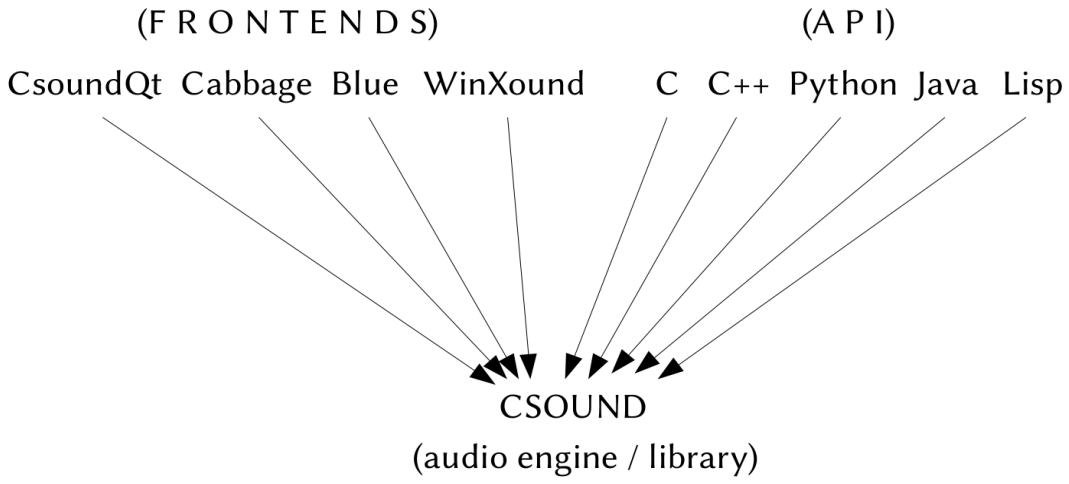
Joachim Heintz
Live-Elektronik mit modernem Csound (und CsoundQt)

Musikhochschule Freiburg
8 November 2018

01 Download und Hilfe

- Download **Csound** (**CsoundQt** ist in den Installern für Mac und Windows **enthalten**)
  https://csound.com/download.html
- **Letzte CsoundQt** Version (falls nötig)
  https://github.com/CsoundQt/CsoundQt/releases
- **Csound6~** Objekt für **PD**
  https://github.com/csound/csound_pd
- Freies Online Lehrbuch (Csound **FLOSS Manual**)
  http://write.flossmanuals.net/csound
- CsoundQt Website und Documentation
  http://csoundqt.github.io
  http://csoundqt.github.io/pages/documentation.html
- *Getting Startet* ist im CsoundQt Menu enthalten
  Examples > Getting Started
- Neues Csound Book (2016)
  Lazzarini et.al. — Csound, A Sound and Music Computing Language
  Springer Seite Inhaltsverzeichnis

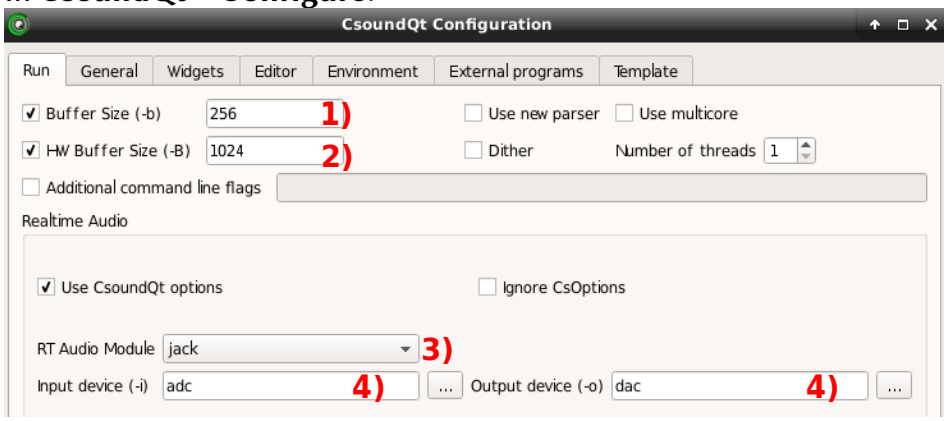
02 Csound und Frontends



Im Fall von CsoundQt
CsoundQt -> Editor, Widgets, Umgebung
Csound -> Syntax, Reference Manual, Audio Engine

03 Wichtige Einstellungen

In **CsoundQt > Configure:**



1) Software buffer size auf 256 Samples setzen
2) Hardware buffer size auf 1024 setzen
3) Real Time Audio Module gemäß Betriebssystem setzen (meist portaudio)
4) Immer **adc** als input device und **dac** als output device setzen. Das Standard-Audiogerät über das Betriebssystem auswählen.

