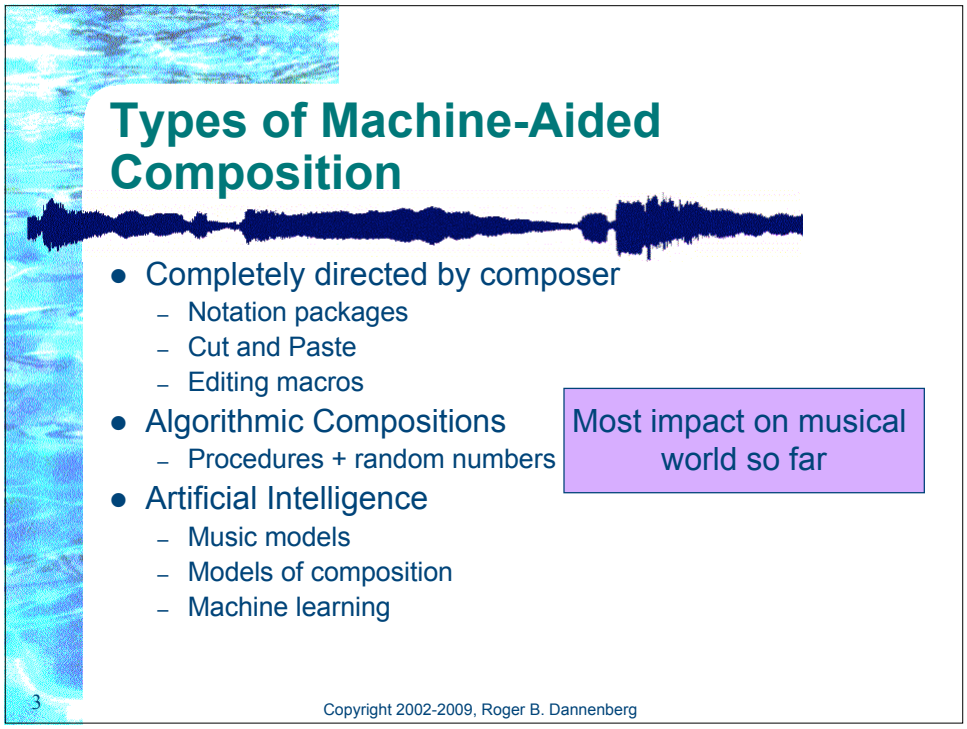
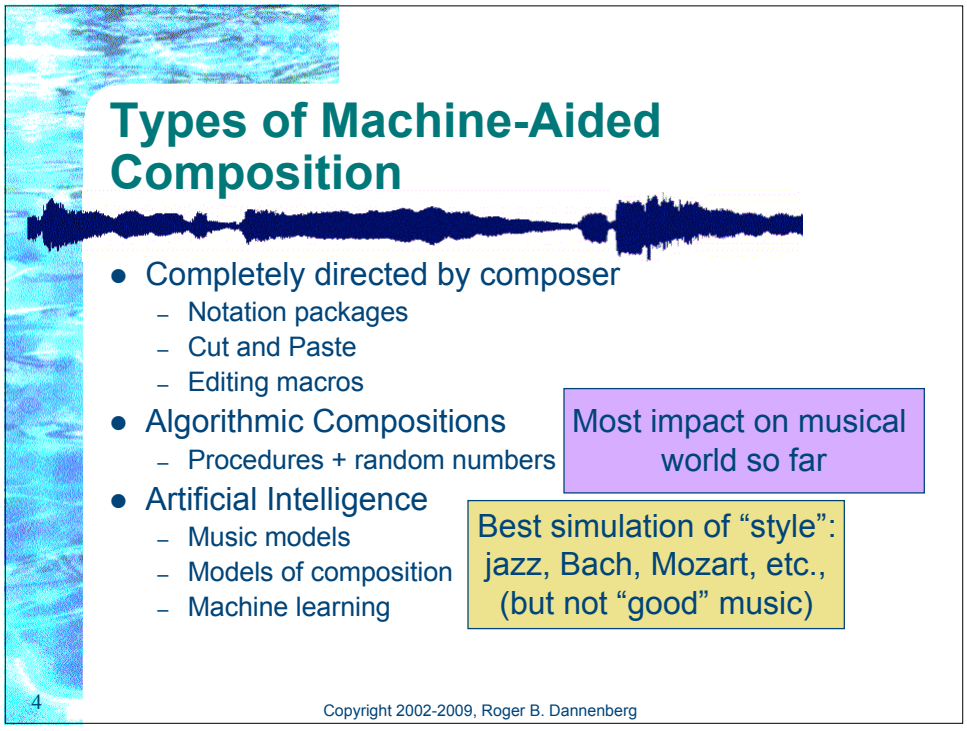
Introduction to Computer Music
Week 6
Algorithmic Composition

