

```
;; Conversations in Colour
;; initial studies No.1 - "lucid"
```

```
;; material
```

```
(setq t1 '(l u c i d))
```

Create repetitions of initial phrase.

```
(setq tlist (build-list t1 5))
```

```
; ((l u c i d) (l u c i d) (l u c i d) (l u c i d) (l u c i d))
```

Create lists of phrases consisting of 2-7 rests.

```
(setq gaps (gen-collect 0.71 6 :list '(build-list '= (get-random 2 7))))
```

```
; (=====) (---) (====) (=====) (-----) (=====)
```

```
gaps2 (gen-collect 0.717 6 :list '(build-list '= (get-random 2 7)))
```

```
(setq initial (flatten (symbol-interleave gaps tlist)))
```

Combine rests and repeating phrase.

```
; (===== l u c i d ===== l u c i d ===== l u c i d ===== . . .)
```

```
series1 (flatten (symbol-interleave gaps2 (mapcar 'reverse tlist))))
```

```
(setq textend1 (find-change (cf-noise-white (length (flatten gaps)) 1.0 .371 t1)))
```

Create a series of pitches driven from (l u c i d). White noise is used to redistribute values, while repeating notes are replaced with rests.

```
textend2 (find-change (cf-noise-white (length (flatten gaps)) 1.0 .3712 t1))
```

```
textend3 (find-change (cf-noise-white (length (flatten gaps)) 1.0 .3713 t1)))
```

```
(setq s1 (symbol-transpose -19 (fill-rest initial textend1))
```

Define raw material for the piece by making transpositions and scalings of textend1-3.

```
s2 (symbol-transpose 12 (fill-rest initial textend1))
```

```
s3 (symbol-transpose -17 (fill-rest initial textend2))
```

```
s4 (symbol-transpose 7 (fill-rest initial textend3))
```

```
s5 (symbol-transpose 5 (fill-rest initial textend1))
```

```
s6 (symbol-transpose -12 (fill-rest initial textend2))
```

```
s7 (symbol-transpose -7 (fill-rest series1 textend1))
```

```
s8 (symbol-scale '(-q c) (fill-rest series1 textend1))
```

```
s9 (symbol-scale '(x c) (fill-rest series1 textend1))
```

```
s10 (symbol-scale '(i -q) (fill-rest series1 textend2))
```

```
s11 (symbol-scale '(i x) (fill-rest series1 textend3))
```

```
)
```

```
(setq keys=s (append initial (symbol-mix initial s1) (symbol-mix initial s2)
```

Combine material in various configurations to create the keyboard part. Generally the other two instruments in the continuo trio (bass & mallet instrument) are derived from the keyboard part.

```
(symbol-mix initial s3 s4) (symbol-mix initial s5 s6)
```

```
(symbol-mix series1 s7) (symbol-mix series1 s8 s9)
```

```
(symbol-mix s10 s11)))
```

```
;; score
```

```
(def-tonality
```

```
default (activate-tonality (chromatic c 5))
```

```
elbass (activate-tonality (chromatic c 4))
```

```
)
```

```

(def-symbol
keybrd keys-s
mallet (filter-lowcut 'c (filter-delete '(u i) (get-symbols-of 'keybrd)))
elbass (append '(=) (delete '= (filter-highcut 'b keys-s)))
)

(def-length
default '(1/8)
elbass (get-timing '1/8 (filter-highcut 'b keys-s))
)

(def-zone
default (symbols-to-zone 'keybrd)
)

(def-velocity
default '(64)
)

(def-channel
keybrd 1
mallet 2
elbass 3
)

(def-program gm-sound-set
keybrd electric-piano-1
mallet vibraphone
elbass fretless-bass
)

(def-instrument-controller gm-controllers
(keybrd panning '((64)))
(mallet panning '((100)))
(elbass panning '((20)))
)

(def-tempo 140)

(compile-instrument-p "ccl:output:" "colour1"
keybrd
mallet
elbass
)