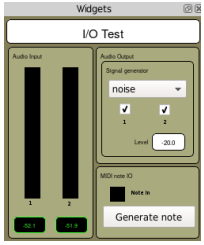
Im **Csound Programm (.csd Datei):**

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 44100 5)
ksmps = 32  6)
nchnls = 2  7)
0dbfs = 1  8)

instr 1
endin

</CsInstruments>
<CsScore>

</CsScore>
</CsoundSynthesizer>
```

5) Sample rate setzen (normalerweise 44100 oder 48000)
6) Anzahl von Samples per Block (control-cycle) setzen (normal 32)
7) Anzahl von Kanälen setzen (2 = Stereo)
8) Immer 1 als die Zahl setzen, die 0 dB repräsentiert

Wenn man dies in Configure > Template als default setzt, wird es für jede neue Datei in CsoundQt aufgerufen.

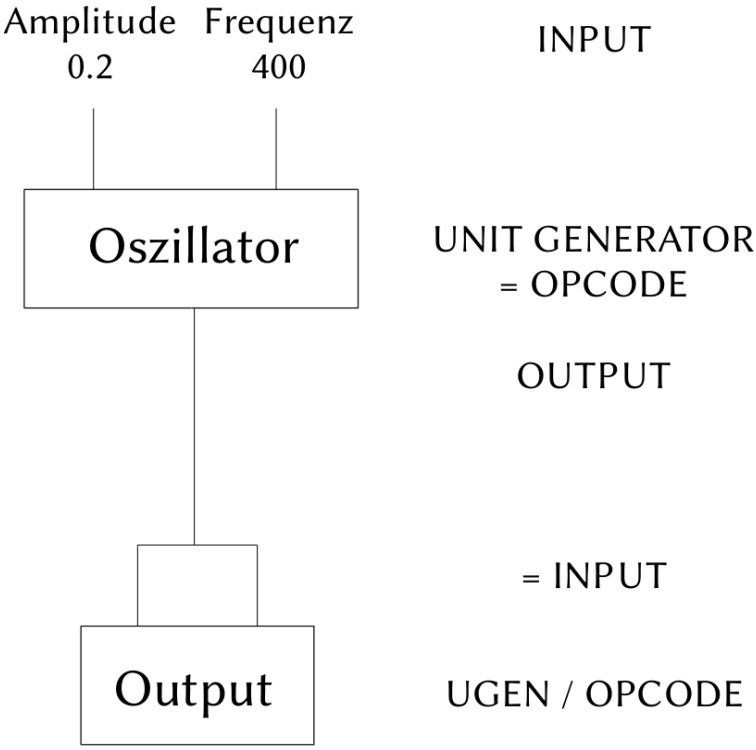
Zum Testen jetzt **CsoundQt > Examples > Useful > IO Test** ausführen:
1. Links oben auf den “Run” button klicken.
2. Auf den “Widgets” button klicken. Es sollte das Fenster rechts erscheinen.



3. Auf die Checkboxes klicken um das Testsignsl zu hören.

04 Coding

Ein typisches Fließdiagramm



Die Übersetzung eines solchen Diagramms in Csound Code funktioniert so:

- Man muss die **Opcode Namen** wissen (hier: Oszillator = *poscil*, Output = *out*)
- Für jede Einheit aus Input-Opcode-Output wird eine Zeile in dieser Reihenfolge geschrieben:
  OUTPUT OPCODE INPUT
  (Falls es mehrere Inputs oder Outputs gibt, werden sie durch Kommas getrennt.)
- Der erste Buchstabe einer Variablen bezeichnet den Typ:
  - **i**Bla -> bleibt konstant für einen Aufruf des Instruments
  - **a**Bla -> Audio Signal (jedes Sample wird neu berechnet)
  - **k**Bla -> Kontroll Signal (jeder Block wird neu berechnet)

Das Ganze wird dann noch in ein **Instrument** (beginnend mit *instr* und endend mit *endin*) eingeschlossen:

