Violencia orgánica

5:04 min stereo (2004)

Die konzeptuelle Idee dieses Stückes war eine Rhythmusgenerator zu programmieren. Anstatt ein Metrum mit periodischen Abstände zu benutzen, bauen sich die rhythmischen Strukturen aus Accelerandos und Diminuendos auf. Es gibt eine lange Liste von Abständen die linear ein Accelerando oder Diminuendo entwickelen (lista-1), der als Axe (wie ein Metrum) funktioniert. Zu diese Axe werden andere kürzere Listen (lista-2) addiert, die sich durch eigene Accelerandos und Diminuendos von der längeren Axe entfernen um mit ihrer Umkehrung wieder zurück zu kommen.

Beispiel:

Lista-1 (Axe) ---> (1 2 3 4 5 6 7 8 9 10)

Lista-2 =(1 2 3)
(Entfernung von der Axe) ---> (0 0 0 1 2 3 3 2 1 0)

Addition beider Listen ---> (1 2 3 5 7 9 10 10 10 10)

Um aus diese metrischen Werten rhythmische Bewegungen zu schaffen definierte ich verschiedenen Werte für die Lautstärke jeden einzelnen Ereignisses. Diese Werte kommen aus einen Algorithmus (Akzente) der verschiedene Amplitudenwerte aus verschiedenen Möglichkeitswerte herausgibt: % 20 für 0.1, % 30 für 0.3, % 30 für 0.5 und % 20 für 1.0.

Das Klangmaterial ist ein 4.2105 Sekunden langes Guitarrensample. Schnitte davon werden in 1000 Schritte gelesen. Die Dauer jedes Events verlängert sich linear von 0.1 bis 1.0 Sekunden. Der Zeitpunkt im Sample ab der die Events gelesen werden geht auch linear in 1000 Schritte von Anfang des Samples bis zur Sekunde 3.2105.

Das Ergebnis ist eine sehr bewegte Klangfläche die ich Collage_a für den Kanal 1 nenne, und Collage_b für Kanal 2. Beide Kanäle haben dieselbe Diminuendoaxe (Entrydelay von 0.05 bis 0.15 in 1000 Schritte), aber verschiedene Variationslisten, um durch Raumverteilung mehr Bewegung zu schaffen.

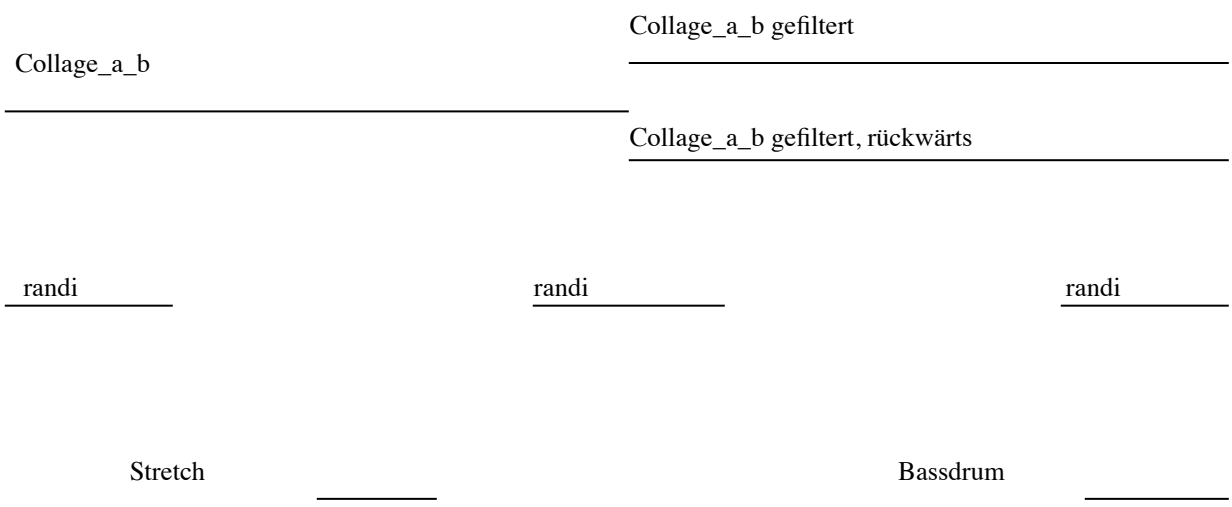
Die Klangfläche hat aufgrund des Materials einen sehr harmonischen Charakter. Ich wollte ein Geräusch als nächstes Material benutzen und entschied mich für gefiltertes Rauschen (randi.orc.) der mit randi in der Amplitude moduliert wird. An manchen Stellen des Stücks verlaufen die zwei Klangmaterialen gleichzeitig.