```

Create the keyboard part from the above material. The mallet part is based on the keyboard, excluding u and i and any note below c. Similarly the bass excludes notes above b.

```

;;; Conversations in Colour
;; initial studies No.2 - "serious"

;; functions

(defun rhy-maker (lis seed)
"derives rhythmic values from chord size"
(prog (out e11)
  (if seed (init-rnd seed))
  loop
  (let* ((initial diagnose-verbose)
        (diagnose-verbose nil))
    (cond ((null lis) (return out)))
    (setq e11 (symbol-melodize (list (car lis))))
    (setq out (append out
                       (list
                        (cond ((equal (length e11) 4) (pick-random
                                                       (gen-symbol-ratio nil
                                                         '(4 3 2 2)
                                                         '(1/4 1/2 1/2. 6/4))))
                              ((equal (length e11) 2) '1/16)
                              ((equal (length e11) 3) (pick-random '(1/8 1/4 1/4)))
                              (t '1/8))))))
    (setq diagnose-verbose initial))
  (setq lis (cdr lis))
  (go loop)))

; material

(setq t2 '(s e r i o u s))

(setq t2s (gen-process '(symbol-scale x y) '(((-x -d) (-t a) (-j d) (-c i) (e s)) t2)
  ; (-g -x -h -s -k -d -g -c -t -e -o -h a -c c -j b -g -c d c h -c q a e i h q e p i n s q)
  t2sm (symbol-transpose -5 (filter-lowcut '-c (symbol-select-skip 10 t2s 0.1)))
  ; ( = = = = = = = = -f -h -d = -e = -h -c = c -h b -f -b d = = -b k d i n l)
  t2sb (filter-highcut '-c (symbol-select-skip 12 t2s 0.1)))

(setq ty (symbol-bundle '(2) t2))

(setq tyx (find-change (octavise (gen-variants-rndt 0.32178 6 -5 ty) -12))
  tyxm (symbol-transpose -17 (symbol-select-skip (/ (length tyx) 4)
                                                (symbol-inversion 's tyx) 0.12))
  tyxb (symbol-transpose -12 (octavise (symbol-select-cut 10
                                                       (symbol-demix 2 tyx) 0.12) -7)))

```

Create differently scaled instances of (s e r i o u s).

Use symbol-select-skip to add random rests to material for mallet part - then filter & transpose.

Create variations on bundled notes (s e r i o u s).

```

(setq tc (symbol-bundle '(3 4) t2))

(setq tcv (gen-variants-rndt 0.328 6 -7 tc)
      tcvm (octavise (symbol-mix (symbol-demix 1 tcv) (symbol-demix 2 tcv)) 7)
      tcvb (symbol-transpose -12 (octavise (symbol-mix (symbol-demix 4 tcv)) -12)))

(setq tcx (expand+transpose (gen-random 0.1 (* (length tcv) 2) (g-integer -12 7)) tcv)
      tcxm (filter-lowcut '-h (octavise
                              (symbol-mix (symbol-demix 1 tcx) (symbol-demix 2 tcx)) 7))
      tcxb (symbol-scale '(b -w) (octavise (symbol-mix (symbol-demix 4 tcx)) -12)))

(setq cda (flatten (symbol-interleave
                  (gen-repeat 3
                    (subseq tcx (- (length tcx) 2) (length tcx)))
                  (symbol-divide 7 nil nil (reverse t2s))))
      cdam (symbol-transpose -7
                (filter-lowcut '-c
                  (symbol-select-cut 10
                    (symbol-demix 1 cda) 0.1 :end)))
      cdab (symbol-transpose 5
                (filter-highcut '-c
                  (symbol-select-cut 10
                    (symbol-demix 1 cda) 0.1 ))))

;; score

(def-tonality
  default (activate-tonality (chromatic c 5))
  elbass (activate-tonality (chromatic c 4))
)

(def-symbol
  keybrd (append t2s tyx tcv tcx cda)
  mallet (append t2sm tyxm tcvm tcxm cdam)
  elbass (append t2sb tyxb tcvb tcxb cdab)
)

(def-length
  default (rhy-maker (get-symbols-of 'keybrd) 0.1)
)

(def-zone
  default (make-zone (get-lengths-of 'keybrd))
)

```

Use symbol-demix to extract notes from chordal material and symbol-mix to combine extracted notes into new chords.

