



Hochschule Macromedia für angewandte Wissenschaften University of Applied Sciences

Kursbezeichnung: FDN & Spectral Processing – eine Projektarbeit im Rahmen
der Veranstaltung: Signalverarbeitung und Audiotechnik

Prüfer: Prof. Ephraim Wegner

Vom Studierenden auszufüllen:

091603717

Matrikelnummer

F-HBh IA IAV 7d 17W

Kohorte

Braun

Nachname

Simon

Vorname

Die Arbeit wird eingereicht als:

(Tragen Sie bitte in die zutreffende Box den Buchstaben X ein)

☒

Einzelarbeit

☐

Gruppenarbeit

Trifft nur auf Gruppenarbeiten zu: (Nur bei Gruppenarbeiten auszufüllen)

Falls Sie eine Gruppenarbeit einreichen, dann müssen bitte die Vor- und Nachnamen aller Gruppenmitglieder aufgeführt werden. Die Namen sind von den jeweiligen Gruppenmitgliedern selbst elektronisch einzutragen. Durch Eintrag des Namens wird bestätigt, dass der/die Studierende mit der Abgabe der Arbeit in der vorliegenden Form einverstanden ist. Ferner wird mit Eintrag des Namens erklärt, die Projektarbeit (bei Gruppenarbeit: den von dem/der jeweiligen Studierenden erstellten und entsprechend gekennzeichneten Teil der Arbeit selbständig und ohne fremde Hilfe angefertigt zu haben. Dabei hat der/die Studierende sich keiner anderen Hilfsmittel bedient als derjenigen, die im beigefügten Quellenverzeichnis genannt sind. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind von dem/der Studierenden als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

1) _____

2) _____

3) _____

4) _____

5) _____

6) _____

7) _____

8) _____

Bewertung der Gruppenarbeiten:

(Tragen Sie bitte in die zutreffende Box den Buchstaben X ein)

☐

Ich beantrage bei meiner Gruppenarbeit eine Individualbewertung

☐

Ich beantrage bei meiner Gruppenarbeit eine Gruppenbewertung

Eine Individualbewertung bedeutet, dass jedes Gruppenmitglied eine individuelle Note bekommt und eine Gruppenbewertung bedeutet, dass jedes Gruppenmitglied eine identische Note bekommt.

23.01.20

Ort/Datum

Simon Johannes Braun

Vollständiger Vor- und Nachname

Vom Prüfer auszufüllen: (Freitext für die Zweitkorrektur)

FDN & Spectral Processing - eine Projektarbeit im Rahmen der Veranstaltung: Signalverarbeitung und Audiotechnik

Simon Johannes Braun
Matrikelnummer: 091603717

1 Feedback Network Delay

Im Seminar Signalverarbeitung im Wintersemester 19/20 haben wir in Gruppenarbeit und als gemeinsames Projekt ein Feedback Delay Network entwickelt. Nach gemeinsamen konzeptionellen Gesprächen, übernahmen wir, die Seminarteilnehmer, die eigenständige Entwicklung einzelner Komponenten, die anschließend zu dem Endergebnis zusammengeführt wurden.

Das Feedback Delay Network besteht aus mehreren eigenständigen Klangerzeugern und der Aufnahme und Verarbeitung von Umgebungsgeräuschen. Das aufgenommene Audiosignal wie auch die intern generierten Audiosignale werden mit Hilfe der "Fast Fourier Transformation" in einzelne Frequenzspektren aufgeteilt. Diese einzelnen analysierten Frequenzen werden danach wiederum an die internen Klangerzeuger übergeben, die mit diesen Frequenzen neue Audiosignale erzeugen. Dieser Prozess wiederholt sich in zufälligen Abständen in einer Endlosschleife. Dadurch wird stetig neuer Raumklang erzeugt, der gleichzeitig wieder aufgenommen, analysiert, re-synthetisiert und wiedergegeben wird.