Formal gesehen besteht das Stück aus zwei symmetrischen (jeder 2.5 Minuten lang) Teilen. Der erste Teil ist der Prozess Collage_a und b, der zweite Teil ist eine sehr leise gefilterte Wiederholung, plus eine ebenfalls gefilterte Umkehrung von Collage_a und _b. Das Rauschen ist am Anfang, zwischen den beiden Teilen und am Ende zu hören.

Zwischen den ersten und zweiten Rauschen hört man ein Material das ich Stretch nenne. Im gesamten Prozess bewegt sich der Anfangspunkt des Samples. Jedes Event fängt ein bisschen später an. Diese Stretches sind einfache schnelle Wiederholungen (0.0555 Sekunden Abstand) von einzelne Events die aus der Liste Posiciones kommen.

Als letztes Material erscheint ganz am Ende ein sehr verhalltes Bassdrum-Sample, mit denselben Zeitpositionen wie in Collage_a.

Die Mischung wurde mit das Program Nuendo gemacht. Die Form des Stückes ist folgendes:



collage.orc File:

```
sr = 48000
kr = 48000
ksmps = 1
nchnls = 1

instr 1

a1 diskin  "/snd/icemserv/doerries/diktator_ohne_land/dracula/riff.aiff", 1.0, p5   ;Gitarrensample.
kamp   linseg 0, p3/4, 1, p3/4, 0.5, p3/2, 0                                        ;Hüllkurve für die Events.
out a1*kamp*p4

endin
```

collage_a.lisp:

```
(defun my-lin (idx start end)          ;Lineare Funktion für gleiche Abstände zwischen Start und Ziel, in.
  (+ (* (- end start) idx) start))     ;idx Werte.

;*************************************************************************

(defobject ritmo (i)                   ;Definiert eine neue Objektklasse
  (time duration amp-silencio skip)
  (:parameters time duration amp-silencio skip))

;**************************Zeitpositionierung der Events**************************

(defun listen (stw zw steps)           ;In der Anzahl von Steps geht man von Startwert(stw) zum Endwert(zw).
 (loop
          for x from 1 to steps
          collect (* x (+ stw (* (/ x steps)(- zw stw))))))

(defun mksegment (nullen varia)
          (append (listen (nth 0 nullen)(nth 1 nullen)(nth 2 nullen))          ;Nullen, keine Variation
                  (listen (nth 0 varia)(nth 1 varia)(nth 2 varia))             ;Variation.
                  (cdr (reverse (listen (nth 0 varia)(nth 1 varia)(nth 2 varia))))) ;Wiederkehrung zu
                                                                               ;Achse lista-1.
(defun mksegments (list)
          (cond ((null list) nil)
      (t (append (mksegment (nth 0 (car list))(nth 1 (car list))) ;mksegment mit liste Nullen (nth0) und
                  (mksegments (cdr list))                         ;Variationen (nth1).
                        ))))
```

```
(setf lista-1 (listen 0.05 0.15 1000))  ;Metrum diminuendo in 1000 Schritte, von 0.05 zu 0.15 Sekunden

(setf lista-2 (mksegments '(            ;Variation, Nullen. Geraden die von die Achse lista-1 sich
                                        ;entfehrnen und wiederkehren.
        ((0 0 150) (0.05 0.1 30))
        ((0 0 150) (0.1 0.02 20))       ;Nullen, Variation.
        ((0 0 400) (0.01 0.1 50))
        ((0 0 64) (0 0 20))

                )))


(setf lista-final (mapcar #'+ lista-1 lista-2))  ;Endliste. Paralleladdition von liste-1 (Metrum) und liste-2
                                                 ;(Variationen).

;**********************Wahrscheinlichkeiten für Amplituden**********************

(defun accente (a b c)                          ;Anzahl, Wahrscheinlichkeit, Amplitude.
  (loop for i in
  (loop for x from 1 to a
   collect (random 1.0))
        append (cond ((< i (nth 0 b)) (list (nth 0 c)))                 ;Wenn i <0.2 dann Amp=0.1
                ((< i (+ (nth 0 b) (nth 1 b))) (list (nth 1 c)))        ;i<0.5 >0.2dann Amp=0.3
                        ((< i (+ (nth 0 b) (nth 1 b) (nth 2 b))) (list (nth 2 c)));i<0.8>0.5 dann Amp=0.5
                        ((< i (+ (nth 0 b) (nth 1 b) (nth 2 b) (nth 3 b))) (list (nth 3 c)));i<1>.8 Amp=1
                        )))                                             ;
(setf acentos (accente 1000 '(0.2 0.3 0.3 0.2) '(0.1 0.3 0.5 1.0)))
;*********************************************
  (progn
   (defprocess acelerando ()
    (let ((e (new ritmo)))
     (process for count from 1 to 1000 do
              (let ((index (/ count 1000)))

                (sv e

                   ins             1
                   time             (nth (- count 1) lista-final)   ; Einsatzposition.
                   duration         (my-lin index 0.1 1.0)         ;von 0.1 bis 1.0 in 0.9/1000 Schritte.
                   amp-silencio     (nth (- count 1) acentos)      ;Lautstärke.

                   skip(my-lin index 0.0 3.2105)                   ;Skip time in Sample, von Anfang bis 3.2105.
                   )
                (output e)
                ))))

  (events (acelerando) “/users/doerries/kompo/violencia_4/collage_a.sco"))
```