Call rhy-maker function (defined above) to create note-lengths from chord sizes.

```
(def-velocity
  default '(64)
)

(def-channel
  keybrd 1
  mallet 2
  elbass 3
)

(def-program gm-sound-set
  keybrd electric-piano-1
  mallet vibraphone
  elbass fretless-bass
)

(def-instrument-controller gm-controllers
  (keybrd panning '((64)))
  (mallet panning '((100)))
  (elbass panning '((20)))
)

(def-tempo 70)

(compile-instrument-p "ccl/output:" "colour2"
  keybrd
  mallet
  elbass
)
```

```

;;; Conversations in Colour
;; initial studies No.3 - "mighty"

;; functions

(defun rhy-maker-2 (lis seed)
  "derives rhythmic values from chord size"
  (prog (out ell)
    (if seed (init-rnd seed))
    loop
    (let* ((initial diagnose-verbose)
          (diagnose-verbose nil))
      (cond ((null lis) (return out)))
      (setq ell (symbol-melodize (list (car lis))))
      (setq out (append out
                        (list
                         (cond ((equal (length ell) 4) (pick-random
                                                         (gen-symbol-ratio nil
                                                           '(4 3 2 1)
                                                           '(1/4 1/8 1/16 1/2))))
                               ((equal (length ell) 2) (pick-random '(1/16 1/16 1/8)))
                               ((equal (length ell) 3) (pick-random '(1/8 1/8 1/4)))
                               (t '1/4))))
            (setq diagnose-verbose initial))
      (setq lis (cdr lis))
      (go loop)))

;; material

(setq t3 '(m i g h t y)
      t3a (symbol-transpose -4 t3)
      t3b (symbol-transpose -3 t3a))

(setq tc1 (symbol-transpose 12 '(mge-h))
      tc2 (symbol-transpose 12 '(ic-d-e))
      tc3 (symbol-transpose 12 '(fe-e-m)))

(setq inter '(= -e i j c a f e -g -d a g -f )
            inter1 (symbol-bundle 2 inter)
            inter2 (cf-sin '3 .5 36 inter))

```

Modulate the list inter to a sine wave to create self-similar sequence.

```
(setq t3c (flatten (g-chord .23 2 5 -7 5 (gen-repeat 12 (list t3))))
      t3ca (flatten (g-chord .25 2 5 -7 7 (gen-repeat 15 (list t3a))))
      t3cb (flatten (g-chord .27 2 5 -7 12 (gen-repeat 18 (list t3b)))))
```

g-chord will move through the list creating chords of between 2-5 notes and transposed between -7 and 5 semitones (e.g. from a fifth below to a fourth above). There will be some single notes in the resulting sequence since g-chord works on each zone in the list in turn.

```
(setq t3c-r (strip-singles (make-repeats t3c '(0 1 2 3) 0.17))
      t3ca-r (strip-singles (make-repeats t3ca '(0 1 2 3) 0.19))
      t3cb-r (strip-singles (make-repeats t3cb '(0 1 2 3) 0.21)))
```

Create a stream of repetitions of the above material, replacing any single notes with rests.

```
(setq rhy1 (rhy-maker-2 t3c-r 0.1)
      rhy2 (rhy-maker-2 t3ca-r 0.12)
      rhy3 (rhy-maker-2 t3cb-r 0.13))
```

```
(setq r1 '((1/4) (1/2))
      r2 '((1/8) (1/2.))
      r3 '((1/16) (1/1))
      r4 '((1/8))
      r5 '((1/2 1/2. 1/1))
      r6 '((1/4))
      r7 '((1/16)))
```

```
(setq d1 '((70))
      d2 '((110))
      d3 (list (vector-to-list (vector-round 50 110 (gen-noise-white (length t3c-r)))))
      d4 '((80))
      d5 '((120))
      d6 (list (vector-to-list (vector-round 40 120 (gen-noise-white (length t3ca-r)))))
      d7 '((90))
      d8 '((125))
      d9 (list (vector-to-list (vector-round 40 120 (gen-noise-white (length t3cb-r)))))
      d10 '((100))
      d11 '((110 120 125))
      d12 (list (gen-cresc 60 120 (length inter1)))
      d13 (list (gen-cresc 60 127 (length (gen-palindrome inter))))
      d13a (list (gen-dim 127 60 (length inter2)))
      d14 (list (gen-cresc 60 127 (length (but-last t3)))))
```

```
(setq df1 '((1/32) (1/2))
      df2 '((1/32) (1/2.))
      df3 '((1/32) (1/1))
      df4 '((1/32)))
```

```
;; score
```

```
(def-tonality
  default (activate-tonality (chromatic c 5))
  mallet (activate-tonality (chromatic c 5))
  elbass (activate-tonality (chromatic c 3))
)
```

```
(def-symbol
  keybrd (list (append '(=) (but-last t3)) tc1 t3c-r
              (append '(=) (but-last t3a)) tc2 t3ca-r
              (append '(=) (but-last t3b)) tc3 t3cb-r
              (append '(=) (but-last t3)) (append tc1 tc2 tc3)
              inter1 (gen-palindrome inter) inter2
              (symbol-transpose -12 (append '(=)(but-last t3b)))
              (symbol-transpose -12 (reverse (append tc1 tc2 tc3))) (reverse inter1)
              (symbol-transpose -24 tc1)
              (symbol-transpose -24 (reverse (but-last t3))))
```

