Joachim Heintz
Live Electronics with Modern Csound (using CsoundQt)

Workshop at Seoul International Computer Music Festival 2018
28 - 30 September 2018

III. Some FFT-Based Instruments in Csound

21 Basic FFT and IFFT in Csound

The familiy of opcodes here starts with **pvs** = phase vocoder streaming.
With ***pvsanal*** we go from time-domain to frequency-domain. There we can manipulate the FFT data.
With ***pvsynth*** we go then back to time domain (inverse FFT).

```
;settings for FFT
iFFTsize = 512 ;depending on the input
iOverlap = iFFTsize / 4
iWinSize = iFFTsize
iWinType = 1 ;von Hann

;mic input
aMic inch 1
;FFT (resulting in a f-signal
fMic pvsanal aMic, iFFTsize, iOverlap, iWinSize, iWinType
;doing anything here ...

;and then back to time domain
aOut pvsynth fMic
out aOut, aOut
```

22 Transposition (Harmonizer)

Basic transposition can be done with ***pvscale***. You can use Csound's *cent()* function to convert cent values to a ratio (cent(0) = 1, cent(1200) = 2, cent(-1200) = 0.5).

```
;doing anything here: transpose
fTranspose pvscale fMic, cent(100)
```

Exercise:

a) Try to write a transposition of three intervalls, for instance -300, 300 and 600 cents.
b) Replace the fixed values by three randomly moving lines, each in the range -50 .. 50 cents. The opcode for this is *randomi*; here you should write randomi:k(-50,50,1,3) which means: minimum=-50, maximum=50, frequency for generating a new value = 1 Hz, start from anywhere in between minimum and maximum.

23 Spectral Shift

The spectrum can be shifted by **pvscale**. Inputs are the frequency shift and the lowest frequency the shift applies to.

```
;doing anything here: spectral shift
kShiftFreq invalue "shiftfreq"
kLowestFreq invalue "lowestfreq"
fShift pvshift fInput, kShiftFreq, kLowestFreq
```

Exercise:

a) Apply again a moving random shift with *randomi*. Display the values in the widget panel.

b) For the shift amount, change the random from interpolating (random**i**) to jumping (hold = random**h**) and try a frequency of about 100 Hz for the changes, so that the spectral shift becomes noisy.

24 Spectral Extract

The opcode **pvstrace** returns a f-signal containing only the N loudest bins.

```
;doing anything here: extract a number of loudest bins
kNumBins invalue "numbins"
fExtract pvstrace fInput, kNumBins
```

Exercise:

a) Let the number of bins change randomly between a minimum and a maximum. Use again *randomh* for this approach.
b) Get rid of the regular changes in introducing another *randomh* which for the *kcps* input of the *randomh* used in a).
c) Make the number of bins depending on the RMS level.

25 Analyze Pitch

In 09, we triggered an instrument depending on the input crossing a threshold. We can now extend this instrument by sending the frequency and amplitude of the signal in this moment, too. Here is an implementation which compares the results of **pvspitch** and **ptrack**. It uses chnget/chnset instead of invalue/outvalue. This is always preferable for more complex programs, with a lot of interaction between Csound and the GUI. Make sure to declare your channels first.

```
;continuous pitch analysis
kFreqPvspitch, kAmpPvspitch pvspitch fInput, 0.01
kFreqPtrack, kDbPtrack ptrack gaIn, 1024

;checkboxes to activate the different analyzers
kOnPvspitch chnget "on_pvspitch"
kOnPtrack chnget "on_ptrack"

;look for threshold crossing and trigger instrument
kThreshDb chnget "threshdb"
kThresh = ampdb(kThreshDb)
kMinTime chnget "mintime"
kPause init 0 ;sec
kPreviousRms init 0
kRms rms gaIn
chnset dbamp(kRms), "rms"
if kRms > kThresh && kPreviousRms <= kThresh && kPause <= 0 then
 if kOnPvspitch == 1 then
  event "i", "Beep", 0, random:k(1,6), kFreqPvspitch, kAmpPvspitch
  chnset kFreqPvspitch, "pvspitchfreq"
 endif
 if kOnPtrack == 1 then
  event "i", "Beep", 0, random:k(1,6), kFreqPtrack, ampdb(kDbPtrack)
  chnset kFreqPtrack, "ptrackfreq"
 endif
 kPause = kMinTime
endif
kPause -= ksmps/sr ;count backwards
kPreviousRms = kRms
```

Exercise:

a) Make instrument “Beep” more interesting.
b) Compare the results of pvspitch and ptrack before sending them to Beep. Define a condition, for instance “if the result deviation is less than 10 Cents”, and a consequence for this, or/and for the other cases.