collage_b.lisp:

```
(setf lista-2 (mksegments '(

        ((0 0 150) (0.1 0.02 20))          ;Das selbe Algo. Mit andere Werte für Kanal 2.
        ((0 0 150)(0.05 0.1 30))
        ((0 0 200) (0.01 0.1 30))
        ((0 0 244) (0 0 50))

                )))
```

collage_a.sco:

;p1	p2	p3	amp	skip
i1	0.050100002	0.1009	0.3	0.0032105
i1	0.1004	0.1018	0.3	0.006421
i1	0.1509	0.1027	0.3	0.0096315
i1	0.2016	0.1036	0.5	0.012842
i1	0.2525	0.1045	0.3	0.0160525
...				
i1	30.110899	0.41769996	0.3	1.1333065
i1	30.2316	0.4186	0.1	1.136517
i1	30.3525	0.41949996	0.1	1.1397275
i1	30.4736	0.4204	1.0	1.142938
i1	30.594902	0.4213	1.0	1.1461484
...				
i1	100.046104	0.8029	1.0	2.5074005
i1	100.25241	0.8038	0.3	2.510611
i1	100.45891	0.8047	0.3	2.5138214
i1	100.66561	0.8056	0.1	2.517032
i1	100.87251	0.8065	0.3	2.5202427

Usw.

collage_b.sco:

;p1	p2	p3	amp	skip
i1	0.050100002	0.1009	1.0	0.0032105
i1	0.1004	0.1018	1.0	0.006421
i1	0.1509	0.1027	0.5	0.0096315

```
i1      0.2016          0.1036      0.3     0.012842
i1       0.2525         0.1045      1.0     0.0160525
...
i1      30.095402       0.4132      0.5     1.117254
i1      30.296768       0.4141      0.3     1.1204646
i1      30.50167        0.415       0.5     1.123675
i1      30.710104       0.4159      0.1     1.1268855
i1      30.92207        0.4168      0.5     1.130096
...
i1      100.046104      0.8029      1.0     2.5074005
i1      100.25241       0.8038      1.0     2.510611
i1      100.45891       0.8047      0.3     2.5138214
i1      100.66561       0.8056      0.3     2.517032
i1      100.87251       0.8065      0.5     2.5202427
```

Usw.

collage_bajo.orc:

```
sr = 48000
kr = 48000
ksmps = 1
nchnls = 1

instr 1
a1 diskin  "/snd/icemserv/doerries/drum_wave/golpes/bombo.aiff", 1.0    ;Bassdrum Sample mit den
out a1*p4/5                                                             ;selben Algo, aber nur am
endin                                                                   ;Ende zu hören
```