2 Fast Fourier Transformation

Die Fast Fourier Transformation (FFT) ist ein optimierter Algorithmus zur Implementierung der "Diskreten Fourier Transformation"¹. Dabei wird ein bestimmter und begrenzter Anteil eines Signals in seine Bestandteile zu einzelnen Sinusschwingungen zerlegt. Deren jeweilige Amplituden und Phasen können dann bestimmt werden. Um ein Signal auf Werte in einem bestimmten Frequenzbereich zu untersuchen, ist für die FFT die geeignete Abtastrate und Abtastwerte (Samples) zu bestimmen. Der Wert der Abtastrate ist mindestens doppelt so groß zu wählen, wie die höchste zu analysierende Frequenz sein soll. Also legt der halbe Wert der Abtastrate die Obergrenze für das Frequenzspektrum fest, mit dem das Signal untersucht werden kann. Die Samples bestimmen die Auflösung der einzelnen Amplituden-Messwerte. Je mehr Samples, desto genauer aber auch aufwendiger die Berechnung einzelner Amplituden zu bestimmten Zeitpunkten. Bei einer Abtastrate von 48000 Hz und einer Samplerate von 1024 (2^{10}) können so Frequenzunterschiede in 46.88 Hz Schritten ($48000 \text{ Hz} / 1024$) über eine Messdauer von 21.33 ms ($1024 / 48000 \text{ s}$) ermittelt werden. Da ein bestimmter Bestandteil eines Signals analysiert werden kann, ist es wahrscheinlich, dass die Amplituden am Anfang und am Ende des Signalausschnitts nicht null sind.² Das führt bei der FFT zu unerwünschten Fehl-

¹ McGee, Kevin J., An Introduction to Signal Processing and Fast Fourier Transform, [online] <http://www.fftguru.com/fftguru.com/tutorial.pdf> [14.01.2020]

² National Instruments, Understanding FFTs and Windowing, [online] <https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf> [14.01.2020]

interpretationen, die sich in Frequenzen bemerkbar machen, die im Ursprungssignal gar nicht vorhanden sind. Um dem entgegenzuwirken, wird die Methode des Fensters (windowing) angewandt. Dabei wird der Signalausschnitt am Anfang und am Ende mit null multipliziert (z.B. Hamming).

3 drone.csd

Das Feedback Delay Network verarbeitet akustische Signale. Die nach der FFT berechneten Frequenzen und Amplituden werden gespeichert und können durch programmierte Instrumente in CSound wiedergegeben werden. Ich hatte die Idee solch einen Synthesizer für das Feedback Delay Network zu entwickeln. Dieser Synthesizer soll anhand von einer zugewiesenen Frequenz eine gedehnte Klangfläche erzeugen, die sich über einen bestimmten Zeitraum auf- und abbaut. Der folgende Quellcode zeigt umsetzung des Instruments in CSound.

instr 1

```
43 <CsoundSynthesizer>
44 <CsInstruments>
45 instr 1
46
47 shift      =      .00666667      ;shift it 8/1200.
48 ipitch     =      cpspch(p5)      ;convert parameter 5 to cps.
49 ioct       =      octpch(p5)      ;convert parameter 5 to oct.
50 kadsr      linseg  0, p3/3, 1.0, p3/3, 1.0, p3/6, 0 ;ADSR envelope for output
51 kmod       linseg  0, p3/3, 5, p3/3, 3, p3/6, 0 ;ADSR envelope for I
52 kenv       linseg  p6, p3, p7      ;moves from p6->p7 in p3 sec.
53 ;ares      oscil   1, ipitch, 21
54 ares2      oscil   1, ioct, 21
55 kamp1      =      kmod*((kenv-1)/kenv)/2
56 kamp2      =      kmod*((kenv+1)/kenv)/2
57
58 aindx1     =      abs(kamp1*1/20) ;kamp1 normalized between 0-1. for tbl 23
59
60 aamp3      tablei  aindx1, 23, 1 ;lookup tbl in f23, normal index
61
62 aosc1      oscil   aamp3, ipitch, 22 ;cosine
63 aosc2      oscil   kamp2*ioct,ipitch, 22 ;cosine
```

```

64
65 aamp4      =      exp(-0.5*aamp3+aosc1)
66
67 aoutl      oscil      0.05*kadsr*aamp4, aosc2+cpsoct(ioct+ishift), 21 ;fml outleft
68 aoutr      oscil      0.05*kadsr*aamp4, aosc2+cpsoct(ioct-ishift), 21 ;fml outright
69
70           outs      Sigmoid(aoutl), Sigmoid(aoutr)
71
72 endin
73 </CsInstruments>
74 </CsoundSynthesizer>

