

Music 680, Fall 2007: Special Topics in Music - Compositional Algorithms
Class 4: October 1, 2007 - canon I: Nancarrow

Nancarrow

- as a textbook of different canonic and algorithmic techniques
 - and Gann's chapters as a catalog of these techniques...
- sharing with the serialists a concern for music from first principles
 - in part because of composition for new instruments / apart from performers?
 - what about his relative geographic isolation in Mexico City?
 - canon as a means of "restricting variation" in non-rhythmic parameters
- particular techniques
 - isorhythm
 - using multiple tempi inside a single *talea* (Study 6)
 - tempo canon
 - accelerating/ decelerating tempos (Studies 21, 22, 27)
 - duration dependent on both acceleration rate and initial durations
 - complex proportions (incl. irrational relationships)
 - canonic forms
 - vertical and horizontal axes of symmetry
 - convergence points
 - initial/ central/ golden section/ final
 - and also outside of a work's boundaries...
 - applying non-convergent effects n times for n voices to create unison events
 - sound-mass canon
 - in which issues of counterpoint "fade from relevance" (Gann)
- psychoacoustics and streaming
 - Nancarrow's concern for perceptibility
 - registral, textural, rhythmic alignments
 - incl. use of register to disintegrate a single canonic voice into multiple perceptual streams
 - first making polytempo audible
 - then concentrating on formal effect
 - which means relationships between form and content
 - consider use of longer phrases farther away from convergence points
 - and briefer gestures closer in
 - but also emphasizing sound masses rather than imitative relationships
 - omission of convergence points in Study 37

canon in Common Lisp

also relationships between voices/layers? (contrapuntal logic)

```
(defparameter pitch-list (list 60 61 63 67 64 66 70 69 72))
(defparameter rhythm-list (list 1.5 1.125 1.375 1.0 1.625 1.875 2.0))
(defparameter dynamics-list (list 0.5 0.75 1.0 0.33 0.66))

(defun make-note (start pitch length dynamic)
  (new midi
    :time start
    :keynum pitch
    :duration length
    :amplitude dynamic))

(defun rotate-left (input-list)
```



```

                12 0 pitch-list rhythm-list dynamics-list)
"test.mid")
|#

(defparameter upper-voice-set1
  (list 71 72 80 79 76 77 84 75
        70 73 72 69 65 78 77 80
        82 83 88 76 81 86 85 83
        82 78 81 74 77 76 67 75
        70 80 90 89 92 91 84 85
        82 83 80 78 81 69 70 73
        66 65 87 86 84 72))

(defparameter lower-voice-set1
  (list 35 36 44 43 40 41 48 39
        34 37 36 33 29 42 41 44
        46 47 52 40 45 50 49 47
        46 42 45 38 41 40 31 39
        34 44 54 53 56 55 48 49
        46 47 44 40 45 33 34 37
        42 41 51 50 48 36))

(defparameter transpositions
  (list 0 1 -5 1 5 1 -5 1 5
        1 1 1 -7 -1 5 -1 -5
        -1 5 -1 -5 -1 11 -1 -5
        -4 8 7 -2 -7 -4 -4 11
        2 6 -10 -3 10 -9 8 9
        -10 3 4 -1 -2 5 -10 9
        -2 -5 -4 -3 5))

(defun make-voice-pitches (pitch-set transposition-series)
  (if (or (null pitch-set)
          (null transposition-series))
      nil
      (append (transpose pitch-set (car transposition-series))
              (make-voice-pitches (cdr pitch-set) (cdr transposition-series)))))

(defun make-voice (pitches start-time start-duration duration-step)
  (if (null pitches)
      nil
      (cons (make-note start-time (car pitches) start-duration 1.0)
            (make-voice (cdr pitches)
                        (+ start-time start-duration)
                        (+ start-duration duration-step)
                        duration-step))))

(defun calculate-step (number-events start-bps end-bps)
  (/ (- (/ 1 end-bps) (/ 1 start-bps))
     (- number-events 1)))

; upper voice decelerando from 37 bps (0.0270 dur) to 2.33 bps (0.42918 dur) over 1485 pitches
; produces a duration step of .000271

; (make-voice (make-voice-pitches upper-voice-set1 transpositions) 2 0.027 0.000271)

#| (events (make-voice (make-voice-pitches upper-voice-set1 transpositions) 2 0.027 0.000271)
"test.mid") |#

; lower voice accelerando from 3.5 bps to 111 bps

#| (events (make-voice (make-voice-pitches lower-voice-set1 transpositions) 0 0.28571
-0.000186459) "test.mid") |#

```

```
(defun make-acceleration (pitches start-time start-duration tempo-percentage)
  (if (null pitches)
      nil
      (cons (make-note start-time (car pitches) start-duration 1.0)
            (make-acceleration (cdr pitches)
                              (+ start-time start-duration)
                              (* start-duration (/ 1 tempo-percentage))
                              tempo-percentage))))

#| (events (make-acceleration (make-voice-pitches lower-voice-set1
transpositions) 0 0.28571 1.01) "test.mid") |#
```