MORE ALGORITHMIC COMPOSITION

Probability Distributions, Random Walks, Grids, Masks, and more

Types of Machine-Aided Composition

- Completely directed by composer
  - Notation packages
  - Cut and Paste
  - Editing macros
- Algorithmic Compositions
  - Procedures + random numbers
- Artificial Intelligence
  - Music models
  - Models of composition
  - Machine learning
  - Search

Types of Machine-Aided Composition

- Completely directed by composer
  - Notation packages
  - Cut and Paste
  - Editing macros
- Algorithmic Compositions
  - Procedures + random numbers
- Artificial Intelligence
  - Music models
  - Models of composition
  - Machine learning
  - Search

Greatest "High" Culture Impact

Types of Machine-Aided Composition

- Completely directed by composer
  - Notation packages
  - Cut and Paste
  - Editing macros
- Algorithmic Compositions
  - Procedures + random numbers
- Artificial Intelligence
  - Music models
  - Models of composition
  - Machine learning
  - Search

Greatest "High" Culture Impact

Increasing "Pop" Culture Impact

Techniques – Tricks of the Trade

- Rhythm using Negative Exponential distribution
- Melody using random walk
- Markov algorithm
- Rhythmic pattern generation
- Melodic transformations & serialism
- Fractals
- Grammars
- Pitch and Rhythm grids
- Tendency masks

Scores and Score Manipulation

- SCORE-BEGIN-END is not synthesized:
  (score-begin-end
  <start-time>
  <end-time>)
- Pitch lists are expanded as chords

```
set myscore =
 {{0 1 {score-begin-end 0 2}}
  {0 1 {tpt :pitch {64 67 72} :vel 100}}
  {1 1 {tbn :pitch 48 :vel 80}}}

function tpt(pitch: 60, vel: 100)
   return trumpet(pitch, vel)

eval score-play(myscore)
```


Scores and Score Manipulation (2)

score-shift(score, offset)
score-stretch(score, factor)
score-transpose(score, keyword, amount)
score-scale(score, keyword, amount)
score-sustain(score, factor)
score-voice(score, replacement-list)
score-merge(score1, score2, …)
score-adjacent-events(score, function)
score-apply(score, function)
score-stretch-to-length(score, length)

score-append(score1, score2, …)
score-select(score, predicate)
score-filter-length(score, cutoff)
score-repeat(score, n)
score-filter-overlap(score)
score-print(score)
score-play(score)
score-last-index-of(score, function)
score-randomize-start(score amt)
score-sort(score, [copy-flag])

Scores and Score Manipulation (3)

- All score functions take some optional keyword parameters:
  - :from-index *i*
  - :to-index *i*
  - :from-time *seconds*
  - :to-time *seconds*
- Score functions construct new scores
- Standard MIDI File I/O:
  - score-read-smf(*filename*)
  - score-write-smf(*score, filename*)

Workspaces

- How do you save score data?

```
set my-score = {… score data …}
set new-score = score-transpose(my-score,
                                :pitch 3)
exec add-to-workspace(quote(my-score))
exec describe(my-score, "original data")
exec add-to-workspace(quote(new-score))
exec describe(new-score, "transposed version")
exec save-workspace()
```

- Later, you can just load workspace.lsp to restore everything. The variable names are in `*workspace*`.

The Negative Exponential Distribution

- "Random" is interesting(!)
- What does it mean to be random in time?
  - Uniform random interval between events?
  - Gaussian?
  - Some other distribution?
- Examples from real world:
  - Atomic decay
  - Sequence of uncorrelated events (yellow cars driving by)

Negative Exponential Distribution (2)

- The inter-arrival time has a negative exponential distribution: longer and longer intervals are less and less likely
- Equivalently: in each very small interval of time, generate an event with some small probability P = density * interval duration
- Equivalently: generate events at times that are uniformly random across total duration.

```
score-gen(score-len: 50,
          time: real-random(0, 12.5),
          dur: 1,
          name: quote(s-pop-kwp),
          lo: 800, hi: 1200)
```


Probability distributions in Nyquist

- load “distributions”
- See Distributions, probability in Nyquist index

```
begin
  with ne-score
  loop
    for i below 30
    for now = 0 then now + exponential-dist(1.0)
    set ne-score @= list(now, 1, {s-pop 800 1200})
  end
  return reverse(ne-score)
end
```

score is in reverse order, so rather than sorting, we can just reverse it