```
instr Sinus_Test
 aSinus poscil 0.2, 400
 out aSinus, aSinus
endin
```

Es gibt verschiedene Möglichkeiten, ein Instrument zu **rufen**. (Oder etwas präziser: eine Instanz zu erzeugen.)
1. Eine Zeile wie diese in den *CsScore* Abschnitt der .csd Datei schreiben:

```
i "Sine_Test" 0 3
```

2. Im “Widgets” Fenster Rechtsklick und “Create Button” auswählen. Es öffnet sich der *properties* Dialog. Dort dieselbe Score Zeile als “Event” einfügen:

```
i "Sine_Test" 0 3
```

Jetzt Csound **mit leerer score section** starten. Start auf den Button startet dann das Instrument.

3. Auch innerhalb des Csound Codes (also nicht im Score) kann das Instrument gerufen werden:

```
schedule "Sine_Test", 0, 3
```

05 Selbstruf Spiel

Durch *schedule* kann sich ein Instrument auch selbst rufen. Dieser Code lässt das Instrument *Perpetuum_Sinus* zwei Sekunden spielen und ruft dann nach einer Sekunde Pause die nächste Instanz:

```
instr Perpetuum_Sinus
 schedule “Perpetuum_Sinus”, 3, 2
 aSinus poscil 0.2, 400
 out aSinus, aSinus
endin
```

Durch verschiedene Zufallsprozesse kann man vielleicht auf diese Version kommen:

```
instr Perpetuum_Sinus_5

 iPreviousNote = p4
 iDeviation random -3, 3
 iThisNote = iPreviousNote + iDeviation

 iDb random -30, -6
 iAmp = ampdb(iDb)
 aEnv transeg iAmp, p3, -3, 0
 aSinus poscil aEnv, cpsmidinn(iThisNote)
 out aSinus, aSinus

 iStart random 1/3, 2
 iDur random 1, 7
 schedule "Perpetuum_Sinus_5", iStart, iDur, iThisNote

endin
schedule "Perpetuum_Sinus_5", 0, 3, 60
```

Nach Belieben kann man das auch in **funktionaler Schreibweise** schreiben:

```
instr Perpetuum_Sinus_6

 iThisNote = p4 + random:i(-3,3)

 aEnv = transeg:a(ampdb(random:i(-30,-6)),p3,-3,0)
 aSinus = poscil:a(aEnv,cpsmidinn(iThisNote))
 out(aSinus,aSinus)

 schedule("Perpetuum_Sinus_6",random:i(1/3,2),random:i(1,7),iThisNote)

endin
schedule("Perpetuum_Sinus_6",0,3,60)
```

06 Live Input

Ein angeschlossenes Mikrofon wird über den *inch* opcode in Csound als Audiosignal empfangen. (Achtung — beim folgenden Code gibt es Feedbackgefahr.)

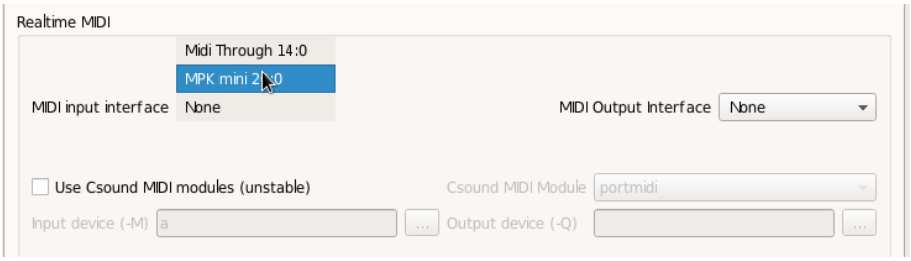
```
 aMic inch 1
 out aMic, aMic
```

Einfaches Beispiel für Ringmodulation des Live-Inputs