Posiciones:

collage a

```
;p1     p2              p3          amp     skip
i1      108.1488        0.8344      0.3     2.619768
i1      112.21391       0.8461      0.5     2.6615045
i1      116.155205      0.856       0.5     2.69682
i1      119.3104        0.8632      1.0     2.722504
i1      120.016304      0.8749      0.1     2.7642405

i1      130.8741        0.9289      0.1     2.9568706
```

i1	133.93561	0.94060004	0.3	2.9986072
i1	137.75002	0.955	0.5	3.049975
i1	142.59001	0.97300005	0.1	3.114185
i1	142.83412	0.9739	0.3	3.1173956
i1	145.53241	0.9838	0.1	3.15271
i1	147.26212	0.9901	1.0	3.1751845
i1	149.00162	0.9964	0.3	3.197658
i1	150.0	1.0	1.0	3.2105

collage b

;p1	p2	p3	amp	skip
i1	110.17411	0.8461	0.5	2.6615045
i1	113.214905	0.8587	0.3	2.7064514
i1	118.07252	0.8785	0.5	2.7770824
i1	119.862915	0.88570005	1.0	2.8027666
i1	120.9884	0.8902	0.1	2.818819
i1	131.10841	0.9298	1.0	2.9600809
i1	133.93561	0.94060004	1.0	2.9986072
i1	139.4349	0.9613	0.3	3.0724485
i1	143.0784	0.9748	1.0	3.120606
i1	143.3229	0.9757	1.0	3.1238165
i1	146.0256	0.98560005	0.1	3.159132
i1	148.50362	0.9946	0.5	3.191237
i1	149.2509	0.99729997	0.3	3.2008684
i1	149.7501	0.9991	0.5	3.2072895

posiciones.lisp:

```
(defun my-lin (idx start end)  ;Lineare Funktion.
(+ (* (- end start) idx) start))

;=========================================

(defobject posiciones (i)                    ;Definiert eine neue Objektklasse
(time duration amp skip)
(:parameters time duration amp skip))
```

```
;=================================================================

(progn                                          ;Returns value of the last expression.
(defprocess play- ins ()
(let ((e (new posiciones)))                     ;New: allocates and initializes a new object.
(process for count from 1 to 100 do
(let ((normalized-idx (/ count 100)))
(sv e
ins         1
time        (my-lin normalized-idx 149.7501 155.3)   ;Letzter von posiciones.
duration      (my-lin normalized-idx  0.9991 0.9991)  ;Dauer Konstant.
amp             (my-lin normalized-idx 0.0 0.5)       ;Amplitude
skip        (my-lin normalized-idx 3.20728953 1.0)    ;Skip time in Sample von 3.20728953 bis 1.0.

)
(output e)
(wait       (my-lin normalized-idx 0.1 0.1))))))        ;Einsatzabstand 0.1 Sekunden.

(events (play- ins)"/users/doerries/kompo/violencia_4/stretch.sco")
)
```

posiciones.sco:

```
;letzte von posiciones
;p1     p2          p3          amp         skip
i1      149.8056    0.9991      0.005       3.1852167
i1      149.8611    0.9991      0.01        3.1631436n
i1      149.91661   0.9991      0.015       3.1410708
i1      149.9721    0.9991      0.02        3.1189978
i1      150.0276    0.9991      0.025       3.096925
i1      150.0831    0.9991      0.03        3.074852
i1      150.1386    0.9991      0.035       3.0527792
i1      150.19409   0.9991      0.04        3.0307064
i1      150.2496    0.9991      0.045       3.0086334
i1      150.3051    0.9991      0.05        2.9865606
usw...
```

randi.orc:

```
sr = 48000
kr = 48000
ksmps = 1
nchnls = 2

instr 1

a1 rand 1

krit1 linseg p7, p3, p9          ;Die verschiedenen Dichten von Bewegung
krit2 linseg p8, p3, p9
                                 ;Straight-line interpolations between each number
asig1 randi 5000, krit1          ; and the next (Band-limited noise. Krit specifys
asig2 randi 5000, krit2          ; rate of new random numbers).

kband linseg p5, p3, p6

afilt1   reson a1, p4, kband     ;Rekursives Bandpassfilter 2. Ordnung 12 dB/Oktave
afilt2   reson a1, p4, kband

abal1 balance  afilt1, a1        ;Balancierung gefiltertes mit original Signal
abal2 balance  afilt2, a1

kamp linseg 0, p3/8, 0.5, p3/8, 0.5, p3/2, 1.0, p3/4, 0      ;Hüllkurve

outs abal1*kamp*asig1, abal2*kamp*asig2                      ;Amp Modulation
endin
```

randi.sco:

;p1	p2	p3	Mittelfreq	Bandbreite-Anfang	Bandbreite-Ende	Rhytmlinks1	Rhytmrechts1	Rhytmende
i 1	0	50	8000	1	100	100	90	1
i 1	0	50	10000	1	100	100	90	1
i 1	0	50	12000	1	100	100	90	1
i 1	0	50	14000	1	100	100	90	1
i 1	0	50	16000	1	100	100	90	1

Usw.