Create keyboard part from above defined material, cf. bar 1 of the score, which corresponds to (= m i g h t), or (append '(=) (but-last t3))

```
  mallet (mapcar 'find-anacrusis
                (do-section (append '(= = x = = x = = x (build-list '= 10)))
                            '(randomize-octaves 0.1 nil -1 0 x)
                            (mapcar 'symbol-mix
                                    (do-section :all '(symbol-demix 3 x )
                                                (get-symbols-of 'keybrd))
                                    (do-section (append '(x x = x x) (build-list '= 14))
                                                '(symbol-transpose -24 x)
                                                (do-section (append '(build-list '= 17) '(x x))
                                                            '(symbol-transpose 12 x)
                                                            (do-section :all '(symbol-demix 1 x )
                                                                (get-symbols-of 'keybrd)))))))
                elbass (mapcar 'find-change (do-section (append '(x x = x x) (build-list '= 14))
                                                         '(symbol-transpose -12 x )
                                                         (do-section (append (build-list '= 14) '(x x x x))
                                                                     '(symbol-transpose 12 x )
                                                                     (do-section :all
                                                            (symbol-demix 1 x :sort)
                                                            (get-symbols-of 'keybrd))))))
```

Create the mallet part from the keyboard part. Note the use of symbol-mix/ symbol-demix to create areas of differing complexity.

```
)
```

```

(def-length
default (eval-list '(r1 (list rhy1)
r2 (list rhy2)
r3 (list rhy3)
r4 r5
r6 r7 r7
(list (car r2))
(reverse r5) r4
(reverse r1)))

)

(def-duration
default (eval-list '(df1 (list rhy1)
df2 (list rhy2)
df3 (list rhy3)
df4 r5
r6 df4 df4
df4
(reverse r5) r4
(reverse df1)))

)

(def-zone
default (make-zone-list (get-symbols-of 'keybrd) (get-lengths-of 'keybrd))
)

(def-velocity
default (eval-list '(d1 d2
d3 d4
d5 d6
d7 d8
d9 d10
d11 d12
d13 d13a d10
(reverse d11)
(reverse d12) d2 d14))

)

```

```
(def-channel
keybrd 1
mallet 2
elbass 3

)

(def-program gm-sound-set
keybrd electric-piano-1
mallet vibraphone
elbass fretless-bass
)

(def-instrument-controller gm-controllers
(keybrd panning '((0)))
(mallet panning '((90)))
(elbass panning '((30)))
)

(def-tempo 105)

(compile-instrument-p "ccl/output:" "Colour3"
keybrd
mallet
elbass
)
```

```
;;; Conversations in Colour
;; initial studies No.4 - "serene"
```

```
;; material
```

```
(setq t4 '(s e r e n e))
```

```
(setq t4i (symbol-inversion 'e t4))
```

```
(setq t4v
 '(s e r e n e -k e -j e -f -t -f -s -f -o j -f i -f e n ))
```

This material was derived from inversions and transpositions of (s e r e n e)

```
(setq process (gen-palindrome
 (append t4v (gen-process '(symbol-transpose x y) '(9 18) t4v)))
 ; (s e r e n e -k e -j e -f -t -f -s -f -o j -f i . . . i -f j -o -f -s -f -t -f e -j e -k e n e r e s)
```

```
(seq :process-a (symbol-scale '(-i q) process))
(seq :process-b (symbol-bundle 2 (symbol-scale '(-h r) process)))
```

Note the use of seq, this allows sections of a long list to be picked sequentially, returning to the start once all symbols have been returned.

```
(seq :process-a :reset)
(seq :process-b :reset)
```

```
(def-neuron bline
(all-in 1 '(r e) -12 12) (gen-seq :process-a 4)
(all-in 1 '(e k) -12 12) (gen-seq :process-a 4)
(all-in 1 '(e n) -12 12) (gen-seq :process-a 4)
(otherwise '=))
```

Define a neural network to create the bass line. E.g. when r e occurs in any transposition, replace it with four notes from the process-a sequence, else replace it with a rest.

```
(setq mat (run-neuron 'bline process))
```

```
(def-neuron bline-r
(in 1 '=) '1/8
(otherwise '1/32))
```