```
 aMic inch 1
 aRM poscil aMic, 400
 out aRM, aRM
```

07 MIDI Keyboard (note on/off messages)

Zuerst das MIDI Gerät in CsoundQt's Configure > Run auswählen



Per default startet jeder Druck einer Taste des Keyboards eine Instanz des Instruments

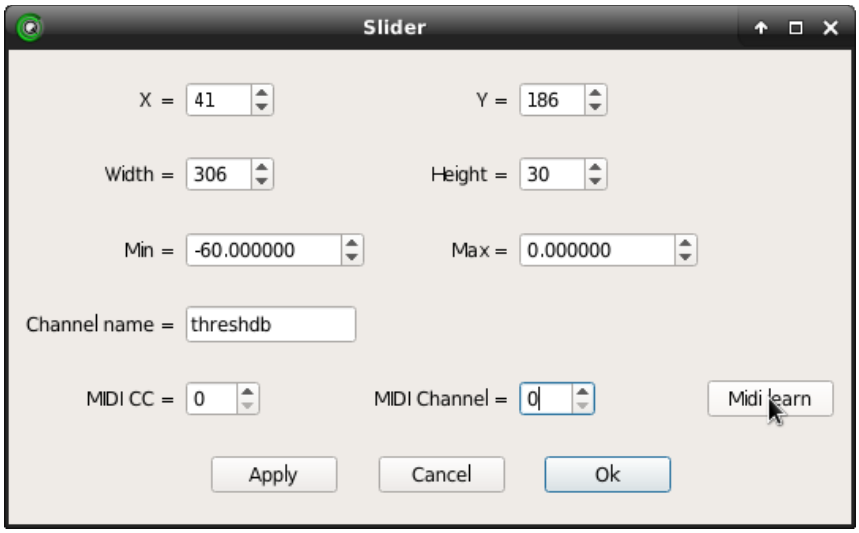
```
instr OneMIDINote
 aNote poscil .2, 400
 out aNote, aNote
endin
```

Liest man die Tastennummer (60 = c') und die Velocity (1-127) mit, und sorgt dafür, dass beim Hochheben der Taste ein Fade-Out stattfindet, hat man einen einfachen (und unbegrenzt polyphonen) Synthesizer:

```
 iCps cpsmidi
 iVel veloc
 aNote poscil ampdb(-64+iVel/2), iCps
 aEnv linenr aNote, 1/20, 1/5, 1/100
 out aEnv, aEnv
```

08 MIDI Control Change

Benutzt man CsoundQt, sollte man einen MIDI Controller am besten direkt mit einem Widget (Slider, Scroll Box, Spin Box oder ähnliches) verbinden. Am einfachsten über den "Midi learn" Button beim Property Dialog eines Widgets:



09 ASCII Tasten (Computertastatur) als Steuerung

Der Opcode *sensekey* gibt zwei Informationen heraus: *kDown* wird 1 wenn eine Taste gedrückt wurde (sonst 0); *kKey* gibt die ASCII Tastennummer heraus. Der folgende Code zeigt die Tastennummer an:

```
kKey, kDown sensekey
if kDown == 1 then
 outvalue "key", kKey
endif
```

Und so kann man beispielsweise ein Instrument durch Drücken der Leerstaste (ASCII 32) rufen:

```
if kDown == 1 && kKey == 32 then
 event "i", "Beep", 0, random:k(0.2,1)
endif
```

Siehe auch in CsoundQt Examples > Useful > SF Snippets Player.

10 OSC (Open Sound Control) empfangen (z.B. von PD)

Das muss Csound wissen, wenn OSC von einer anderen Anwendung empfangen werden soll:

- den port über OSC durch die andere Anwendung gesendet wird
- der String mit der OSC Adresse
- der Typ der zu empfangenden Werte (i=integer, f=float, s=string etc.)

Dieser Code empfängt eine Zahl von PD bei der Adresse "/test" und zeigt sie an:

```
giPort OSCinit 9001

instr Receive
 kVal init 0
 kPing OSClisten giPort, "/test", "f",  kVal
 outvalue "val", kVal
endin
schedule "Receive", 0, 999
```

11 OSC senden (z.B. zu PD)

Dies muss angegeben werden, wenn eine OSC Nachricht geschickt werden soll:

- die IP Adresse des Empfängers, oder "localhost" für diesen Computer
- der Port über den die OSC Nachricht gesendet wird
- die OSC Adresse
- die Datentypen die in einer Nachricht gesendet werden

Als sehr einfaches Beispiel, bei dem eine Zufallszahl über Port 9002 zu PD geschickt wird:

```
instr Send
 kVal randomh 1, 10, 1
 OSCsend kVal, "", 9002, "/test2", "f",  kVal
 outvalue "val", kVal
endin
schedule "Send", 0, 999
```