15-322
Roger B. Dannenberg
Research Associate Professor of Computer Science and Art, SCS & CFA

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


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
Most impact on musical world so far
3
Copyright 2002-2009, Roger B. Dannenberg


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
Most impact on musical world so far
Best simulation of "style": jazz, Bach, Mozart, etc., (but not "good" music)
4
Copyright 2002-2009, Roger B. Dannenberg

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>)
- Keyword Parameters
- 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 interarrival 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.

Look at first code example: ne-test

Probability distributions in Nyquist

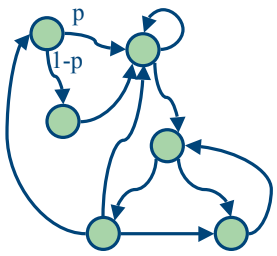
- load “distributions”
- See Distributions, probability in Nyquist index
- Look at second code example.

Melody Using Random Walk

- What kinds of pitches create interesting melody?
- Uniform random pitch has too many large intervals (try it).
- Lots of small intervals is more typical. (try it)
- Melodies are said to have fractal properties.

Markov Algorithm

- Generate sequence of "states"
- Probability of being in a state depends only upon probability of being in previous state (First Order)
- Or previous 2 states (Second Order)
- Etc.
- See example code



Rhythmic Pattern Generation

- Of course there are many techniques; here's one:
  - Generate a sequence length from some probability distribution or just by your choice
  - Generate a random number with that number of bits, e.g. length N → (random 2^N)
  - Translate 0 to rest, 1 to event
- Example code

Serialism

- Arnold Schoenberg and Serialism
- Chromatic scale – 12 notes/octave with equal ratios between (half)steps
- Pitch – an element of the chromatic scale
- Pitch class – pitch mod 12, e.g. “C-sharp” without regard to octave
- Tone row – permutation of the 12 pitch classes

17

Melodic/Tone Row Transformation

- Original: p[i]
- Transposition: T(p[i],c) = (p[i] + c) mod 12
- Inversion: I(p[i]) = (-p[i]) mod 12
- Retrograde: R(p[i]) = p[12 – i]
- Also: (p[i]*5) mod 12 = I(p[i]*7)

18

Why serialism?

- In general, listeners cannot hear retrograde and/or inversion relationships
- Intervals are preserved
- Tone “row-ness” is preserved
- “Denial of tonality” produces new textures

19

Fractals and Nature

- Melodic contours are often fractal-like
- Composers often use fractal curves to generate music data
- Examples:
  - Austin, Canadian Coastline
  - Cage, Atlas Eclipticalis

20

Grammars

melody ::= intro middle ending
middle ::= phrase | middle phrase
phrase ::= sequence-a | sequence-b

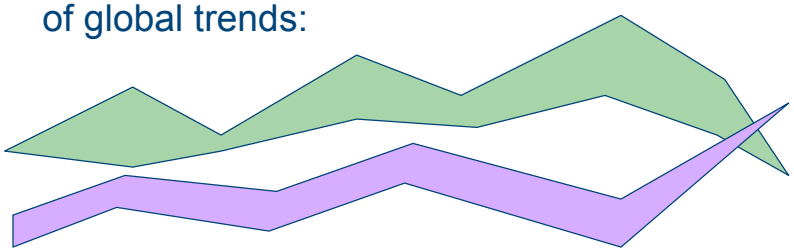
```
function melody()
   return seq(intro(), middle(), ending())
function middle() return #?(random() < 0.5,
                     phrase(), seq(middle(), phrase()))
function phrase() return #?(random() < 0.3,
                     sequence-a(), sequence-b())
```


Pitch and Rhythm Grids

- Quantize random numbers to scales, grids
- Program example
- Note that the last example, triads, might sound more natural if repetitions were eliminated – the patterns approach is more appropriate

Tendency Masks

- A problem with algorithmic composition is that things can get static
- Manual parametric change allows composition of global trends:



Listening

- The Silver Scale, Newman Guttman 0:24
- Study No. 1, David Lewin 1:36
- Eight Tone Canon, John R. Pierce 3:53
- The Days Are Ahead, Charles Dodge (tracks 8-13)
- Exerpt of work by John Chowning: Phone (track 38)
- Gravity, Jonathan Norton 1:00
- OTOdeBLU, Clarence Barlow 3:33