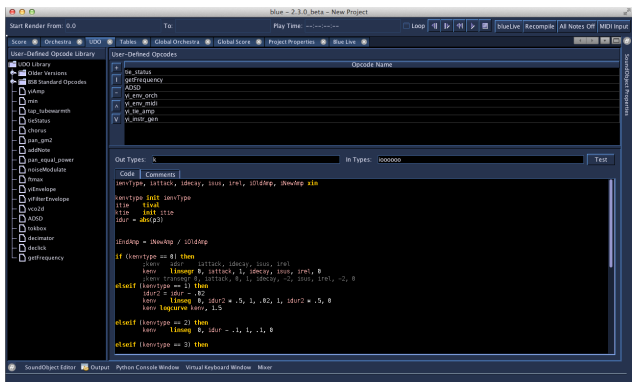

Figura: *Screenshot: blue - User defined Opcodes.*

Figura: *Screenshots: Timeline, Orchestra, Mixer.*

Ferramentas

- **HPK composer**: construir cenas de 3D e exportar para Csound
- **Cabbage**: construção de instrumentos e GUIs (para depois rodar *standalone*)

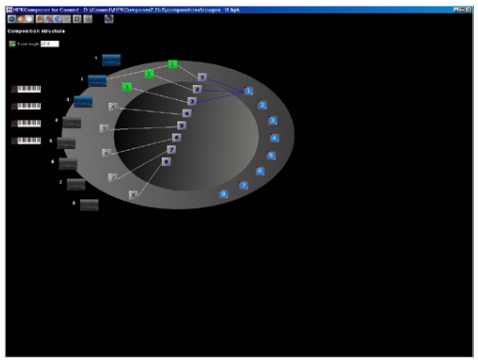

Figura: *Screenshot: HPK composer.*

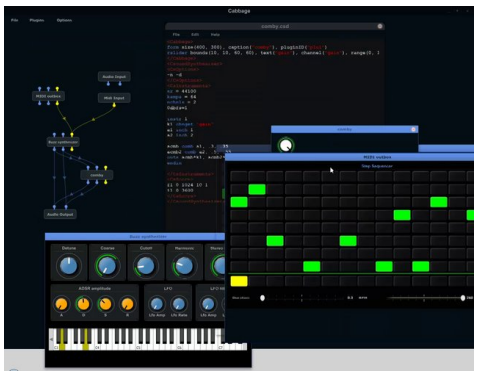

Figura: *Screenshot: Cabbage.*

- **beats**: domain specific language (DSL) para gerar score em notação ocidental
- **Extended score language**: *bin*= produzir score com ferramenta externa

Extended score language

Csound File:

```
<CsoundSynthesizer>
<CsOptions>
-odac
</CsOptions>

<CsInstruments>
     instr    101
a1   oscil    15000, 440
     out       a1
     endin
</CsInstruments>

<CsScore bin="python genscore.py">

</CsScore>
</CsoundSynthesizer>
```

Python Script:

```
from sys import argv

with open(argv[2], 'w') as f:
    s = "i 101  0   3 "
    f.write(s)
f.closed
```

Integração

- csound-x.el: integração com emacs
- netcsound: csound renderização *online* http://dream.cs.bath.ac.uk/netcsound
- csound~ (MAX / Pd), csound-VST, exportar para FAUST, fluidsynth
- Csound *bindings/API* para Python, Lua, Tcl, Java, Lisp
- OPCODES para integar, por exemplo: Wii, P5Glove, MIDI, OSC, jack

Novidades da versão 6

Mais um passo incremental

- *live coding*: substituir instrumentos ao vivo enquanto Csound está rodando, adicionar eventos (notas)
- aplicar valores k com precisão de amostra
- novos OPCODES
  - faustgen: importar códigos do FAUST
  - *arrays* multidimensionais
  - readscore (gerar eventos ao vivo), compilestr (compilar instrumentos ao vivo)
- opção *realtime* para ler e escrever asincronamente no disco (importante para performances ao vivo)
- embelezação de síntaxe: *score line* com mais que um *string*, operadores +, - (e não mais: add, div etc.)
- fixes (!)

Fontes e links

- **csound em casa** http://www.csounds.com
- **tools/frontends** http://www.csounds.com/resources/utilitiestools
- **FLOSS manual** http://www.flossmanuals.net/csound
- **John Chowning na TV em 1983** https://www.youtube.com/watch?v=8QTGLQq3DwU
- **Cabbage videos** http://vimeo.com/user5771754/videos

Muito obrigado!