```

Das Instrument `instr 1` greift auf den zugehörigen `i1` Werte aus dem `<CsScore>` zu. Es werden mehrere Instrument-Instanzen gleichzeitig und nacheinander gestartet. Dort sind unter dem Parameter `p5` die verschiedenen Ausgangsfrequenzen für das Instrument angegeben. Um vereinfachter und zugänglicher mehrtönige Klänge zu erzeugen werden diese Frequenzen unter `p5` als Kommazahlen angegeben, die harmonische Töne in Oktaven-Intervallen repräsentieren. Da für die Klangerzeugung in CSound Oszillatoren `oscil` die eine ganzzahlige Frequenzangabe als eingabe Parameter erwarten, werden die Zahlen aus dem `i1 table` mit `ioct = octpch(p5)` in Oktaven und mit `ipitch = cpspch(p5)` in Schwingungen pro Sekunde (Frequenzen) umgerechnet. Um ein langsames aufbauen und wieder abflachen des Klanges zu erreichen, wird eine Hüllkurve `kadsr` verwendet, die mit der Instanzdauer des aufgerufenen Instruments `p3` einen linearen Pegelverlauf von 0 über 1 bis 0 auf das Ausgangssignal anwendet. Zwei weitere Hüllkurven `kmod` und `kenv` werden für die letztendliche zusätzliche Modulation der Amplitude des Ausgabe Oszillatoren verwendet. Dafür werden weitere Zwischenschritte durchlaufen. Der Wert von `kamp1` wird durch `kmod*((kenv-1)/kenv)/2` berechnet und wird in einem weiteren Schritt für die Erzeugen eines Indexwertes verwendet `aindx1 = abs(kamp1*1.5/20)`. Dieser Indexwert ist auf einen Wertebereich zwischen 0 und 1 normalisiert und wird zum Abfragen eines Wertes aus einer Besselfunktion (Zylinderfunktion) `f23` verwendet, der wiederum die Amplitude eines weiteren Oszillator `aosc1` bestimmt. Dieser Oszillator verwendet die Frequenzen `ipitch` aus dem `<CsScore>`.

f table

```
162 <CsoundSynthesizer>
163 <CsScore>
164 ;f tables for instr 1
165 f21 0 8192 10 1 ;hughres sine wave
166 f22 0 8192 11 1 ;highres cosine wave
167 f23 0 8192 -12 20.0 ;unscaled ln(I(x)) from 0 to 20.0
168 </CsScore>
169 </CsoundSynthesizer>
```

Ein Weiterer Oszillator `aosc2` wird mit der Grundfrequenz `ipitch` erstellt, dessen Schwingungen die Frequenz des Ausgabe Oszillator modelliert `aosc2+cpsoct(ioct+ishift)`. `ishift` sorgt für eine zusätzliche unterschiedliche Tonverschiebung auf dem rechten und linken Ausgangssignal.

Ausschnitt aus CsScore

```
175 <CsoundSynthesizer>
176 <CsScore>
177 ;instr1 start dur gain freq amp amp2
178 ;p1 p2 p3 p4 p5 p6 p7
179
180
181 i1 0 80 1 4.01 1.0 0.1
182 i1 . . . 4.05 . .
183 i1 . . . 4.08 . .
184 i1 . . . 3.01 . .
185 i1 . . . 3.05 . .
186 i1 . . . 3.08 . .
187
188 i1 40 80 1 4.08 1.0 0.1
189 i1 . . . 4.12 . .
190 i1 . . . 5.03 . .
191 i1 . . . 3.08 . .
192 i1 . . . 3.12 . .
193 i1 . . . 4.03 . .
194
```

```
195 i1 80 100 1 5.12 1.0 0.1
196 i1 . . . 6.03 . .
197 i1 . . . 6.07 . .
198 i1 . . . 4.12 . .
199 i1 . . . 5.03 . .
200 i1 . . . 5.07 . .
201
202 i1 120 100 1 5.06 1.0 0.1
203 i1 . . . 5.10 . .
204 i1 . . . 6.01 . .
205 i1 . . . 4.06 . .
206 i1 . . . 4.10 . .
207 i1 . . . 5.01 . .
208 </CsScore>
209 </CsoundSynthesizer>
```

