

Music 680, Fall 2007: Special Topics in Music - Compositional Algorithms
Class 13: December 10, 2007 - Cage II: later works

artistic process across media

printmaking

I Ching determination of mark types, locations, activities
(examples from www.crownpoint.com)
and the use of fire / smoke as uncontrollable elements

film

"without regard for illumination" as a byproduct of *Europa* theatrical lighting experiences

composition

the number pieces: separation of structure and content
(and *Europas* as intensifying that separation through borrowed content)

poetry

mesostics and the "writing through" process

unifying activities

opera, film, music, and "circuses" all designed according to time-bracket structure
and visual art uses I Ching determined spatial location as an analogy to temporal location
the only exception is the literary mesostics, which don't quite function in the same way

... and a unifying aesthetic: sparseness or abundance - but never in between

"Cage and the Computer"

the computer as an agent of existing processes rather than as an avenue to new techniques
and the possibility of doing projects on an even larger scale than previously
the labor-saving device making time available for new laborious projects

number pieces in CM

```
; Cage "number piece" process  
; lcb 12.06.2007
```

```
; first, make the sectional proportions of the piece  
; random numbers between 0 and 1 that govern the length of individual sections  
; (with the beginning of the piece defined as zero, and the end as one)  
; always include a zero in the list so that the piece begins at the beginning
```

```
(defun make-proportions (number-of-sections)  
  (if (<= number-of-sections 1)  
      (list 0)  
      (cons (random 1.0)  
            (make-proportions (- number-of-sections 1)))))
```

```
; second, sort the proportions in ascending order  
; (we could do this by hand, but common lisp provides a (sort) function  
; so we'll use that....
```

```
(defun sort-proportions (proportion-list)  
  (sort proportion-list #'<=))
```

```

; third, fit the proportions to the actual duration of the piece
; this creates an actual set of time brackets for one voice

(defun scale-proportions (proportion-list start total-duration)
  (if proportion-list
    (cons (+ (* (car proportion-list) total-duration) start)
          (scale-proportions (cdr proportion-list) start total-duration))
    (list (+ start total-duration))))

; generate the actual MIDI for a voice by specifying register and a set of time brackets

(defun number-voice (lowest-pitch highest-pitch time-brackets)
  (if (>= (length time-brackets) 2)
    (let* ((bracket-start (car time-brackets))
          (bracket-end (cadr time-brackets))
          (event-start (+ bracket-start
                          (random (- bracket-end bracket-start))))
          (event-duration (random (- bracket-end event-start))))
      (cons (new midi
                :time event-start
                :keynum (+ (random (- highest-pitch lowest-pitch))
                           lowest-pitch)
                :amplitude (random 1.0)
                :duration event-duration)
            (number-voice lowest-pitch highest-pitch (cdr time-brackets))))))

; use this function to create a single voice using time-brackets

(defun make-number-voice (lowest-pitch highest-pitch events start duration)
  (number-voice lowest-pitch
                highest-pitch
                (scale-proportions
                 (sort-proportions
                  (make-proportions events)) start duration)))

; (events (make-number-voice 70 100 15 0 50) "test.mid")

; and finally, the function that creates a number of voices worth of time-bracket music
; voice-registers is a list of lists, where each sublist is a pair of values:
; the low and high boundary for the register of each voice
; make-number piece makes as many voices as there are registers specified

(defun make-number-piece (voice-registers events-per-voice start duration)
  (if voice-registers
    (append (make-number-voice (caar voice-registers)
                              (cadar voice-registers)
                              events-per-voice
                              start
                              duration)
            (make-number-piece (cdr voice-registers) events-per-voice start duration))))

; (events (make-number-piece '((40 70) (50 80) (60 90) (70 100) (80 110)) 15 0 50) "test.mid")

```