Creat note lengths for the bass part using a neural network. Where a rest occurs, set the duration to 1/8, else use 1/32.

```
(setq b-rhy (run-neuron 'bline-r mat))
```

```
(def-neuron mline
(all-in 1 '(e r) -12 12) (gen-seq :process-b 2)
(all-in 1 '(-k e) -12 12) (gen-seq :process-b 2)
(all-in 1 '(-f e) -12 12) (gen-seq :process-b 2)
(otherwise '=))
```

```
(setq mat1 (run-neuron 'mline process))
```

```
(def-neuron mline-r
  (in 1 '=) '1/8
  (otherwise '1/16))

(setq m-rhy (run-neuron 'mline-r mat1))
```

```
(def-tonality
  keybrd (activate-tonality (chromatic c 5))
  mallet (activate-tonality (chromatic c 5))
  elbass (activate-tonality (chromatic c 3))
  )
```

```
(def-symbol
  keybrd process
  mallet mat1
  elbass mat
  )
```

```
(def-length
  keybrd '(1/8)
  mallet m-rhy
  elbass b-rhy
  )
```

Set note lengths in keyboard part exclusively to 1/8, use the patterns developed by the above neurons for the mallet and bass parts.

```
(def-zone
  default (symbols-to-zone 'keybrd)
  )
```

```
(def-velocity
  keybrd (gen-cresc-dim 25 50 (length process))
  mallet (vector-round 20 45 (gen-noise-white (length mat1)))
  elbass (vector-round 15 35 (gen-noise-white (length mat)))
  )
```

```
(def-channel
  keybrd 1
  mallet 2
  elbass 3
  )
```

```
(def-program gm-sound-set
keybrd electric-piano-1
mallet vibraphone
elbass electric-bass-finger
)
```

```
(def-tempo 35)
```

```
(compile-instrument-p "ccl;output:" "Colour4"
keybrd
mallet
elbass
)
```

```
;;; Conversations in Colour
;; initial studies No.5 - "melancholic"
```

```
;; material
```

```
(setq t5 '(m e l a n c h o l i c))
```

```
(setq t5c '(me la me la nc ho nc ho)
      t5c1 '(la me nc ho)
      t5s '(= -e -b c i l)
      t5r '(= -e -i -m -q -u )
      t5rb (symbol-transpose 12 '(= -e -i -m -q -u ))
      t5sx (symbol-mix t5s t5r))
```

```
(setq t5cb (symbol-transpose -12 '(m e l a n c h o))
      t5sb '(l i c)
      t5sbl '(-l -i -c a c))
```

```
(setq t5c-1 (append '(me la) (gen-process
                          '(symbol-scale x y) '((m -b)(e -h) (-b -l)) '(me la)) '(=))
      t5c-2 (append '(nc ho )
                    (gen-process '(symbol-scale x y) '((m -b)(e -h) (-b -l)) '(nc ho )) '(=))
      t5cm-1 (filter-lowcut '-h t5c-1)
      t5cm-2 (filter-lowcut '-h t5c-2)
      t5cb-1 (symbol-demix 2 t5c-1 )
      t5cb-2 (symbol-demix 2 t5c-2 ))
```

```
(setq t5c-3 (symbol-mix t5c (symbol-transpose -12 (reverse t5c)))
      t5c-4 (symbol-mix (reverse t5c1) (symbol-transpose -12 t5c1))
      t5c-5 (symbol-mix (symbol-transpose -1 (reverse t5c1)) (symbol-transpose -11 t5c1))
      t5cb-3 (symbol-demix 4 t5c-3)
      t5cb-4 (symbol-demix 4 t5c-4)
      t5cb-5 (symbol-demix 4 t5c-5))
```

```
(setq t5c-6 (append '(=) (list (first t5c-5)))
      t5cm-6 '(= gna)
      t5cm-7 '(= -b-ed)
      t5cm-5 '(gna mb-h ldc b-ed)
      t5c-7 (append '(=) (last t5c-5))
      t5cb-6 '(= = -l -l) ; (symbol-demix 4 t5c-6 )
      t5cb-7 '(= = -d -d) ; (symbol-demix 4 t5c-7 ))
```

Create sections of musical material for the keyboard based on scaling, transposition, filtering, reversal etc of symbol lists. Use filter and demix to create mallet and bass parts.

```
;; score
```

```
(def-tonality  
keybrd (activate-tonality (chromatic c 5))  
mallet (activate-tonality (chromatic c 5))  
elbass (activate-tonality (chromatic c 3))  
)
```

```
(def-symbol  
keybrd (list t5c t5s t5c-1 t5c-2 (append (but-last (gen-palindrome t5s)))  
t5c-3 t5sx t5c-4 (gen-palindrome t5c-5))  
t5c-6 t5c-6 (nthcdr 1 t5c-5)  
t5c-7 t5c-7 t5c-4 (nthcdr 2 t5c-5))
```