Nach dem Zusammenfügen der einzelnen Arbeiten, ist in Gruppenarbeit eine interessante, sich immer wieder neu generierende, Klanginstallation entstanden. Dies wurde bisher schon zweimal öffentlich aufgebaut. Zuhörer konnten Im E-Werk Freiburg und im Jos-Fritz Kaffee Freiburg den Klängen lauschen und durch aktives erzeugen von Geräuschen selbst teil der Klanginstallation werden.

drone1.csd

```

167 <CsoundSynthesizer>
168 <CsOptions>
169 -o dac
170 ;~i dac
171 </CsOptions>
172 <CsInstruments>
173
174 sr = 44100
175 ksmps = 32
176 nchnls = 2
177 0dbfs = 1
178
179 opcode Sigmoide, a, a
180   aX xin
181   aPow = exp(aX*3.0)
182   aOut = 2.0 * (1.0 / (1.0 + aPow))- 1.0
183   xout aOut
184 endop
185
186 instr 1
187
188 ishift      =      .00666667      ;shift it 8/1200.
189 ipitch      =      cpspch(p5)      ;convert parameter 5 to cps.
190 ioct        =      octpch(p5)      ;convert parameter 5 to oct.
191 kadsr       linseg 0, p3/3, 1.0, p3/3, 1.0, p3/6, 0 ;ADSR envelope for output
192 kmod        linseg 0, p3/3, 5, p3/3, 3, p3/6, 0 ;ADSR envelope for I
193 kenv        linseg p6, p3, p7      ;moves from p6->p7 in p3 sec.
194 ;ares       oscil 1, ipitch, 21
195 ares2       oscil 1, ioct, 21
196 kamp1       =      kmod*(( kenv-1)/kenv)/2
197 kamp2       =      kmod*(( kenv+1)/kenv)/2
198
199 aindx1      =      abs(kamp1*1/20) ;kamp1 normalized between 0-1. for tbl 23
200
201 aamp3       tablei aindx1, 23, 1 ;lookup tbl in f23, normal index
202
203 aosc1       oscil aamp3, ipitch, 22 ;cosine
204 aosc2       oscil kamp2*ioct,ipitch, 22 ;cosine
205

```



```

206 aamp4      =      exp(-0.5*aamp3+aosc1)
207
208 aoutl      oscil      0.05*kadsr*aamp4, aosc2+cpsoct(ioct+ishift), 21 ;fml outleft
209 aoutr      oscil      0.05*kadsr*aamp4, aosc2+cpsoct(ioct-ishift), 21 ;fml outright
210
211          outs      Sigmoide(aoutl), Sigmoide(aoutr)
212
213 endin
214
215
216 </CsInstruments>
217 <CsScore>
218
219 ;f tables for instr
220 f1 0 512 9 1 1 0 ;sine lo-res
221 f2 0 512 5 4096 512 1 ;exp env
222 f3 0 512 9 10 1 0 16 1.5 0 22 2 0 23 1.5 0 ;inharm wave
223 f4 0 512 9 1 1 0 ;sine
224 f8 0 512 5 256 512 1 ;exp env
225 f9 0 512 5 1 512 1 ;constant value of 1
226 f10 0 512 7 0 50 1 50 .5 300 .5 112 0 ;ADSR
227 f11 0 2048 10 1 ;SINE WAVE hi-res
228 f13 0 1024 7 0 256 1 256 0 256 -1 256 0 ;triangle
229 f14 0 512 7 1 17 1 0 0 495 ;pulse for S&H clk osc
230 f15 0 512 7 0 512 1 0 ;ramp up;;;left=>right
231 f16 0 512 7 1 512 0 0 ;ramp down;;;right=>left
232 f17 0 1024 7 .5 256 1 256 .5 256 0 256 .5 ;triangle with offset
233 f18 0 512 5 1 512 256 ;reverse exp env
234 f20 0 1024 10 1 0 0 0 .7 .7 .7 .7 .7 .7 ;approaching square
235
236 ;f tables for instr 3
237 f21 0 8192 10 1 ;hughres sine wave
238 f22 0 8192 11 1 ;highres cosine wave
239 f23 0 8192 -12 20.0 ;unscaled ln(I(x)) from 0 to 20.0
240
241
242
243 ;instr3 start dur gain freq amp amp2
244 ;p1 p2 p3 p4 p5 p6 p7
245

```

```

246
247 ; p5 shift octaves
248 i1 0 5 1 4.01 1.0 0.1
249 i1 . . . 4.05 . .
250 i1 . . . 4.08 . .
251 i1 . . . 3.01 . .
252 i1 . . . 3.05 . .
253 i1 . . . 3.08 . .
254 i1 . . . 5.01 . .
255 i1 . . . 5.05 . .
256 i1 . . . 5.08 . .
257 i1 . . . 6.01 . .
258 i1 . . . 6.05 . .
259 i1 . . . 6.08 . .
260
261 i1 5 5 1 4.08 1.0 0.1
262 i1 . . . 4.12 . .
263 i1 . . . 5.03 . .
264 i1 . . . 3.08 . .
265 i1 . . . 3.12 . .
266 i1 . . . 4.03 . .
267 i1 . . . 5.08 . .
268 i1 . . . 5.12 . .
269 i1 . . . 6.03 . .
270 i1 . . . 6.08 . .
271 i1 . . . 6.12 . .
272 i1 . . . 7.03 . .
273
274 i1 10 5 1 4.12 1.0 0.1
275 i1 . . . 5.03 . .
276 i1 . . . 5.07 . .
277 i1 . . . 3.12 . .
278 i1 . . . 4.03 . .
279 i1 . . . 4.07 . .
280 i1 . . . 5.12 . .
281 i1 . . . 6.03 . .
282 i1 . . . 6.07 . .
283 i1 . . . 6.12 . .
284 i1 . . . 7.03 . .
285 i1 . . . 7.07 . .

```

```

286
287 i1 15 5 1 5.06 1.0 0.1
288 i1 . . . 5.10 . .
289 i1 . . . 6.01 . .
290 i1 . . . 4.06 . .
291 i1 . . . 4.10 . .
292 i1 . . . 5.01 . .
293 i1 . . . 6.06 . .
294 i1 . . . 6.10 . .
295 i1 . . . 7.01 . .
296 i1 . . . 7.06 . .
297 i1 . . . 7.10 . .
298 i1 . . . 8.01 . .
299
300 i1 0 60 1 4.01 1.0 0.1
301 i1 . . . 4.05 . .
302 i1 . . . 4.08 . .
303 i1 . . . 3.01 . .
304 i1 . . . 3.05 . .
305 i1 . . . 3.08 . .
306
307 i1 20 60 1 4.08 1.0 0.1
308 i1 . . . 4.12 . .
309 i1 . . . 5.03 . .
310 i1 . . . 3.08 . .
311 i1 . . . 3.12 . .
312 i1 . . . 4.03 . .
313
314 i1 60 60 1 5.12 1.0 0.1
315 i1 . . . 6.03 . .
316 i1 . . . 6.07 . .
317 i1 . . . 4.12 . .
318 i1 . . . 5.03 . .
319 i1 . . . 5.07 . .
320
321 i1 80 60 1 5.06 1.0 0.1
322 i1 . . . 5.10 . .
323 i1 . . . 6.01 . .
324 i1 . . . 4.06 . .
325 i1 . . . 4.10 . .

```

```
i1 . . . 5.01 . .

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

i1 100 40 1 4.01 1.0 0.1
i1 . . . 4.05 . .
i1 . . . 4.08 . .
i1 . . . 3.01 . .
i1 . . . 3.05 . .
i1 . . . 3.08 . .
i1 . . . 5.01 . .
i1 . . . 5.05 . .
i1 . . . 5.08 . .

i1 120 40 1 4.08 1.0 0.1
i1 . . . 4.12 . .
i1 . . . 5.03 . .
i1 . . . 3.08 . .
i1 . . . 3.12 . .
i1 . . . 4.03 . .
i1 . . . 5.08 . .
i1 . . . 5.12 . .
i1 . . . 6.03 . .

i1 140 40 1 5.12 1.0 0.1
i1 . . . 6.03 . .
i1 . . . 6.07 . .
i1 . . . 4.12 . .
i1 . . . 5.03 . .
i1 . . . 5.07 . .
i1 . . . 6.12 . .
i1 . . . 7.03 . .
i1 . . . 7.07 . .

i1 160 60 1 5.06 1.0 0.1
i1 . . . 5.10 . .
i1 . . . 6.01 . .
i1 . . . 4.06 . .
i1 . . . 4.10 . .
```