Use the musical material generated above to create the final parts for the keyboard. Note repeated sections and use of nthcdr to repeat only certain notes in the sections.

```
mallet (list t5c t5s t5cm-1 t5cm-2 (append (but-last (gen-palindrome t5s)))  
t5c t5s (reverse t5c1) (gen-palindrome (symbol-transpose -1 (reverse t5c1)))  
t5cm-6 t5cm-6 (nthcdr 1 t5cm-5)  
t5cm-7 t5cm-7 (reverse t5c1)(nthcdr 2 t5cm-5))
```

```
elbass (list t5cb t5sb t5cb-1 t5cb-2 t5sbl  
t5cb-3 t5rb t5cb-4 (gen-palindrome t5cb-5)  
t5cb-6 t5cb-6 (nthcdr 1 t5cb-5)  
t5cb-7 t5cb-7 t5cb-4 (nthcdr 2 t5cb-5))
```

```
)
```

```
(def-length  
keybrd '((1/4 1/2 1/4 1/2 1/4 1/2 1/4 1/2) (1/8) (1/4) (1/4) (1/8)  
(1/4 1/2 1/4 1/2 1/4 1/2 1/4 1/2) (1/8)  
(1/4 1/2 1/4 1/2) (1/4 1/4 1/4 1/1 1/4 1/4 1/4)  
(1/4 1/2) (1/4 1/2.) (1/4 1/4 1/1)  
(1/4 1/2.)(1/4 1/2) (1/4 1/2 1/2 1/2.) (1/2 -1/4 1/1))
```

```
mallet (get-lengths-of 'keybrd)
```

```
elbass '((-1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8) (1/4) (1/4) (1/4) (1/4)  
(-1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8) (1/8)  
(-1/2 1/8 1/8 1/8 1/8 -1/2) (1/4 1/4 1/4 1/1 1/4 1/4 1/4)  
(1/4 1/4 1/4) (1/4 1/4 1/4 1/4) (1/4 1/4 1/1)  
(1/4 1/4 1/4 1/4)(1/4 1/4 1/4) (1/4 1/2 1/2 1/2.) (1/2 -1/4 1/1))
```

```
)
```

```

(def-duration
keybrd '((1/8 1/2 1/8 1/2 1/8 1/2 1/8 1/2) (1/8) (1/8 1/4) (1/8 1/4) (1/8)
        (1/8 1/2 1/8 1/2 1/8 1/2 1/8 1/2) (1/8)
        (1/8 1/2 1/8 1/2) (1/8 1/4 1/4 1/1 1/4 1/4 1/8)
        (1/4 1/2) (1/4 1/2.) (1/8 1/4 1/1)
        (1/4 1/2.) (1/4 1/2) (1/8 1/2 1/2 1/2.) (1/4 -1/4 1/1))

mallet (get-durations-of 'keybrd)

elbass '((-1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8) (1/4) (1/4) (1/4) (1/4)
        (-1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8 -1/2 1/8 1/8) (1/8)
        (-1/2 1/8 1/8 1/8 1/8 -1/2) (1/8 1/4 1/4 1/1 1/4 1/4 1/8)
        (1/4 1/8 1/8) (1/4 1/8 1/8 1/8) (1/8 1/4 1/1)
        (1/4 1/8 1/8 1/8)(1/4 1/8 1/8) (1/8 1/2 1/2 1/2.) (1/4 -1/4 1/1))
)

(def-zone
default (make-zone-list (get-symbols-of 'keybrd) (get-lengths-of 'keybrd))
)

(def-velocity
default '((54 80) (40 50 60 70 80 85) (90 85 85 80 80 75 75 70 70) (80 75 75 70 70 65 65 60 60) (40 45 50 55 60 65 70 75 80 85)
        (54 80) (40 50 60 70 80 85) (54 80) (65 70 80 90 80 70 65) (70) (80)(75 70 65) (50) (45) (50 55 60 65) (70 50)))

(def-channel
keybrd 1
mallet 2
elbass 3
)

(def-program gm-sound-set
keybrd electric-piano-1
mallet vibraphone
elbass fretless-bass
)

(def-tempo 70)

(compile-instrument-p "ccl:output:" "Colour5"
keybrd
mallet
elbass
)

```